Machine Learning for Mental Health Detection

A Major Qualifying Project

submitted to the faculty of Worcester Polytechnic Institute

in partial fulfillment of the requirements for the Degree of Bachelor of Science

in Computer Science

Submitted by:

Jerry Assan

Maurice Flannery

Yufei Gao

Adonay Resom

Yuxin Wu

Advised by:

Elke A. Rundensteiner

Date: March 21, 2019

Abstract

Our project goal was to develop a depression sensing application that leverages multi-modal data sources collected from a smartphone, focusing on features extracted from audio, text messages, social media data, as well as GPS modalities. We conducted extensive experiments to study the effectiveness of these features to improve our machine learning model. We deployed our EMU app on Amazon Mechanical Turk for crowd-sourced data collection and incorporated feature extraction techniques and machine learning algorithms to reliably predict levels of depression.

Acknowledgments

The success of our Major Qualifying Project (MQP) would not have been possible without the help of many individuals. Firstly, we would like to thank Professor Elke Rundensteiner, and graduate student mentors Ermal Toto and Monica Lauren Tlachac for their continuous support and guidance throughout the project. In particular, we thank Ermal Toto for the extensive help with putting together the machine learning pipeline, development of the back-end server, and general software support. And we thank Monica Lauren Tlachac for her general advice as well as for her suggestions and ideas related to text message related feature extraction in addition to support for the identification of medical literature in this area.

We would also like to express gratitude towards our university, Worcester Polytechnic Institute (WPI), specifically the Computer Science department, for the logistical support and permission to carry out our experiments, while also providing us with funding that allowed us to conduct our study.

A big thanks also goes to graduate student Walter Gerych for helping with GPS feature extraction, and Eirs Wu for helping with the design of EMU's logo. Also, we thank Damon Ball, Ada Dogrucu, Anabella Isaro, and Alex Perucic, whose prior MQP work, advised by Professors Emmanuel Agu and Elke Rundensteiner in the previous academic year, served as the starting point for our MQP. Our MQP would not have reached this point if it was not for the countless recommendations and guidance in solidifying our project.

List of Figures

List of Tables

Table of Contents

1. Introduction

Depression is one of the most common mental disorders in the U.S. It causes severe symptoms that negatively affect how people feel, think, and handle daily activities. To be diagnosed with depression, the symptoms must be present for at least two weeks (The National Institute of Mental Health, 2018). These symptoms may include feelings of sadness, tearfulness, emptiness or hopelessness and many other signs (Mayo Clinic Staff, 2018). About 9.5 million people experience depression in the U.S. each year. According to the World Health Organization, by 2020, depression will become the world's second largest medical burden (The National Alliance on Mental Illness, 2018). Although depression is so common, it is often over-detected or under-detected by doctors. Currently, less than 25% of people with depression receive treatment (The National Alliance on Mental Illness, 2018). A study about clinical diagnosis of depression found out that general practitioners only correctly identified depression in 47.3% of 50,371 patients who were examined across 41 other studies. There are even more false positives than either missed or identified cases (Mitchell, Vaze, & Rao, 2009).

Recently, more researchers have begun to use machine learning to help doctors diagnose symptoms of depression while being both accurate and as effortless as possible. Using machine learning for mental health detection can prove to be an excellent technique to detect depression as it is more convenient and can be discovered much faster. Under the hood, there are many factors that go into determining whether or not someone is depressed, and studies have shown that various types of data including voice patterns, text post data, GPS data, social media data, and facial expressions are all related to depression (Ball, Dogrucu, Isaro, & Perucic, 2018). Several voice acoustic measures are significantly correlated with depression severity (Mundt, Snyder, Cannizzaro, Chappie, & Geralts, 2007). A recent work reports that depression is associated with vowel space in speech (Scherer, Lucas, Gratch, Rizzo, & Morency, 2016). Moreover, text patterns such as the usage of first-person pronouns, social references words, negatively and positively valenced words are also correlated with depression (Rude, Gortner, & Pennebaker, 2004).

Ball et al. (2018) worked on a similar project and created a mobile application, named Moodable, that aimed to detect depression by collecting these types of data from a patient’s

smartphone. This application was designed to give people an on-the-spot depression rating on how severe depression symptoms they show (Ball et al., 2018). Our project goal was to develop a new application, named EMU, that improves the performance of the Moodable application and collect more data for our and further studies, and then lastly to conduct a much more comprehensive experimental study using machine learning to improve depression detection. To achieve our project goals, we addressed some key objectives. Through background research, we have determined two major aspects to focus on: audio analysis and text analysis. Further, we advanced feature extraction and generation capabilities for audio analysis, text analysis, and also for GPS data analysis. We then repeated the baseline machine learning experiments using the k-nearest-neighbors, support vector machine, and random forest machine learning algorithms on the Moodable dataset collected by the prior MQP team. with previous data. After mastering these basic machine learning concepts, we implemented and then applied a variety of more advanced machine learning algorithms, including XGBoost, Adaptive Boosting, decision trees, logistic regression, artificial neural networks, Gaussian processes, and more. We also improved the mobile EMU application itself in several significant ways, both to make it more human-computer friendly and to have more secure storage and security during data collection and subsequent transmissions. Thereafter, we conducted a second study to collect additional test data in order to test the performance of the improved EMU application. Finally, we built the EMU website for this project with the ultimate objective of sharing our refined data and results, as appropriate.

2. Background

Depression as a common mental health disorder is on the rise globally (World Health Organization, 2018). A previous MQP team developed an application for doctors to effortlessly detect depression, and in effect, achieve greater coverage in detecting depression over the general population. Literature reviews found that text and voice features contain sufficient information to indicate people's emotion or mental state, since they carry statistical correlations with depression across many studies.

2.1 Summary of the Previous MQP

The previous MQP team deployed a mobile application that can be used to collect user data (texts, social media content, geospatial data, and voice samples that are two weeks prior to the point when they give consent to the application) on the spot. Simultaneously, these data go through a machine learning pipeline that can introduce an evaluation of the state of the user's mental health. Rather than having to wait for a certain amount of time for the application to obtain data as of previous approaches, such application enables the doctor or medical workers to inquire almost real-time feedback on their patients. Yielding an average test set root-mean-square error (RMSE) of 5.67 in predicting the PHQ-9 score, this approach demonstrates a simple and intuitive way to diagnose depression (Ball et al., 2018).

The team firstly ran an exploratory “willingness to share data” study to determine what information participants feel comfortable giving to a member of medical staff. Their results showed that people were most willing to share their microphone data (say a phrase into a microphone) and images of their face. For all the other types of data, including text, social media, GPS, call logs, and browser history, about 40% to 50% of the participants were willing to share those kinds of data.

After the team learned what information would the patients feasibly allow them to obtain, they developed an Android application, which takes and gathers the personal information from the participants’ phone. The information included the GPS data, the text message, the message from Instagram, the sleep pattern, and the audio sample. With the different kinds of information,

they extracted some features, like pause time speaking rate from the audio sample, number of syllables from the text message, and so on.

Meanwhile, all the participants were asked to fill out a questionnaire, called PHQ-9, which is a multipurpose instrument used by medical professionals to screen, monitor, and measure the severity of depression. The participant need to answer every question of the below PHQ-9 form and combine the scores of all the answers indicate. The combined score is the PHQ-9 score of the participant. Scores of 5, 10, 15 and 20 are the cut-points of mild depression, moderate depression, moderately severe depression and severe depression. The benefit of PHQ-9 is that it provides more information on individual depression symptoms for medical professionals or us to analyze the severity of all participants. According to the questionnaires, the previous group got the PHQ-9 scores of all the participants.

PATIENT HEALTH QUESTIONNAIRE-9 (PHQ-9)

Over the last 2 weeks, how often have you been bothered by any of the following problems? *(Use "✔" to indicate your answer)*	**Not at all**	**Several days**	**More than half the days**	**Nearly every day**
1. Little interest or pleasure in doing things	0	1	2	3
2. Feeling down, depressed, or hopeless	0	1	2	3
3. Trouble falling or staying asleep, or sleeping too much	0	1	2	3
4. Feeling tired or having little energy	0	1	2	3
5. Poor appetite or overeating	0	1	2	3
6. Feeling bad about yourself — or that you are a failure or have let yourself or your family down	0	1	2	3
7. Trouble concentrating on things, such as reading the newspaper or watching television	0	1	2	3
8. Moving or speaking so slowly that other people could have noticed? Or the opposite — being so fidgety or restless that you have been moving around a lot more than usual	0	1	2	3
9. Thoughts that you would be better off dead or of hurting yourself in some way	0	1	2	3

Figure 1: The PHQ-9 Questionnaire (Ball et al., 2018)

Then, the previous group used these features and the PHQ-9 scores to train the machine learning systems. They used 85% of the data as the training set and the rest as the testing set to test the accuracy of the machine learning systems.

Finally, the previous group's Android application was deployed with the trained machine learning systems, and gathered data from the participants' phones and analyze the severity of depression of the participants.

Simultaneously, the previous group hoped that the future developers of the application develop the function of facial image analyze to increase the accuracy. Another area for improvement is in the machine learning systems, which is also what we aim to achieve. Along with that, more data sets and more complete data are required. The previous team also urged any future groups to gather more data to train the machine learning systems (Ball et al., 2018).

2.2 Background on Depression

Certain factors, such as age and gender, influence the rate of depression among people. Depression occurs across different races, genders, ages, and socioeconomic statuses. Among these, age can be a deciding factor in associating to the rate of depression. Furthermore, within the same age groups, depression incidents happen somewhat more on women than on men. However, such phenomena do not necessarily predict depression by any means, but rather, as depression is a complex biological and social issue, a subset under a complex system.

The table below shows some forms of depressive disorders, which were mentioned in the report of the previous MQP group.

Disruptive Mood Dysregulation	Most common type of depression amongst children of age 12, and is characterized by persistent irritability and frequent episodes of extreme behavioral decontrol.
Major Depressive Disorders	Often associated with considerable morbidity, disability and an elevated risk of suicide and is categorized into two types of chronic and non-chronic major depressive disorders.

Persistent Depressive Disorders	Persistent depressive disorder is a consolidation of chronic major depressive disorder and dysthymic disorder, but differ in occurrences.
Premenstrual Dysphoric Disorder	A more severe form of premenstrual syndrome that affects about 50 - 80% of women in their menstruation cycle. Symptoms such as irritability, dysphoria, muscle pain and insomnia can be essential features of the premenstrual dysphoric disorder.
Substance/Medication-Induced Depressive Disorders	Refers to the specificity of the substance causing the depressive symptoms.

Table 1: Some Forms of Depressive Disorders (Ball et al., 2018)

This next table shows some clinically validated measures of depression, which were also covered in the previous MQP group's report.

Screening tools	Number of items	Scoring time	Psychometric properties	Cost and development
Hamilton Depression Rating Scale (HDR)	21	15 to 20 min	Sensitivity: 93% Specificity: 98%	Free
Beck Depression Inventory (BDI)	21	5 to 10 min	Sensitivity: 84% Specificity: 81%	Proprietary ($115/kit)
Patient Health Questionnaire (PHQ)	9	< 5 min	Sensitivity: 88% Specificity: 88%	Free with permission
Major Depression Inventory (MDI)	10	< 5 min	Sensitivity: 86% Specificity: 82%	Free

Table 2: Some Clinically Validated Measures of Depression (Ball et al., 2018)

These tools help people to determine the severity of depression and the diagnosis of depression. In our project, we will use PHQ (also called PHQ-9) as the tool to predict depressive behavior.

2.3 Detecting Depression from Text

The text of messages is always an important indicator for depression. The sentence structure and content are two main things we will analyze.

2.3.1 Sentence Structure

The first of two factors worth considering in text analysis is how a sentence is structured. More simply, arrangement, syntax, grammar choice, and spelling, among other features. The more relevant features for the analysis of text are the number of words, word frequency, the number of syllables, the word case (capitalization), and punctuation usage (Morales & Levitan, 2016). Our word choice can be used to get an idea of how we are feeling at the time of writing a text.

Directly considering a sentence's word count doesn't seem too useful alone, but when used in conjunction with the analysis of content features and word frequencies, we can determine how much of the sentence contains words from certain categories. When used with syllable count, we learn just a little bit more about what's being said. This helps in misleading situations where, for instance, a sentence may be short, but uses words with more syllables and is more useful to us than a simple "yes okay" statement. When it comes to word case and punctuation, we are able to detect the strength of the topic being discussed. A word in ALL CAPS provides emphasis, and punctuation, namely the exclamation point, indicates an increase of intensity without altering the words in the sentence (Hutto, 2015) (e.g., "That lecture was interesting." versus "That lecture was interesting!").

A final clarifying notes about analyzing sentences: When analyzing text, we don't have to look at one sentence at a time. A sentence is just the minimal unit, and we are able to do all of the things stated above on multiple sentences, on paragraphs, on autobiographies, and essentially all kinds of forms of text.

All features of sentence structure of text will be discussed in detail in Section 3.1.4.

2.3.2 Content

The second factor we want to look at is the content. Specifically, we want to know what is being talked about and how the subject is choosing their words. We'll be considering self-references, social words, positive emotions, and negative emotions. (Morales & Levitan, 2016). The big takeaway from these features is that we want to predict signs of depression, using text samples from the subject.

The amount of occurrences of references to the self is indicative of depressed behavior (Rude et al., 2004). This makes sense, as someone who is going through a hard time would be inclined to talk a lot about what is going wrong with *them*; how *their* day is going; how *they* feel about something that has happened, overall being more self-focused (Tausczik, & Pennebaker, 2009). Extraverted persons tend to use more social words, as well as express more positive and less negative emotions (Tausczik, & Pennebaker, 2009). And it's no surprise that those who do not show symptoms of depression use more positive and less negative words in their everyday life, while conversely, those who either appear or are depressed via diagnosis use a more negative and less positive tone in their conversations with others.

All features of content of text will be discussed in detail in Section 3.1.3.

2.4 Detecting Depression from Voice

Recent research has found that features of human voice play an important role in depression prediction. In particular, speaking style, vowel space, and speech pause time.

2.4.1 Speaking Style & Vowel Space

For speaking style, a study of depression detection concludes that comparing with reading speech, spontaneous speech has more variability, which increases the recognition rate of depression (Alghowinem et al., May 2013). Another study also suggests that data collections for depression diagnosis or prediction should include spontaneous speech to get more accurate results, rather than only include read speech (Mitra & Shriberg, Apr 2015).

Vowel space is defined as the frequency range spanned by the first and second formant of the vowels (Scherer, et al., 2016). In Scherer, et al.'s journal, they found that people suffering

from depression tend to have reduced vowel space, which is often reported when comparing speech affected by depression to speech from healthy subjects (Cummins, Sethu, Epps, Schnieder, & Krajewski, 2015).

2.4.2 Speech Pause Time and Speech Rate

Significant correlation has been found between depression and speech pause time (SPT) in recent years. In fact, SPT is often used as a depression indicating biomarker in clinical studies. Clinicians can even intuitively sense the change in SPT across patients in order to detect their emotional state without the need of any analytical tools. We will discuss SPT related features we used in our study in details in Section 3.2.2.

In addition, scholars have produced reports that show strong statistical correlation between SPT and depression measures/scales. Stassen et. (1998) discovered the SPT and Hamilton Rating Scale for Depression (HAMD) score of 60% her patients were positively correlated. Cannizzaro et al. (2004) discovered that patients with reduced speaking rates (number of pauses) showed increased level of depression on HAMD scores. Mundt et al. (2012) learned that different properties of pause time (such as percent pause time and speech to pause ratio) correlated strongly with HAMD scores. Hong et al. (2014) revealed a positive correlation between average syllable duration and depression.

Furthermore, studies have shown that the change in the length and frequency of pause can detect different types of depression. Alpert et al. (2001) split depressed patients into groups with agitated and retarded forms of depression. They discovered that the group with retarded depression had briefer pause times but longer utterances while the group with agitated depression exhibited longer pauses and shorter utterances.

2.5 Detecting Depression from GPS Data

A recent study found that GPS data or mobility data sponsored by smartphones can predict depression with good accuracy (Farhan et al., 2016). In their study, for Android, raw GPS data such as longitude and latitude information are collected through Android's location services, and activity data is sensed using Google's Activity Recognition API (Farhan et al., 2016). They categorized mobility features into three types, features based on raw GPS data, features based on

location clusters, and features based on activity data. The features they extracted include location variance, time spent in moving, total distance, average moving speed, number of unique locations, entropy, normalized entropy, time spent at home and percentage of time in a state. In Canzian and Musolesi's study, they used similar features such as the total distance covered and the number of different places visited (Canzian & Musolesi, 2015). Inspired by their work, we also used some of those features in our project. These features will be discussed in detail in Section 3.3.

3. Feature Engineering Methodology

To be more specific than "machine learning on text, audio, and GPS data," the methodology will go into great detail about how we extract features from those data, and what tools and scripts are used. Moreover, we will explore several machine learning algorithms and apply them to the features that will be extracted from the existing data.

3.1 Text Features

3.1.1 Feature Selection

In the report, *Language use of depressed and depression-vulnerable college students*, Stephanie Rude, Eva-Maria Gortner and James Pennebaker found that the number of occurrences of "I," used in the text, is associated with the mood of the writer. The figure below shows the statistics of the report.

Means and (standard deviations) on depression measures and linguistic dimensions for the three diagnostic groups

	Currently-depressed (n = 31)	*Formerly-depressed* (n = 26)	*Never-depressed* (n = 67)
	Mean (SD)	*Mean (SD)*	*Mean (SD)*
Age	17.97 (0.41)	18.96 (2.82)	18.78 (4.03)
BDI	20.05 (6.39)	4.17 (1.36)	3.58 (1.37)
IDD-L	25.60 (17.21)	40.59 (13.14)	1.18 (2.21)
Linguistic dimensions			
First person singular (I, me, my)	12.17 (2.91)	10.76 (3.10)	10.76 (2.51)
''I'' only	8.50 (2.22)	7.88 (2.15)	7.27 (2.02)
Negatively valenced words	2.92 (1.26)	1.70 (0.80)	1.63 (0.91)
Positively valenced words	2.64 (1.28)	3.49 (1.30)	3.12 (1.45)
Social words	6.32 (2.17)	5.89 (2.53)	5.78 (2.83)

Figure 2: Statistics of the Report (Rude et al., 2004)

All the data from these statistics are collected from the result of an experiment that was held by Rude et al. First of all, all the participants of the experiment were classified into 3 groups

of people by their scores of Beck Depression Inventory (BDI) and Inventory to Diagnose Depression, Lifetime Version (IDD-L). The person, with a BDI score higher than 14, was classified as currently depressed. Among the rest people, the person, with IDD-L score lower than 9, was classified as never depressed, and the person, with IDD-L score higher than 25, was classified as formerly depressed. After that, all the participants were asked to write a paragraph related to a specific topic. Then, all their paragraphs were analyzed to get the statistics above.

They found that the number of the first-person singular (I, me, my) words used by the currently depressed people were greater than the same kind of words used by the never depressed person. Moreover, the difference between the number of first-person singular words was mainly caused by the frequency of the pronoun "I" (Rude et al., 2004).

The number of the first-person singular is also considered to be an indicator of depression (Zimmermann, Brockmeyer, Hunn, Schauenburg, & Wolf, 2017). As a result, the number of first-person singular words is a reliable feature of text analysis for our project. More obviously, more currently-depressed persons tend to use more negative emotion words than someone who has never been depressed (Rude et al., 2004).

The degree modifiers always play an important role in the text. For example, "lonely", "a little bit lonely", and "very lonely" are very different in showing the severity of depression. Thus, the degree modifiers should also be considered to keep the precision of the result. To achieve this function, we consider a library of all degree modifiers and combine it with other features to get a better and more reliable result (Wang, et al., 2013).

3.1.2 Data Preprocessing

3.1.2.1 Data Pretreatment

We group all the data in the database by the users' ID via Python script. Since we are focusing on the extraction of text features, we are only interested in the data which is of type "phq" or "text". The "phq" data contain the Patient Health Questionnaire (PHQ) score for the user, which indicates whether or not, and if so how much, the patient is depressed. The text data contain the text that has been received by the user, which is what we need to extract the text features.

The tables below show an example that the data before and after being grouped.

id	type	content
1111	phq	10
1111	text	text1
1111	text	text2
1111	tweet	tweet1
2222	phq	9
2222	audio	audio1

Table 3: Data Before Grouped

id	content
1111	phq:10; [text1, text2]

Table 4: Data After Grouped

3.1.2.2 Spam Removal

To increase the effectiveness and accuracy of the extraction of the text features, our team decided to remove the spam in the database. For us, spam mainly means messages sent by business, messages trying to sell something and notifications. We had tried to use Naive Bayes spam filtering to remove the spam. However, the Bayes filtering is too sophisticated for the problem. Because of this, our team tried to use other methods to remove the spam. The first method we used is the keywords method. The keywords method deletes the text which contains the specific words or sentences (like “PLEASE REPLY XXX-XXX” or “PLEASE DOWNLOAD XXX”). The second method we used is the address check method. This deletes the text which is sent by a suspicious address (phone number), such as ones that start with a letter instead of a number. However, we failed to implement this method in our project.

3.1.3 Features Based on Content

3.1.3.1 Frequencies and Volume

For the frequencies of a particular category, we count the number of the words per text message that fall in that category, and then divide by the total number of words in that text message. Additionally, we create a volume feature by counting the number of texts a user receives. These frequencies and volumes are then regarded as text features!

3.1.3.2 The Speech Tag Frequency

The speech tag frequency means the frequency of different kinds of words for every user. The table below shows all the 33 kinds of tag and their abbreviations and examples.

Tag	Description	Example
CC	conjunction	and, but
CD	cardinal number	seven, 16%
DT	determiner	a, these
EX	existential there	**there** is a boy
FW	foreign word	tsunami
JJ	adjective	big, small
JJR	adjective, comparative	bigger, smaller
MD	verb, modal auxiliary	may
NN	noun, singular or mass	child
NNS	noun, plural	children
NNP	noun, proper singular	God
NNPS	noun, proper plural	we met two **Christmases** ago
PDT	predeterminer	**either** his children
RB	adverb	loudly
RBR	adverb, comparative	better

RBS	adverb, superlative	best
RP	adverb, particle	about
SYM	symbol	%
TO	infinitival to	when **to** go
UH	interjection	gosh
VB	verb, base form	grow
VBZ	verb, 3rd person singular present	it **grows**
VBP	verb, non-3rd person singular present	I **grow**
VBD	verb, past tense	grew
VBN	verb, past participle	grown
VBG	verb, gerund or present participle	growing
WDT	wh-determiner	which
WP	wh-pronoun, personal	who
WP$	wh-pronoun, possessive	whose
WRB	wh-adverb	where
IN	conjunction, subordinating or preposition	on, of
PRP	pronoun, personal	you, me
PRP$	pronoun, possessive	your, my

Table 5: All Tags of Speech Tag (CLiPS Research Center, 2010)

3.1.4 Features Based on Sentence Structure

Two scores are considered when it comes to sentence structure: polarity score and subjectivity score. Polarity score is calculated according to the words and the sentence structure

used by a user. It is a float within the range [-1.0, 1.0]. The higher the score is, the more positive that particular segment of the user's text is. Subjectivity score is calculated according to the words and the sentence structure used by a user. It is a float within the range [0.0, 1.0] where 0.0 means the user's text is very objective and 1.0 means it is very subjective.

3.1.5 Feature Extraction

In order to get the text data in a format we can work with, we first grouped all texts by participant ID, and ran each ID through the tools mentioned in the following subsections. After getting results, we combined all three tools' output into one centralized CSV file for easy access and readability.

3.1.5.1 Empath

Empath is a tool developed by a PhD student at Stanford University that allows us to extract features across lexical categories. We passed in the data and enabled normalization, which will return the percentage of the text in each category as opposed to the number of words in each category. The output format of Empath is a JSON file; this had to be converted to a CSV file via Python script.

One thing unique to Empath is category creation. This allows us to extract features from categories that the tool doesn't extract out of the box. We made use of this to extract features in an acronym category, which encompasses things like "lol", "omg", "idk", and more.

More information about Empath can be found in Ethan Fast, Binbin Chen, and Michael S. Bernstein's *Empath: Understanding Topic Signals in Large-Scale Text*, including access to the tool itself (Bernstein, Chen, & Fast, 2016).

3.1.5.2 Linguistic Inquiry and Word Count

The Linguistic Inquiry and Word Count (LIWC) tool is a widely-accepted tool for text analysis and using its application program interface (API), we were able to run the data and receive output in the form of a JSON file, which also had to be converted to a CSV file.

LIWC is very special in the sense that it does exactly what we are looking for in regards to text analysis. It that counts words in psychologically meaningful categories (like an emotion

dictionary). It detects positive and negative words, words referencing other people, and "pronouns which can capture inclusive language (us, we) vs. exclusive language (you, they, them), and words referencing how the person is feeling (sad, anxious, sleep)", to name a few (Morales, 2018).

3.1.5.3 TextBlob

We mainly are interested in two functions of TextBlob to extract text features. One is Sentiment Analysis and the other is Speech Tag. Sentiment Analysis enables us to get the polarity score as well as the subjectivity score of a sentence or a user. The table below shows how Sentiment Analysis works.

Text	Polarity	Subjectivity
I am so happy.	0.8	1.0
I am so sad.	-0.5	1.0

Table 6: Examples of Sentiment Analysis 1

Since most users have more than one text message, we also did tests to figure out how TextBlob works when there are multitudes of sentences instead of just one sentence and found out that TextBlob calculates the average scores of the sentences as the final result.

The table below shows how Sentiment Analysis analyzes a paragraph which contains several sentences.

Text	Polarity	Subjectivity
I am so happy.	0.8	1.0
I am so sad.	-0.5	1.0
I love this world.	0.5	0.6
I am so happy. I am so sad. I love this world.	0.26666	0.86666

Table 7: Examples of Sentiment Analysis 2

After concatenating all the texts of a user into a paragraph, we then analyze the polarity score and the subjectivity score of a user, and take the two scores as two text features.

3.1.5.4 Manual Extraction

There are some text features that we've decided to extract manually, such as the frequency of certain words, and the volume of a user's texts. For the frequency, it is as simple as iterating through the texts and counting the occurrences of the desired word. For volume, we count the number of texts per ID and make that its own feature.

The reason we've chosen to do it this way is because none of the tools we're using have the capability to do these things, but they're important enough to take the time and manually calculate. Both of the above two features are easily calculated with a Python script created and maintained by us.

3.2 Audio Features

3.2.1 Feature Selection

A previously conducted MQP study (Ball et al, 2018) generated 1583 audio features using OpenSmile, a leading audio feature extraction tool (Eyben et al, 2013). Models were trained using features provided from a dynamic feature selection technique. The technique scans through all 1583 features generated by openSmile and suggests the top 40 features that it estimates as most correlated with depression. In this study, instead of using dynamic feature selection for audio, we plan to test and use specific biomarkers as audio features in order to improve prediction results as measured by Accuracy, Precision, Sensitivity and F1 scores. There are numerous published studies that have closely examined features that demonstrate sufficient correlation with measurements of depression (PHQ-9, HAMD, BDI-II, etc…) (Morales, 2017). For example, scholars such as Cannizzaro et al. (2004), Mundt et al (2012), and Honig et al (2014) ran studies that performed statistical analysis on acoustic measures such as speaking rate, pause time, and pitch variation. We will test these research approved features against features from the selection process in order to compare their accuracy and performance.

3.2.2 Features Based on Speech Pause Time (SPT)

Several features can be generated from acoustic measures related to speech pause time and/or speaking rate.

James C. Mundt et al. gathered controlled voice data from depressed patients and produced the following table that shows the correlation between acoustic measures (including pause time) and the Hamilton Rating Scale for Depression (HAMD).

VA measure	Reliability[b]	HAMD Association[c]	*p*
F_0 COV[a]	.61	-.01	*ns*
F_1 COV[a]	.24	-.03	*ns*
F_2 COV[a]	-.02	-.17	*< .05*
Total recording length of speech sample	.57	.20	*< .01*
Total vocalization time	.66	.07	*ns*
Total pause time	.71	.29	*< .001*
Number of pauses	.71	.10	*ns*
Pause variability (St. Dev.)	.64	.38	*< .001*
Percent pause time	.04	.18	*< .01*
Vocalization/pause ratio	.48	-.22	*= .001*
Speaking rate (syllables/sec)	.65	-.23	*< .001*

[a]Coefficient of Variation
[b]Cronbach's alpha, assessed across speech tasks within each telephone call
[c]Pearson's correlation

Table 8: Correlation between SPT and HAMD (Mundt, 2017)

Acoustical measures F0 COV, F1 COV, and F2 COV represent pitch variability across different frequency bands and are not related to pause time (Mundt, 2017). Therefore their negative reliability and association with HAMD scores are irrelevant. However, the table shows a significant correlation between HAMD scores and total pause time, recording duration, ratio of vocalization to pause, pause duration variability and speaking rates. This proposes that subjects that with higher HAMD scores took longer to accomplish speaking tasks. Speaking duration did not increase because of speech content but was instead a result of large number of pauses, longer pause duration, slower speaking rates and lower vocalization to pause ratios. Therefore, all the

acoustic measures, except for F0 COV, F1 COV, and F2 COV, in the table can be used as features that help detect depression.

The correlation to depression of the acoustic measures/features listed above change depending on the type of speech production tasks used by the test subjects. For example, total pause time best correlated to depression ($r=0.27$, $p<0.01$) during automatic speech tasks such as counting or reading, and pause time ($r=0.21$, $p<0.01$) better correlated to depression during free speech tasks. On the contrary, pause variability and vocalization to pause ratio had a better correlation with depression during free speech tasks ($r=0.34$, $p<0.001$) and ($r=0.22$, $p=0.001$) than with free speech tasks ($r=0.3$, $p<0.001$) and ($r=-0.18$, $p<0.01$). Therefore, different acoustic measures should be used as different combination of features while using different type of speech tasks.

The following is a list of features that are related to pause time and speech rate. Features are supported by research work from Mundt et al (2017), deJong et al (2009), and Klára et al (2012):

3.2.2.1 Number of Pauses

This feature is a count of pauses found in between the subject's speech. Pauses were determined at a silence threshold of -25dB.

3.2.2.2 Number of Syllables

This feature is a count of the number of syllables that found in between the subject's speech. Syllables were detected by the number of voiced counts in between pauses detected.

3.2.2.3 Total Recording Length

This feature is calculated as the amount of time it took the subject to complete the entire speech task successfully.

$$\textbf{Vocalization time} = \sum_{x=0}^{n} \textbf{(time for syllable)}_{\textbf{n}}$$

3.2.2.4 Vocalization Time

This feature is calculated by adding the amount of time it took to complete each syllable during the subject's speech task. In other words, it's the total time that the subject was producing sound during the entire recording session.

3.2.2.5 Total Pause Time

This feature is calculated by adding the amount of time that the user took during each pause.

$$\textbf{Pause time} = \sum_{x=0}^{n} \textbf{(time for pause)}_{\textbf{n}}$$

3.2.2.6 Speech Rate

This feature calculates the speed at which the subject speaks. The number of syllables a subject's produces over the entire recording time represents the speaking rate of the subject.

Speech Rate = Number of syllables / Total recording length

3.2.2.7 Articulation Rate

The articulation rate is a calculation of speech rate where are the pauses are excluded during the calculation process.

Articulation Rate = Number of syllables / Vocalization time

3.2.2.8 Pause Variability (σpause time)

The is a calculation of the standard deviation between all the pause times detected during the subject's speech task.

$$\sigma \textbf{ pause time} = \sqrt{\frac{\sum |x - \bar{x}|2}{n - 1}}$$

3.2.2.9 Percent Pause Time

This feature is the ratio of time spent pausing to the time it took to complete the speech task.

Percent pause time = Total pause time / total recording length

3.2.2.10 Average Syllable Duration

This feature calculated the average amount of time it takes a subject to finish one complete syllable.

Average syllable duration = Vocalization time / number of syllables

3.2.3 Features Based on Signal Analysis

3.2.3.1 Jitter

This feature is a measure of frequency instability or periodicity of vocal fold vibration. It is calculated to be the absolute difference between consecutive time periods (T) in speech divided by the average time period, where N is the number of periods and T is the length of the periods. (Klára et al 2012).

$$jitter = \frac{\Sigma_{i=2}^{N-1}|2T_i - T_{i-1} - T_{i+1}|}{\Sigma_{i=2}^{N-1}T_i} \cdot 100[\%]$$

3.2.3.2 Shimmer

This feature is a measure of amplitude instability. It is calculated to be the average absolute difference between consecutive differences between the amplitudes of the consecutive periods.

$$shimmer = \frac{\Sigma_{i=1}^{N-1}|A_i - A_{i+1}|}{\Sigma_{i=1}^{N-1}A_i} \cdot 100[\%]$$

3.2.3.3 Mean Harmonics to Noise Ratio

This feature is the measure that quantifies the additive amount of additive noise in the voice signal. In other words it represents the degree of acoustic periodicity.

$$HNR = 10 * log\frac{E_H}{E_7}[dB]$$

3.2.3.4 Mean Noise to Harmonics Ratio

This feature is a measure of hoarseness.

3.2.3.5 Mean Autocorrelation

This feature helps detect repeating patterns in signals. Formally, it the measure of the delayed correlation of a given series. Meaning it represents the degree of similarity between the current time series and lagged versions of the current time series over consecutive time intervals. The definition of the autocorrelation between times s and t is

$$R(s,t) = \frac{\mathrm{E}[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s},$$

3.2.3.6 Fractional Locally Unvoiced Frames

This feature represents the fraction of pitch frames that are detected as unvoiced frames.

3.2.3.7 Number of Voice Breaks

This feature represents the distances between consecutive pulses that is longer than 1.25 divided by the pith floor.

3.2.3.8 Degree of Voice Breaks

This feature also closely resembles the pause percent time feature. It is the measure of the total duration of voice breaks that exist between vocal sounds.

3.2.4 Feature Extraction (Pratt)

OpenSmile extracts a vast array of audio features, however, the most important research approved features such as pause time and voice breaks are not contained in it. Therefore, another audio analysis tool was required. The tool we are using to generate new features is called Praat. Praat, developed at the University of Amsterdam, is a software package that focuses on speech

analysis. It is capable of pitch, formant, intensity and spectral analysis and can also detect excitation patterns and voice breaks.

Praat can either be used through its graphic user interface or command line interface. The graphic user interface produces detailed graphs of properties such as pitch, intensity and pulse over waves generated by the test subject's recording. This visual component can help us understand which features exhibit common properties and patterns among subjects that subjects that produce higher PHQ-9 scores. The following graph, generated by Praat, shows pitch, pulse, and intensity over a select pick of voice frames from a recording.

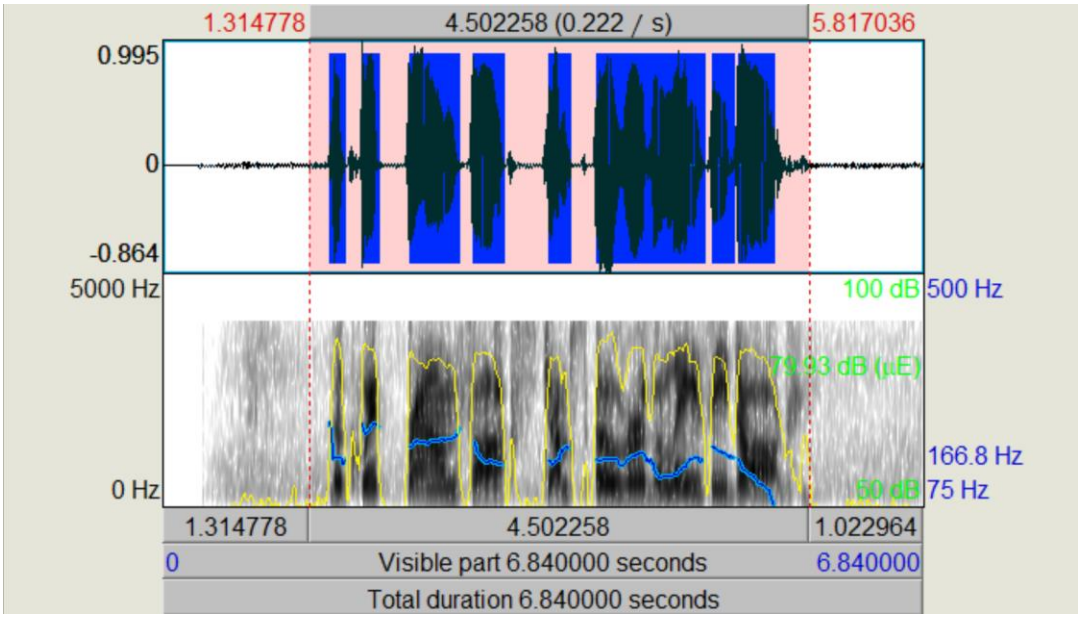


Figure 3: Praat GUI Analysis

Contrarily, the command line interface of Praat enables us to write custom scripts that generate feature data which is used by other machine learning or other analytic tools. In fact, Praat scripts are commonly used in studies that investigate voice source bio markers and depression (Wolters, 2015 and Mundt, 2007). In our study, we created a Praat script that takes a sound object and detects minute silences in order to categorize sound sections into intervals. Those intervals are then analyzed for properties such as pause time, number and degree of vocal breaks, fraction of vocally unvoiced pitch frames, and noise to harmonic ratio (which are features that are strongly associated to depression (Liu, 2017).

3.2.4.1 Extraction of Pause Time Features

Pause time features were extracted using a praat script. The script takes an input of the audio file and other numerical parameters that are used as configuration values. The configuration parameters include silence threshold, minimum pause duration and minimum dip between peeks. These inputs are all used during the pause detection process and are essential to improving the accuracy of the features generated. This is because most of the other features are generated by applying various equations and formulas that contain the number of pauses during speech and the amount of time that each pause (as shown above). For example vocalization time is the difference between total recording length and total pause time, average pause time is the ratio of total pause time to the number of pauses, and pause percent time is the ratio of total pause time to total recording length.

The script also tries to improve the accuracy of features generated by doing some extra audio analysis. For example, voiced count (the number of syllables) can be generated by decrementing the number of pauses detected by one. However, there could have been some external or internal factors that generated errors during the pause detection process, which in turn can ruin all other features generated. Therefore the script tries to detect any errors by attempting to detect number of syllables using built in praat syllable detection techniques and comparing it against the voiced count value generated by decrementing the number of pauses. The script used for detecting these features is attached at Appendix F - 1.

3.2.4.2 Extraction of Signal Based Features

OpenSMILE generates thousands of signal based features, however, some signal based features referenced in depression detection literatures were not found among the openSMILE features. Therefore, we used the praat analysis tool and script to extract those features instead. The praat script used to extract this feature set did not require us to deal with audio analysis details through scripting (unlike the pause time features). The script used for detecting these features is attached at Appendix F – 2.

The Praat script shown above (along with a smaller python script used to combine generated files) was used to process 265 .wav audio recordings. The process generated the following list of features and stored them in .csv format files:

Property	Features
Voicing	Fraction of locally unvoiced frames Number of voice breaks Degree of voice breaks
Harmonicity	Mean autocorrelation Mean noise-to-harmonics ratio Mean harmonics-to-noise ratio
Jitter	Local Local, absolute rap Ppq5 Ddp
Shimmer	Local Local, dB Apq3 Apq5 Apq11 Dda
Pitch	Median pitch Mean pitch Standard Deviation Minimum pitch
Pulses	Number of pulses Number of periods Mean period Standard deviation of period

Table 9: Audio Features Generated by Praat

3.2.5 Feature Extraction (openSMILE)

OpenSMILE feature extraction was done by the previous MQP team. The feature extraction process is simple and direct. Eyben et al. (2017) created a tutorial and user guide that helps install the tool and run commands that automatically generate thousands of signal based features from audio files.

3.3 GPS Features

We extracted GPS features from Google GPS data. We have two types of GPS features, features based on raw GPS data and features based on activity data. From these features, we produced three GPS feature sets, includes raw GPS data features, activity data features, and combined features.

3.3.1 Features Based on Raw GPS Data

Features Based on raw GPS data were calculated from the time, latitude, and longitude values. The first feature we extracted, location variance, is the variability in a user's location. Both Saeb et al. and Farhan et al. found that location variance has a high importance or correlated with PHQ-9 scores. It is calculated as the logarithm of the sum of variances in latitude and longitude values (Farhan et al., 2016).

$$Loc_{var} = \log(\sigma^2_{\text{long}} + \sigma^2_{\text{lat}})$$

We also applied clustering techniques to all coordinates points. The feature number of clusters is calculated from the DBSCAN algorithm (Farhan et al., 2016).

One other feature is entropy. Entropy measures the variability of time that a participant spends at different locations. The entropy is calculated as the equation below, where pi represents the percentage of time that a user spends in location cluster i (Farhan et al., 2016).

$$\text{Entropy} = -\sum(p_i \log p_i)$$

A problem of entropy is that the entropy increases as the number of location clusters increases. Farhan et al. solved this problem by adopting normalized entropy, which is invariant to the number of clusters and depends solely on the distribution of the visited location clusters. The normalized entropy is calculated as the equation below, where Nloc represents the number of clusters calculated formerly (Farhan et al., 2016).

$$\text{Entropy}_N = \text{Entropy}/\log N_{\text{loc}}$$

Since we have two weeks of GPS data, so we calculated each feature for each day's data and the data over the two weeks. In this feature set, we have 4 kinds of features and a total of 60 features.

3.3.2 Features Based on Activity Data

The previous MQP team extracted activity data features. After carefully reviewed their code, we fixed errors in their feature extraction methods and added more features. Each user has 14 KML files that contain their geographic and activity data, one KML file for each day. For one user's one day's data and one user's two weeks' data, we calculated 7 features. So we have 105 features in this feature set in total.

These seven features include the number of placemarks, max distance, total distance, transition time, the number of activities, activities distance and the number of non-activities.

The number of placemarks is the number of different placemarks in a user's KML files, which indicates the number of places a user has been and activities a user has done. Max distance is the distance of the longest trip of a user. Total distance is the total distance a user traveled. Transition time is the sum of time when the data showed a user was walking, cycling, running, driving, flying, motorcycling, moving, on a bus, on a train, on a tram, or on the subway. The number of activities is the number of times a user did the following activities including walking, running, and cycling. Activities distance is the total distance traveled by a user during those three activities. The number of non-activities is the number of times a user did stationary activities such as staying at a place for a long time.

3.3.3 Data Preprocessing

Two weeks' worth of data had to be collected immediately after the user first installs the application. Therefore, raw GPS data was not collected from sensors. In order to retrieve location data that existed before our applications installation, data should have had been getting stored by other applications. Fortunately, Google Maps constantly stores location data in the background inform of a KML format. The KML (short for Keyhole Markup Language) format is a type of XML notation that was developed at Google for expressing geographic annotation and visualization on Google based mapping applications (eg. Google Maps and Google Earth).

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
       at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Figure 4: Example of KML Data

The KML data shown above contains a “Placemark” tag that represents a point drawn over a map. Placemark tags found in the data collected for this study usually represented the activity that user had been performing at that time. Placemark tags provide valuable attributes such as “name” (shown above), “Point”, and “Track” that give us a good indication of the person’s physical activity and location in between time ranges. These attributes reported very specific details about the user’s current state. For example, it reports it detects if the user is using one of the train, bus or tram and stores it in the structure. In addition, we learnt that the multiple Placemark tags found in a single KML file represented different activities a user completed within a single day for the data collected for our study. The data collection study implemented by the previous MQP group provided us with nearly 14 KML file that represented the users location data for the last two weeks.

The KML (XML) structure does not provide a good structure to process data. The data had to be flattened out in order to write scripts that can generate features from it. Therefore, we wrote a script that converted the KML structure to a CSV structure (the minidom module from xml.dom package was used to parse the KML structure). The following shows the conversion result:

Id	Day	Placemark	Name	Address	Category	Distance	Description	Track(altit	Track(coor	Point(coor	LineString(	LineString(	LineString(	LineString(	Begin	End
12	0	0	2469 Frenc	Alpena, MI	49707	0	from 201	clampToGr	-83.45775499999999	45.088784	0				2018-01-1	2018-01-12T13:46:58.519Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.45775499999999	45.088784	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.45775499999999	45.088784	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4597556	45.0896739	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4599845	45.08785	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4613158	45.087092	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4697463	45.0853754	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4716107	45.0837426	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.473146	45.0785698	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4728722	45.0786038	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4728716	45.0785728	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4732987	45.0785502	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4712547	45.051202	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4625784	45.0473392	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4616056	45.0470594	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.4566469	45.0446096	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.45678277645878	45.04466736289075	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	1	Driving		Driving	6513	Driving fro	clampToGr	-83.45678277645878	45.04466736289075	0				2018-01-1	2018-01-12T14:00:12.518Z
12	0	2	Alpena Tov	Michigan		0	from 201	clampToGr	-83.4135747	45.128464699999995	0				2018-01-1	2018-01-12T14:08:00.630Z
12	0	3	Driving		Driving	6793	Driving fro	clampToGr	-83.45678277645878	45.04466736289075	0				2018-01-1	2018-01-12T14:16:10.348Z
12	0	3	Driving		Driving	6793	Driving fro	clampToGr	-83.45678277645878	45.04466736289075	0				2018-01-1	2018-01-12T14:16:10.348Z
12	0	3	Driving		Driving	6793	Driving fro	clampToGr	-83.4707906	45.0474555	0				2018-01-1	2018-01-12T14:16:10.348Z
12	0	3	Driving		Driving	6793	Driving fro	clampToGr	-83.4709811	45.051304	0				2018-01-1	2018-01-12T14:16:10.348Z

Figure 5: Example of CSV Data

The CSV file shown above created a much more helpful structure that tools like *pandas* could use during data manipulation or feature extraction.

3.3.4 Feature Extraction

3.3.4.1 Activity Features

Activity based features can simply get detected by reading strings from rows. For example, in the example shown above, we can easily tell that the person was driving for a specific amount of time till he/she stopped at a certain location (at placemark 2) and then continued driving for given time again. We can generated several features from only this piece of data. For example, we can extract the amount of time the person was driving and count it as part of the “transition_time” feature. We can get the distance driven from the person and add it to the “total_distance” feature. We wrote a python script that goes through the CSV file and automates this process of detecting activity based features from reading strings in rows and columns.

3.3.4.2 Raw GPS Data Features

Raw GPS features were much more complicated to extract. The GPS coordinate data collected from the Google Maps KML file lacked a lot of details and was not compatible for feature generation. However, some extra work was done to artificially fill in missing information in order to test the prediction performance we could attain using the data we already possess. For

example, each “Track” tag in a Placemark represents the movement of the user across a specific path. These Track tags contain an arbitrary number of GPS coordinate readings. None of these readings possess timestamps. However, the existence of timestamps is very important to generate the features selected. Fortunately, the Track Placemark comes with a start and end time. We therefore decided to take the range between the start/end times and divide it with the number of coordinates collected in the Track tag. This gives us the average time range between each coordinate. We can use this average time range value to create artificial timestamps for each coordinate by adding the value to each consecutive coordinates’ new timestamp.

After modifying the data using processes similar to the one above, we created the raw GPS features using a python script. *numpy, pandas* and *sklearn* were used to make the complex calculations required for the formulas listed for each feature in the previous section.

4. Machine Learning Methodology

4.1 K-Nearest Neighbors

K-Nearest Neighbors (KNN) Algorithm, is one of the supervised machine learning algorithm mostly used for classification. It classifies a data point based on how its neighbors are classified.

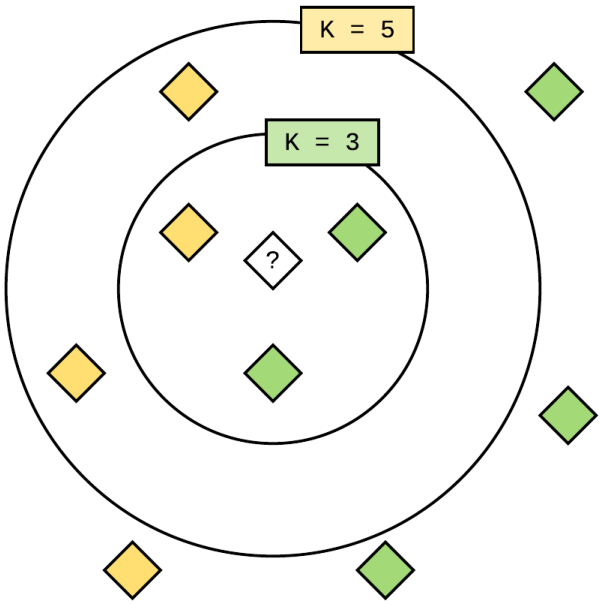


Figure 6: KNN Example

K is a user-defined constant, and an unclassified element is classified by being compared with the k nearest classified elements. KNN makes predictions based on how similar training observations are to the new, incoming observations. The figure below illustrates how the KNN algorithm works.

The white rhombus with a question mark at the center is the unclassified new element. The yellow and green rhombuses around the white one are the classified elements. If we use KNN algorithm to classify the white rhombus and take 3 as the value of K, we need to find the 3

nearest elements, which are two green and one yellow rhombus in the inner circle. Then, the white rhombus will be classified as a green rhombus since there are more green elements next to it. In another case, if we take 5 as the value of K, we need to find the 5 nearest elements, which are two green and three yellow rhombuses in the outer circle. In this case, the white rhombus will be classified as a yellow one since there are more yellow elements next to it.

According to the example above, we can see that the value of K is a significant factor in the KNN algorithm. There are several ways to determine the value of K, such as bootstrap, K-fold cross validation (as known as the KFCV algorithm) and so on. In most cases, the value of k is an odd number to avoid the case that there are same number of element 1 as well as element 2 next to the new element. If the value of k is too large, it will cause underfitting, which refers to a model can neither model the training data nor generalize to new data, in machine learning. If the value of k is too small, it will cause overfitting, which refers to a model that models the training data too well. For example, if the value of k is 1, the machine will just simply classify the new element as its nearest element. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize (Brownlee, 2016).

Both underfitting and overfitting deteriorate the performance of the machine learning. Ideally, we can avoid these problems by using techniques like holding back a validation dataset or using a resampling technique to estimate model accuracy. However, all these techniques cost a lot time and are hard to achieve in our project.

Although KNN is simple and solves problems quickly, it has its own limitations. On classifying depression patients and normal subjects, KNN does not have good performance compare with Logistic regression. (Behshad, Mohammad, & Reza, 2012). According to a study on feature selection methods for depression detection, KNN has a relatively low accuracy and larger standard difference among other machine learning algorithms that have been tested in this study (Cai, Chen, Han, Zhang, & Hu, 2018).

4.2 Support Vector Machine

Support Vector Machine or SVM, is one of the supervised machine algorithms that, like decision tree, is implemented in regression and classification analysis. Although it was first introduced as a suggested way to create nonlinear classifiers by applying the Kernel trick to maximum-margin hyperplanes, it is applicable in numeric prediction, and is vastly applied in many data analysis models (Ghose, 2017).

SVM generally is complex in that the model is hard to understand. Nonetheless, having high accuracy in numeric prediction, SVM is utilized into a black box method. Thus, to analyze the accuracy of an SVM prediction, one would use a training set to build the model and use a testing set to test the outcome of that particular model.

4.2.1 The Basic Concepts of SVMs

As shown in Figure 7, an SVM performs classification by finding the best fit to separate distinct data sets. In this case, we can see a clear line that lies across the plane, on each side has different categorical data.

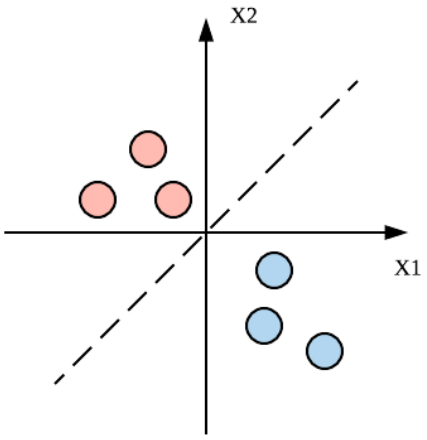


Figure 7: SVM Linear

As shown in Figure 8, there are times when classification does not work with simple linear or curve lines. However, if the data is mapped into a three-dimensional space (as shown in Figure 9), it is easy to find a hyperplane that separates distinct data. As such, SVM projects datasets into higher dimension so as to increase the accuracy on prediction.

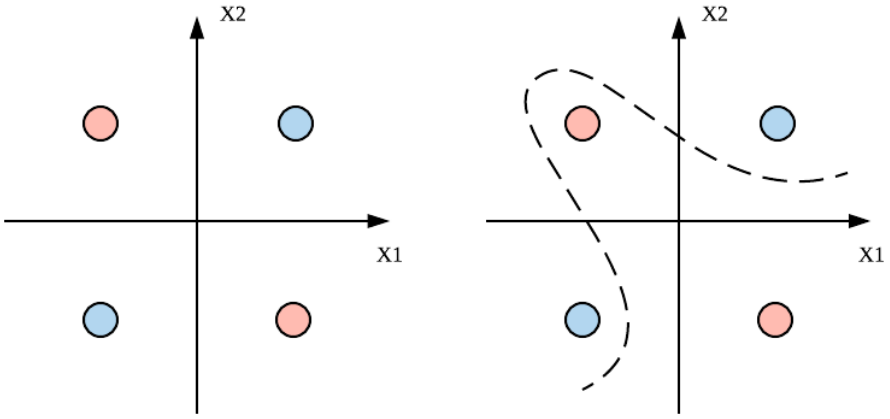


Figure 8: SVM Non-Linear

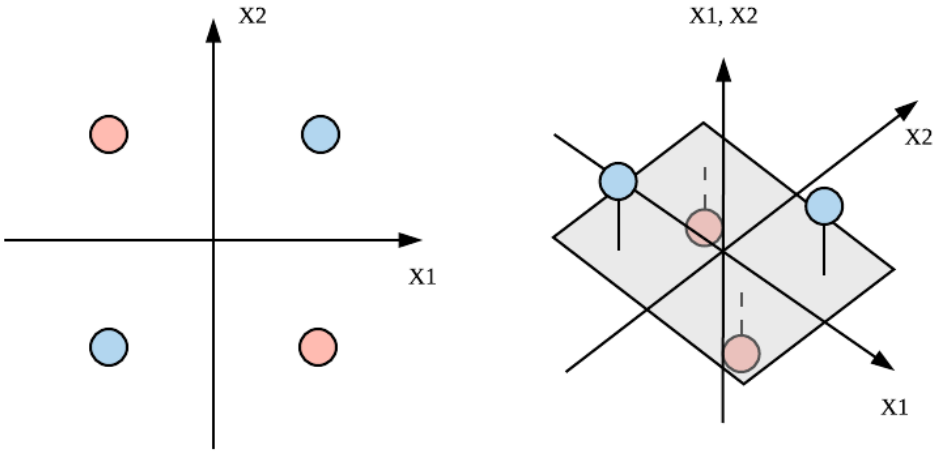


Figure 9: SVM Space

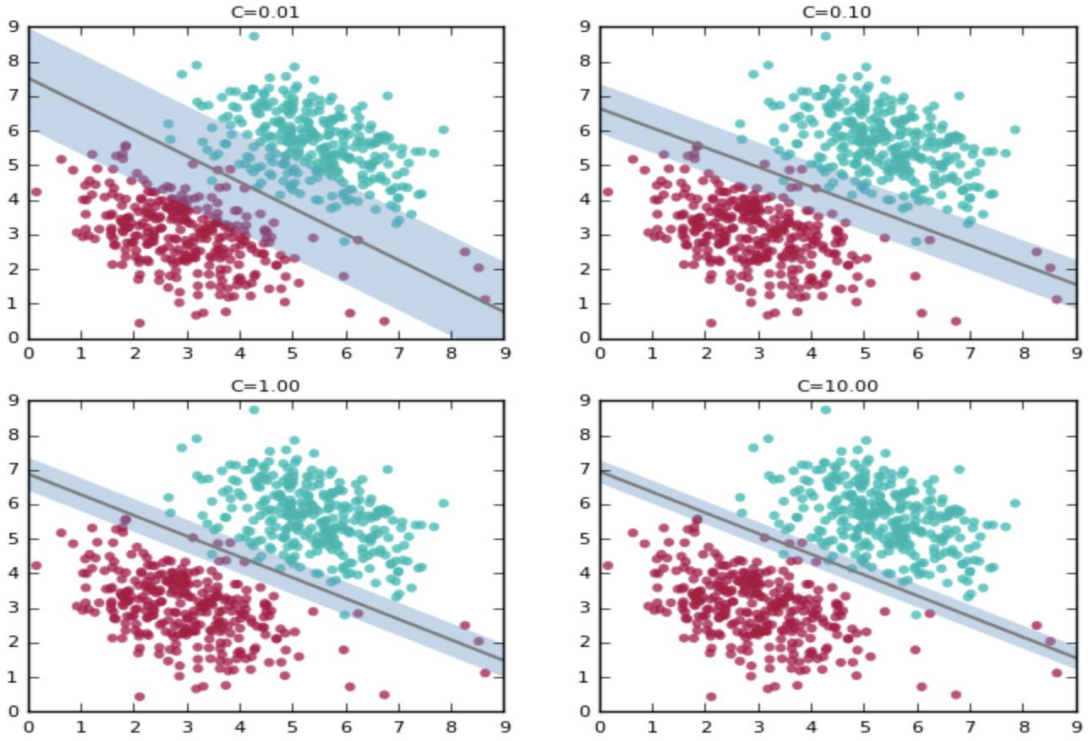


Figure 10: SVM Parameters (Ghose, 2017)

Nonetheless, real life data cannot always be as clean and simple as in the figures, so it is not always possible to completely classify data. Therefore, a parameter, *C*, is needed to indicate the margin of error that the model is willing to take. As shown Figure 10, the increasing value of *C* leads to the shrinkage of margin between the line and furthest distinct data; Because at high values, it tries to accommodate the labels of most of the red points present at the bottom right of the plots.

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma {\mathbf{x}_i}^T \mathbf{x}_j + r)^d, \gamma > 0$.
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$.
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma {\mathbf{x}_i}^T \mathbf{x}_j + r)$.

Figure 11: Basic Kernel Models (Hsu, Chang, & Lin, 2016)

Note that in those basic models shown in figure 11, γ, r, and d are all kernel parameters. Kernel functions maps datasets into higher dimensional space. Among these, RBF is the most useful for basic training model as it does not limited to simple linear model, but also do not have complex parameters as Polynomial kernel function does.

4.2.2 Applications of SVMs

To put in perspective the application of support vector machines, consider object recognition and facial recognition. In object recognition, we can extract the features from an image to develop a "visual vocabulary" and representation of images by frequencies of "visual words," that we can refer to when trying to identify the features of said image. An example of this can be seen in Figure 12.

Figure 12: Object Recognition Example (Stricker, n.d.)

Notice how several features of the image (sky, building, mountain, etc.) were recognized and appropriately labeled.

In facial recognition, we capture the difference between two faces. Upon presenting the algorithm with an image of a face, it reports its best estimate of the unknown face based on its currently known repertoire of faces. Another scenario is presenting the algorithm with a face and an identity, and having algorithm either accept or reject the claim that the given image matches the given identity (and optionally returning a confidence measure of the validity of the claim).

Depression detection from features is also an application of support vector machines. Similar to the above two scenarios, we can extract features from audio and text data that allow us to predict how depressed someone is.

4.3 Random Forest

Random forest is a supervised learning algorithm that can also be used for regression and classification. Random Forest is popular because it's the algorithms that is the most reliable, flexible, and easy to use. Random forest is also an ensemble algorithm. Ensemble algorithms have the ability to combine with a number of other algorithms that either have the same or different type of classifying object. For example, Random Forest uses a number of different decision tree classifiers that fit subsets of the input data and averages those trees in order to improve accuracy.

4.3.1 Basic Concept of Random Forest Algorithm

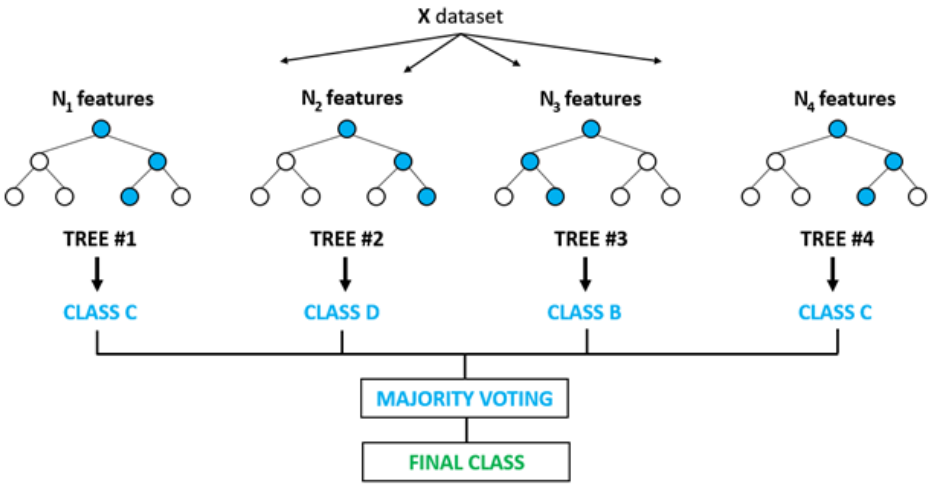


Figure 13: Random Forest Classification (Holczer, 2018)

Random Forest algorithm attempts fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [*sklearn*]. The general process of the algorithm is easy to understand in 4 steps:

1. Split the input dataset into random sub samples
2. Each subsample gets its own decision tree classifier that will give a prediction to that specific sub sample only.
3. Vote on each result predicted from decision trees
4. Finally select the result with the highest vote and present it as output

4.3.2 Advantage of Random Forest - Measuring Feature Importance

Random forest has the ability to measure the relative importance of each feature used in the prediction. For example, scikit-learn (a machine learning tool for python) uses Random Forest to measure feature importance by observing how much tree nodes that use a specific feature reduce impurity across the trees inside the forest (Donges, 2018). It calculated the score of each feature while training data and then scales the score/importance of features in the form of percentages (summing to 100%). This feature of Random Forest will be important to analyze the new features we are using for text and voice analysis.

4.4 XGBoost

XGBoost is an efficient implementation of gradient boosting algorithm. Gradient boosting algorithms produce strong prediction models by summing up the results of weak prediction models (trees) (Chen & Guestrin, 2016). For example, as shown in Figure 14, the final prediction for the boy is 2.9, which is the sum of 2 from tree1 and 0.9 from tree2.

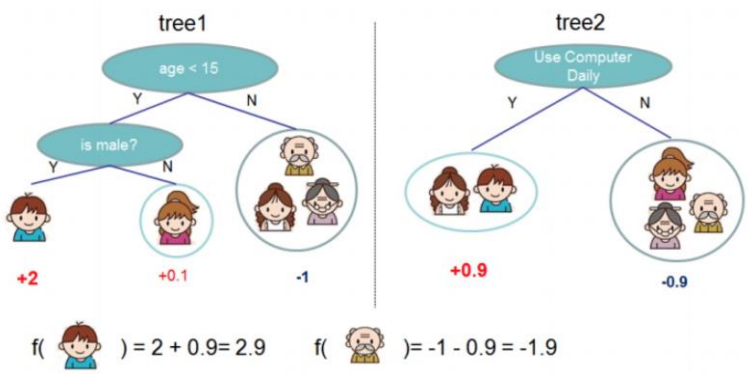


Figure 14: Tree Ensemble Model (Chen & Guestrin, 2016)

In some depression detection research, people chose to use Performance on Different Algorithms and Cutoffs for Combined Features (Saeb et al., 2017).

Just like any tree based modeling packages, there are three options for measuring feature importance in XGBoost: One is weight. In other words, the number of times a feature is used to split the data across all trees. Another one is Cover. A cover is the number of times a feature is used to split the data across all trees weighted by the number of training data points that go through those splits. Last but not least, gain, which is the average training loss reduction gained when using a feature for splitting. These procedures can be summarized as bagging. After that, is where the actual boosting takes place.

Boosting means each subtree “learn” from the mistake of its parent. As shown by the formula: F1 is the child with a fewer bias and variance. So accordingly, the next child F2 has even less bias and variance than its parent F1. This iteration goes on until it reaches the stopping point of the tree.

Algorithm 3: Newton tree boosting

Input : Data set $\mathcal{D}$.
A loss function L.
The number of iterations M.
The learning rate η.
The number of terminal nodes T

Initialize $\hat{f}^{(0)}(x) = \hat{f}_0(x) = \hat{\theta}_0 = \arg\min_{\theta} \sum_{i=1}^{n} L(y_i, \theta)$;

for $m = 1,2,..,M$ **do**

$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

Determine the structure $\{\hat{R}_{jm}\}_{j=1}^{T}$ by selecting splits which maximize $Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{G_{jm}^2}{H_{jm}}\right]$;

Determine the leaf weights $\{\hat{w}_{jm}\}_{j=1}^{T}$ for the learnt structure by $\hat{w}_{jm} = -\frac{G_{jm}}{H_{jm}}$;

$\hat{f}_m(x) = \eta \sum_{j=1}^{T} \hat{w}_{jm} \mathrm{I}(x \in \hat{R}_{jm})$;

$\hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \hat{f}_m(x)$;

end

Output: $\hat{f}(x) \equiv \hat{f}^{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$

Figure 15: Pseudo code for actual boosting (Synced, 2017)

4.5 AdaBoost

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ("weak learners") is combined into a weighted sum that represents the final output of the boosted classifier.

AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. It is sensitive to noisy data and outliers. In

some problems it can be less susceptible to overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge as a strong learner.

The final equation for AdaBoost classification can be represented as follows:

$$F(x) = sign(\sum_{m=1}^{M} \theta_m f_m(x)),$$

Figure 16: Final Equation for AdaBoost Classification (SauceCat, 2017)

4.6 Decision Tree

Decision tree is a machine learning algorithm to solve both classification and regression problems. It often mimics human-level thinking, so it is easy to understand the logic of the data. Take a look at the following visual of this algorithm:

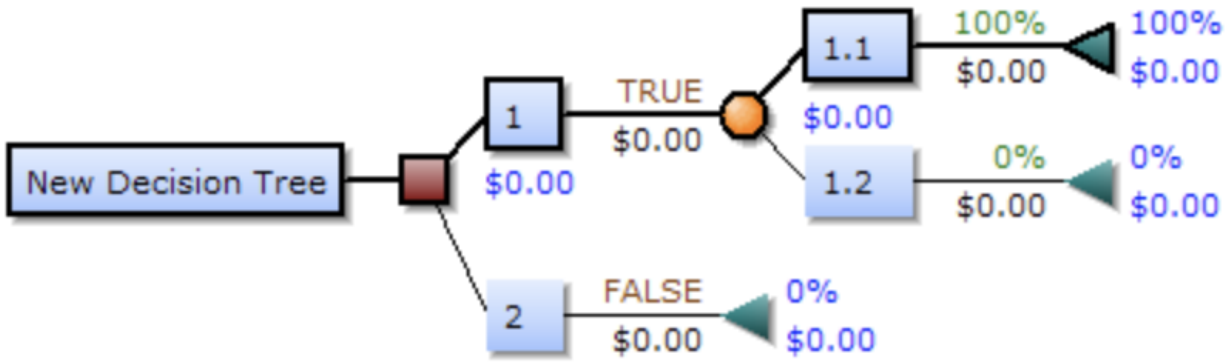


Figure 17: Example of Decision Tree (Gupta, 2017)

Squares represent decision nodes, circles represent chance nodes, and triangles represent end nodes. There are two main types of decisions: classification trees and regression trees. Classification tree analysis is when the predicted outcome is the (discrete) class to which the data belongs. Regression tree analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. Here is the equation to calculate Gini Impurity.

$$\mathrm{I}_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1-p_i) = \sum_{i=1}^{J} (p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2$$

Figure 18: Equation of Gini Impurity (Gupta, 2017)

All in all, the workflow of decision trees are not that complicated. First, they start from the root and observes the value of the attribute at the root. Then, they follow the path that corresponds to the observed value. It then repeats these steps until it reaches a leaf node, which will give us the final decision.

4.7 Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). For example, predicting whether a message is sent by a female or not is a scenario in which logistic regression could be used. Like all regression analyses, logistic regression is a predictive; it is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

There are three kinds of logistic regression: binary logistic regression, multinomial logistic regression, and ordinal logistic regression. Binary logistic regression's categorical response has only two possible outcomes (for example, spam or not spam); Multinomial logistic regression's response has three or more possible outcomes without ordering. (such as predicting which food is preferred more: vegetarian, non-vegetarian, vegan); ordinal logistic regression's response has three or more possible outcomes with ordering (say, a movie rating from one to five). The final equation to calculate the probability is shown below:

$$p = \frac{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}+1} = \frac{1}{1+b^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}$$

Figure 19: Final Equation of Logistic Regression (Binary) (Swaminathan, 2018)

4.8 Artificial Neural Networks

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules (Josh, 2015).

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information (a "signal") from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, they are aggregated into layers, with different layers performing different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times (Josh, 2015).

4.9 Voting

Voting, while similar to artificial neural networks, is not an algorithm. Instead, it is a framework for many different machine learning algorithms to work together. The voting framework is straightforward: it gets results from several machine learning algorithms and lets them vote for the final decision. There are two kinds of voting methods: hard voting and soft voting. The following example shows you how these methods work:

If we have three classifiers (1, 2, 3) and two classes (A, B). Classifier 1 predicts class A with probability 99%; classifier 2 predicts class A with probability 49%; classifier 3 predicts class A with probability 49%. Hard voting chooses class B, since 2/3 (more than half) of classifiers predict class B. Soft voting chooses class A, since (99 + 49 + 49) / 3 = 65.67% (more than class B).

4.10 Gaussian Process

Gaussian processes are stochastic processes (a collection of random variables indexed by time or space), such that every finite collection of these random variables has a multivariate normal distribution (i.e., every finite linear combination of them is normally distributed). The distribution of a Gaussian process is the joint distribution of all (infinitely many) random variables, and as such, is a distribution over functions with a continuous domain (e.g., time or space).

A machine learning algorithm that involves a Gaussian process uses lazy learning and a measure of the similarity between points (the kernel function) to predict the value for an unseen point from training data. The prediction is not just an estimate for that point, but also has uncertainty information; it is a one-dimensional Gaussian distribution (which is the marginal distribution at that point).

Here are the equations for Gaussian processes.

$$\mathbf{X}_{t_1,\ldots,t_k} = (X_{t_1}, \ldots, X_{t_k})$$

$$\mathrm{E}\left(\exp\left(i \sum_{\ell=1}^{k} s_\ell \, \mathbf{X}_{t_\ell}\right)\right) = \exp\left(-\frac{1}{2} \sum_{\ell,j} \sigma_{\ell j} s_\ell s_j + i \sum_{\ell} \mu_\ell s_\ell\right)$$

Figure 20: Equations for Gaussian Processes (Bailey, 2016)

4.11 Grid Search

Grid search is a traditional way to perform hyperparameter optimization. A hyperparameter is a parameter whose value is used to control the learning process. In machine learning, hyperparameter optimization, or tuning, is the idea of choosing a set of optimal hyperparameters for a learning algorithm. By contrast, the values of other parameters (typically node weights) are learned.

Grid search is a method to find the best combination of hyperparameters (an example of a hyperparameter is the learning rate of the optimizer), for a given model (e.g., a CNN) and test dataset. In this scenario, you have several models, each with a different combination of hyperparameters. Each of these combinations of parameters, which correspond to a single model, can be said to lie on a point of a "grid". The goal is then to train each of these models and evaluate them (by using, say, cross-validation). You then select the one that performed best.

5. EMU Pipeline

5.1 Feature Extraction Pipeline

In order to collect raw files for each part of the feature extraction, the first step of this pipeline is having Python run SQL queries to extract each type of data. Each kind of data is stored with an ID and the contents associated with that ID.

For extracting text features, we grouped their text contents by ID. Since the text data contains both sent and received types, this grouping process creates three sets of files, which are:

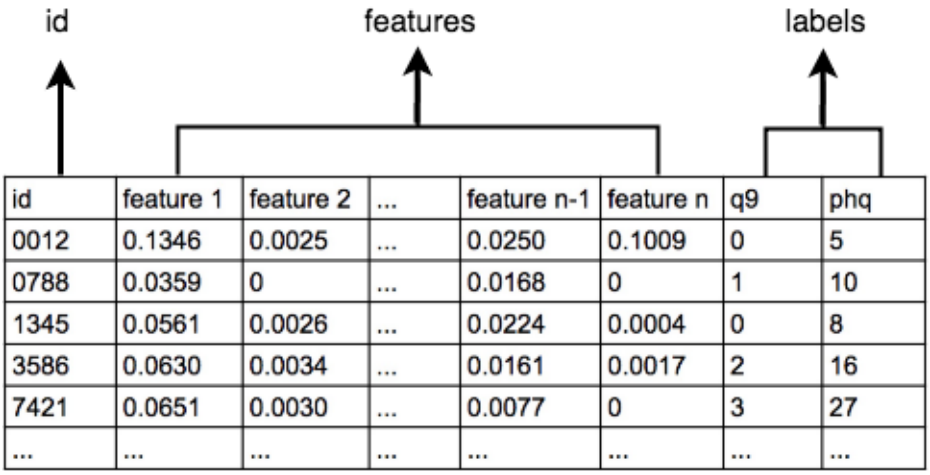
- Each ID having all texts grouped together
- Each ID having only sent texts grouped together
- Each ID having only received texts grouped together

Additionally, We did similar procedures for tweets data:

- Each ID having the tweets body grouped together

In total, we have four grouped files read to apply feature engineering to.

For text feature engineering, we call upon up to three libraries, TextBlob, Empath, and Linguistic Inquiry and Word Count (LIWC), by adding arguments to loop through each person's grouped texts as dictionary chunks. This process might take up to five minutes (depending on system specs) since there are loops for every person. Accordingly, four JSON feature files that contain all of the IDs and associated features are generated. In the next step, we implemented a Python function that simply transposes the JSON file into a clean CSV file with the IDs of each person, and the columns being the features (Figure 21). The last step of our text feature engineering pipeline is to use *pandas* to merge the score of the ninth question in the PHQ-9 and PHQ-9 total score into the four files, according to their IDs.



id	feature 1	feature 2	...	feature n-1	feature n	q9	phq
0012	0.1346	0.0025	...	0.0250	0.1009	0	5
0788	0.0359	0	...	0.0168	0	1	10
1345	0.0561	0.0026	...	0.0224	0.0004	0	8
3586	0.0630	0.0034	...	0.0161	0.0017	2	16
7421	0.0651	0.0030	...	0.0077	0	3	27
...	...	...	...	...	...	...	...

Figure 21: Feature CSV File Structure

For extracting audio features, we first decode Base64 encoded strings and transform them into 3GP files, and then convert the 3GP files to WAV files. Finally, we use Praat and OpenSMILE to extract features from those WAV files, and save the IDs, features, and PHQ-9 scores into a CSV file, which is ready to be fed into the machine learning pipeline.

For extracting GPS features, we firstly convert the strings that we receive from the database into XML files. Following that we catch the information we want from the XML files and store all of the data into one CSV file. The data includes the duration of activities, the name of the activities, the coordinates, etc. Finally, we extract GPS features from the CSV file and store the features in a final feature CSV file.

5.2 Machine Learning Pipeline

5.2.1 Metrics

This section describes the classical metrics that we used in the evaluation of our models. Although we generated all of the metrics for each experiment and stored them in our project repository, in later chapters we mainly use the F1 score to evaluate the performance of models.

1. TP stands for true positive, which is the number of depressed people who are correctly classified as depressed by the model.

2. TN stands for true negative, which is the number of non-depressed people who are correctly classified as non-depressed by the model.
3. FP stands for false positive, which is the number of non-depressed people who are wrongly classified as depressed by the model.
4. FN stands for false negative, which is the number of depressed people who are wrongly classified as non-depressed by the model.
5. Precision explains the proportion of positive identifications that was actually correct, and is calculated as:

$$Precision = \frac{TP}{TP+FP}$$

6. Sensitivity explains the proportion of actual positives that was identified correctly, and is calculated as:

$$Sensitivity = \frac{TP}{TP+FN}$$

7. F1 score is a combination of precision and sensitivity, and is calculated as:

$$F1\ Score = 2 * \frac{Precision*Sensitivity}{Precision+Sensitivity}$$

8. Accuracy explains the proportion of the total number of predictions that was correct predictions, and is calculated as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

We primarily used F1 scores as our reference since a higher F1 score ensures a higher precision as well as a higher sensitivity. In principle, we would rather have more people that are misdiagnosed as depressed than ignoring the people who actually are depressed.

5.2.2 Resampling and feature reduction

In order to address the unbalanced data, the pipeline provides two options to address the issue: downsampling and upsampling. The default is downsampling to randomly remove the majority class, since unlike upsampling, it does not produce a new database on resampling original data, and accordingly is more stable for the whole pipeline.

After downsampling, the machine learning pipeline provides the options of feature reduction, The default feature reduction metrics use chi-square statistics to find the k best features:

$$x_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

A chi-square statistic compares the target variable (PHQ-9 score) with features. The calculation of chi-square statistics between every feature variable and the target variable gives us an observation of the existence of a relationship between the variables and the target. We rank them by how closely (how dependent they are) and then choose the top k features. In this case, k is a user input.

We also implemented principle component analysis (PCA), which reduces the dimensionality of a feature set but still retains and rescales the variation between them. In addition to that, we do also provide the options of linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA), which make predictions by estimating the probability of a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made. The primary difference is that QDA estimates different classes with a different variance. Lastly, we provided the option of doing recursive feature reduction (RFE). This algorithm recursively trains and eliminates the least important features until it reaches the number of desired features.

5.2.3 Training and testing

We implemented a series of baseline machine learning algorithms to train the data. These include logistic regression, random forest classifiers, support vector machines, neural networks, k-nearest neighbors, XGboost, and Gaussian processes. The training process is done using 5-fold cross-validation. The pipeline gives a series of options as the arguments of the pipeline including the following:

1. Cutoff: An integer from 0 to 20. The PHQ-9 score ranges from 0 to 27, so we can select the score we want to set as the marker saying that someone is depressed or not for machine learning. For example, if the cutoff is set to 10, then someone with a score above 10 is classified as depressed, and someone with a score below or equal to 10 is classified as not depressed.
2. Filename: The input path of the feature file (if Filename2 exist, this will be the training set, otherwise, this cross-validation will be applied on this file).

3. (Optional) Filename2: The input path of the feature file (testing set).
4. Feature type: The pipeline takes in 3 arguments, “text”, “gps” and “audio”, which correspond to three specific sets of tuned classifiers.
5. Resample type: This takes in “down” or “up” for the type of resampling. As mentioned, the default is downsampling.
6. Model type: This includes all of the baseline classifiers and ensembled algorithms AdaBoost and voting.
7. Feature selection: The pipeline provides all the feature reduction techniques mentioned above.
8. Target data: In most cases, we input the column index of the PHQ-9 score in the feature file, which is usually -1 or -2.
9. Grid parameters: The parameter scope for hyperparameter tuning. Tuning means that hyperparameter optimization is using exhaustive grid search.

This approach tries all of the combinations of parameters and returns the best F1 score and the corresponding set of parameters. After all is said in done, we use the cross-validation provided by scikit-learn to train and test the feature sets. The training includes a binary classification using the PHQ-9 score as the target within the balanced feature sets.

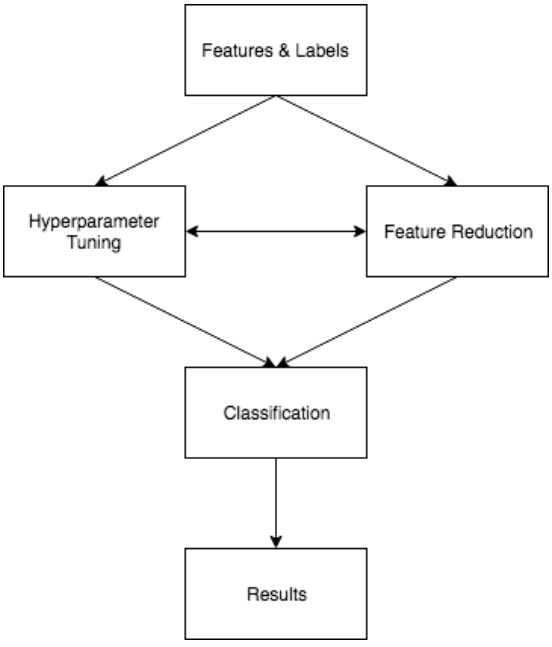


Figure 22: Machine Learning Pipeline Flow

6. Experimental Methodology

6.1 Amazon Mechanical Turk

Amazon Mechanical Turk (MTurk) is a crowdsourcing marketplace that makes it easier for individuals and businesses to outsource their processes and jobs to a distributed workforce who can perform these tasks virtually. This could include anything from conducting simple data validation and research to more subjective tasks like survey participation, content moderation, and more. MTurk enables companies to harness the collective intelligence, skills, and insights from a global workforce to streamline business processes, augment data collection and analysis, and accelerate machine learning development.

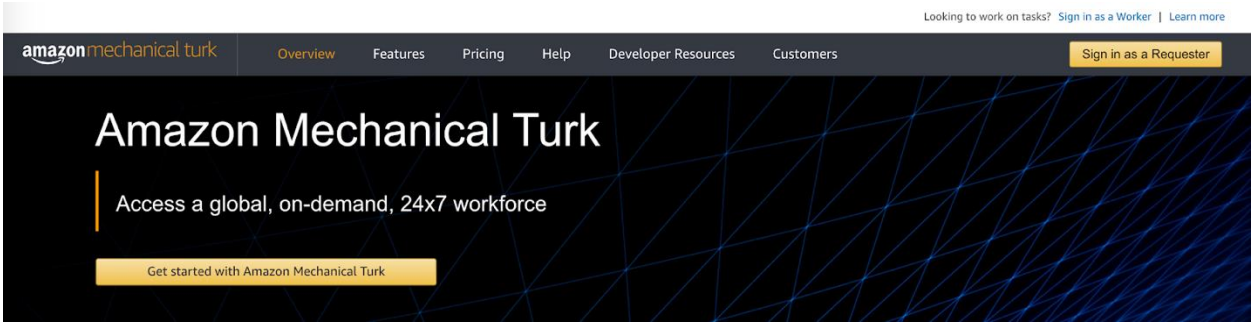


Figure 23: Starting Page of Mturk

We use Mturk to collect more people's phone data to improve our machine learning algorithm and our APP.

6.1.1 Preparation

To setup a project on Mturk. You need to create an Amazon account and sign in as a requester.

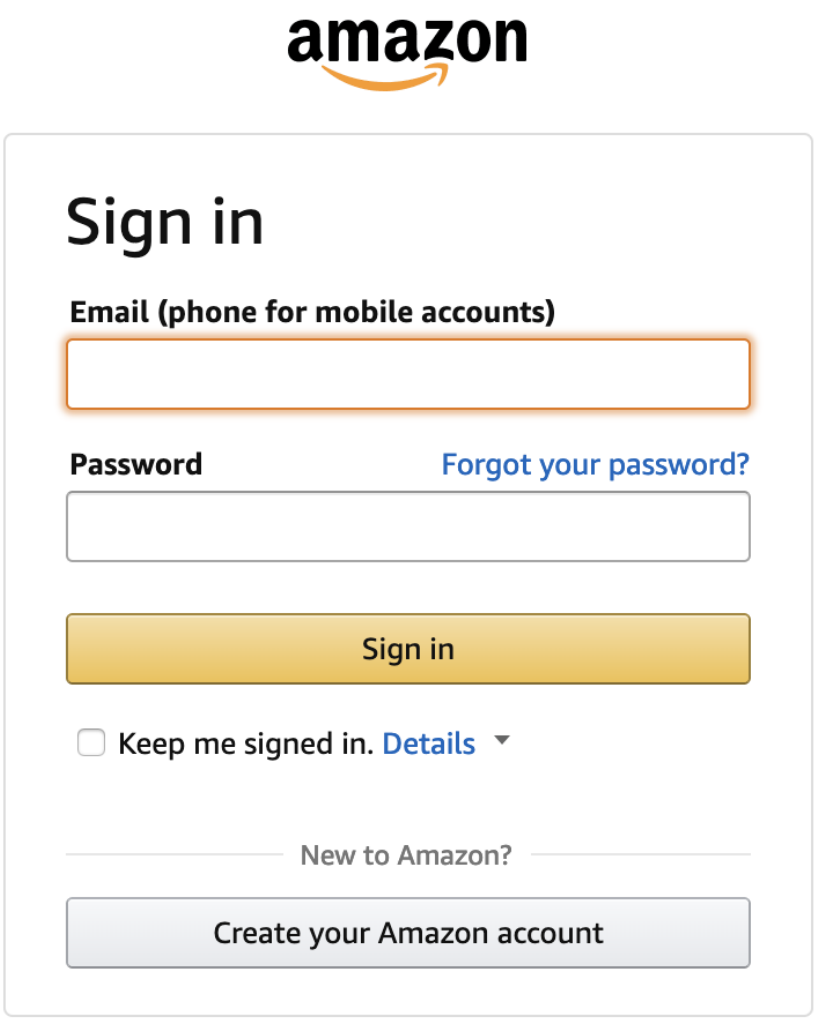


Figure 24: Sign in Page

After that, you need to click “New Project” and select the project type you want. For our project, we choose “Survey Link”.

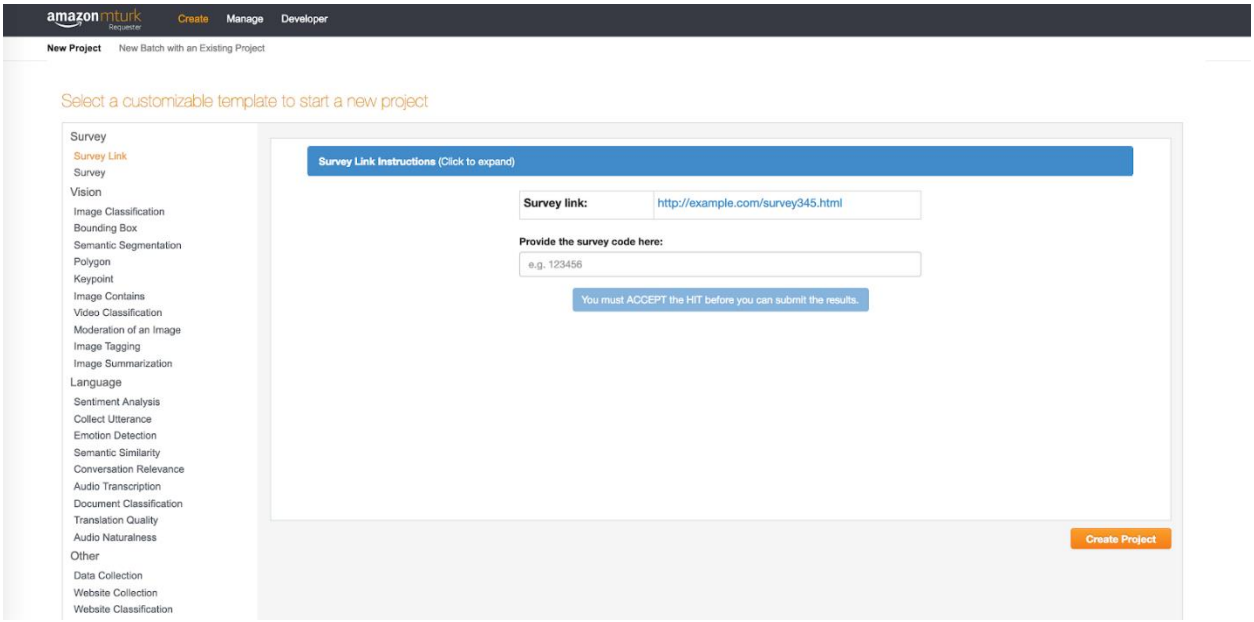
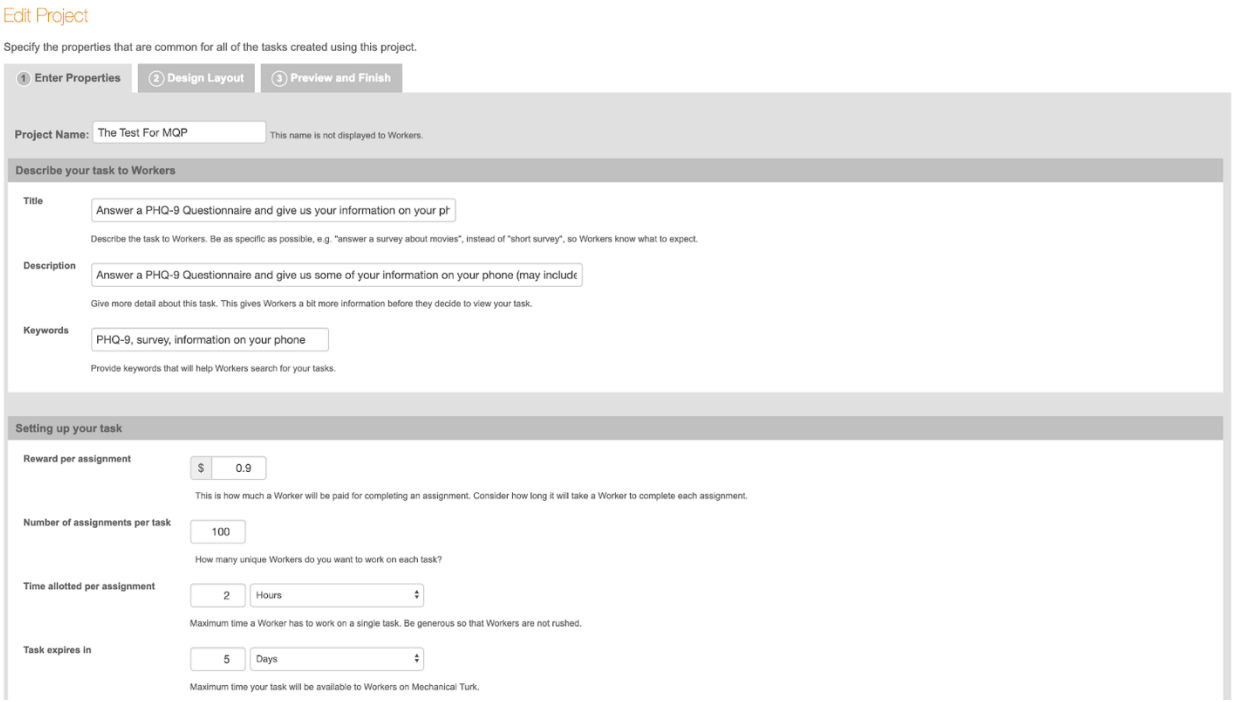


Figure 25: Choosing Project Type

Then, you can edit the project name, title, keywords and so on of your project. You need to decide how many participants you want for your project and how much many they will get. You can also add qualifications for your participants here.



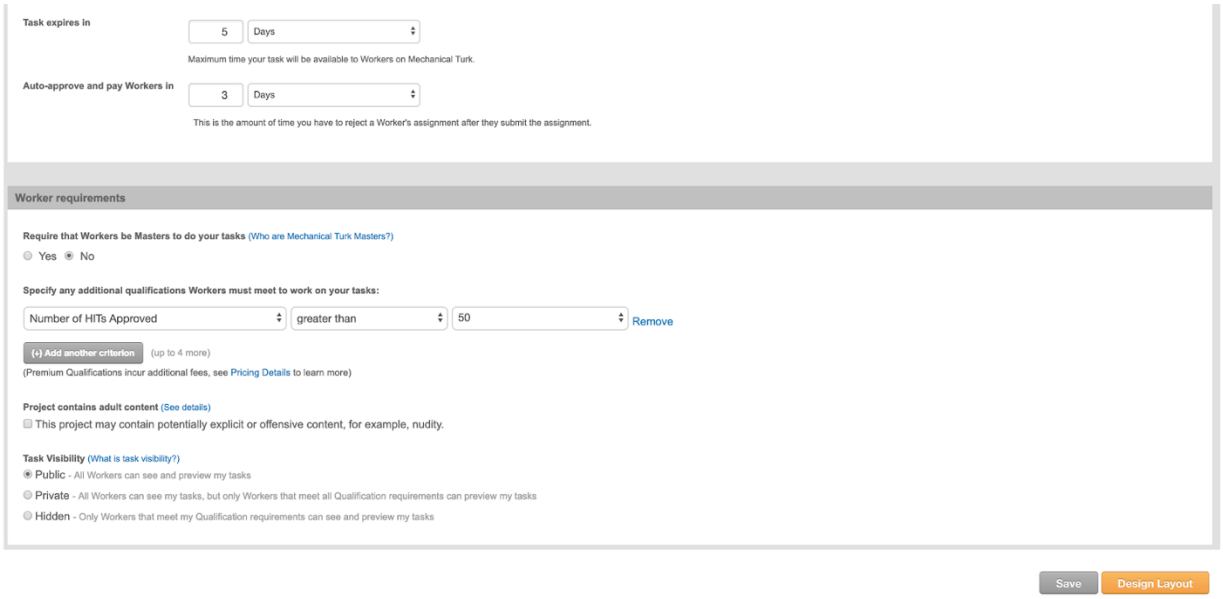


Figure 26: Edit Project Page

After saving, you can start to design the layout of your project. For our project, we put the link to download our APP at “Survey Link”.

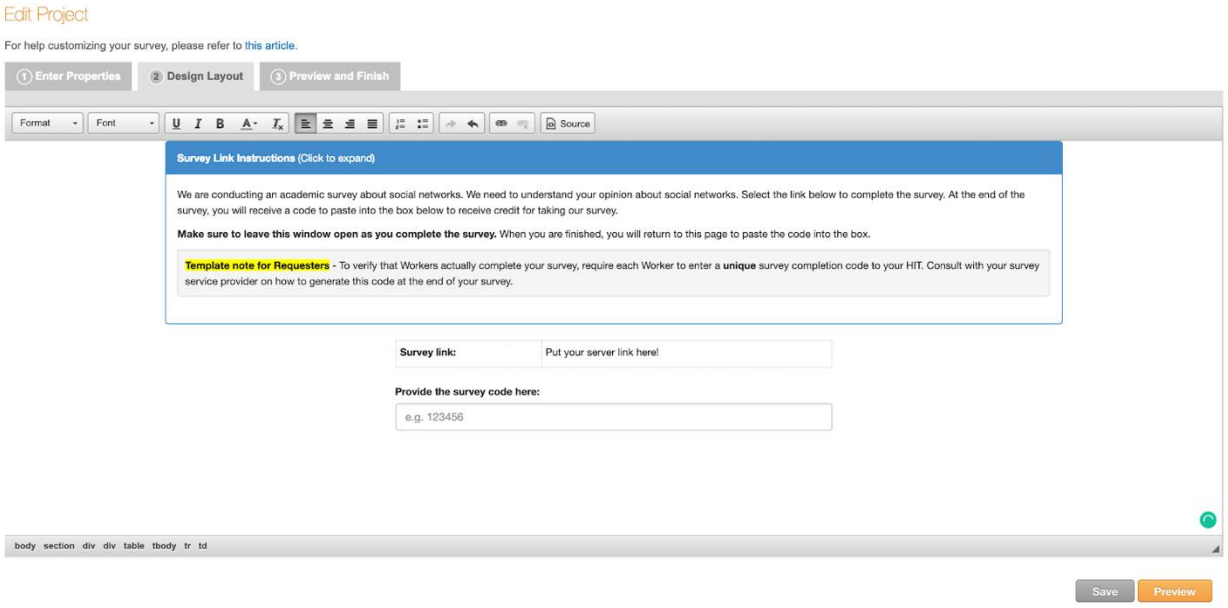


Figure 27: Design Layout Page

After saving, the project is all set.

6.1.2 Publishing a Batch and Viewing the Result

Now, you can publish a batch of your project and choose you payment method. After that, you need to wait for the participants to finish your project.

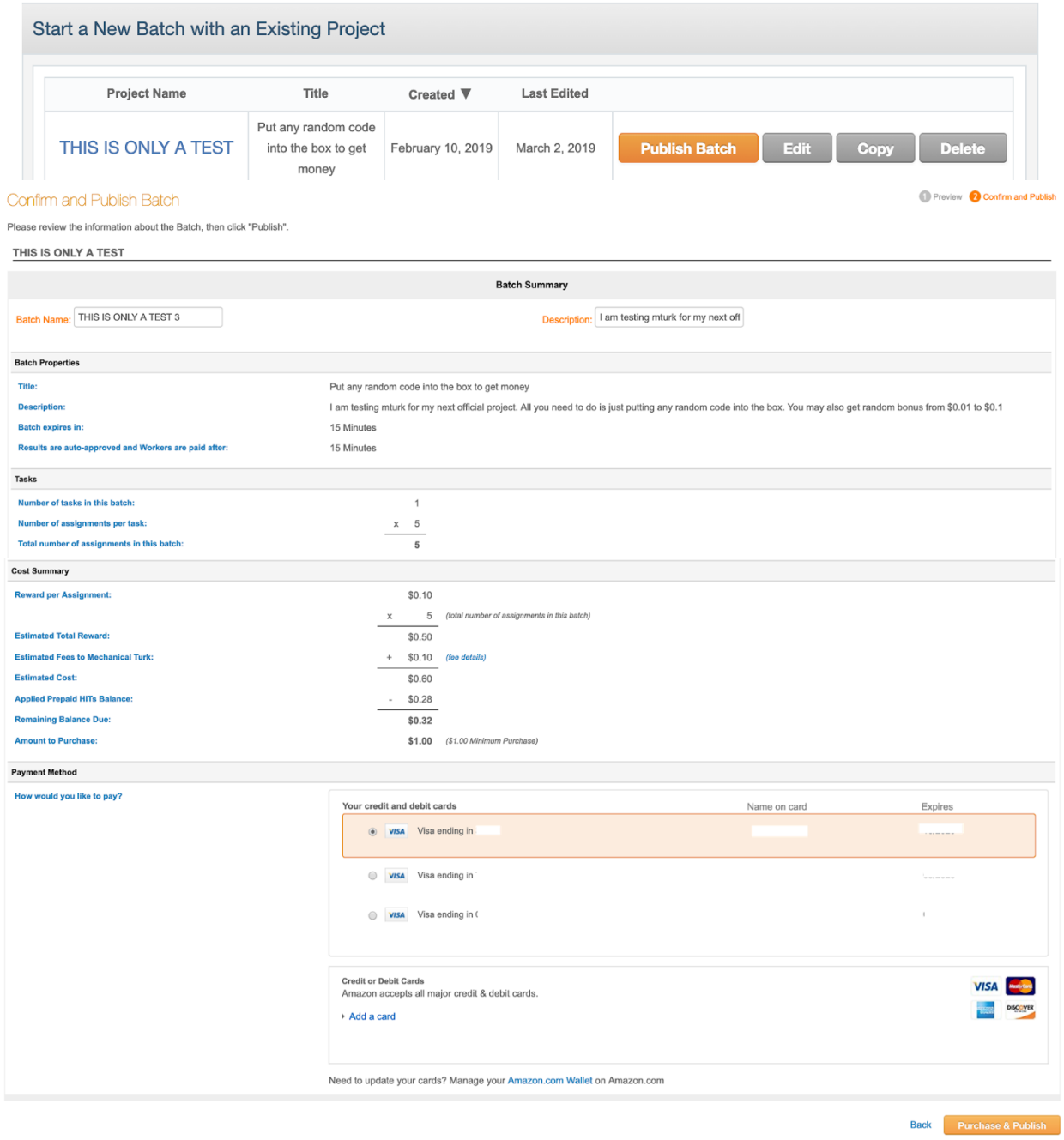


Figure 28: Publish Page

You can manage your project on the manage page. You can accept or reject participants' work on this page. You can also click any work's ID to go to the bonus page and give him/her bonus.

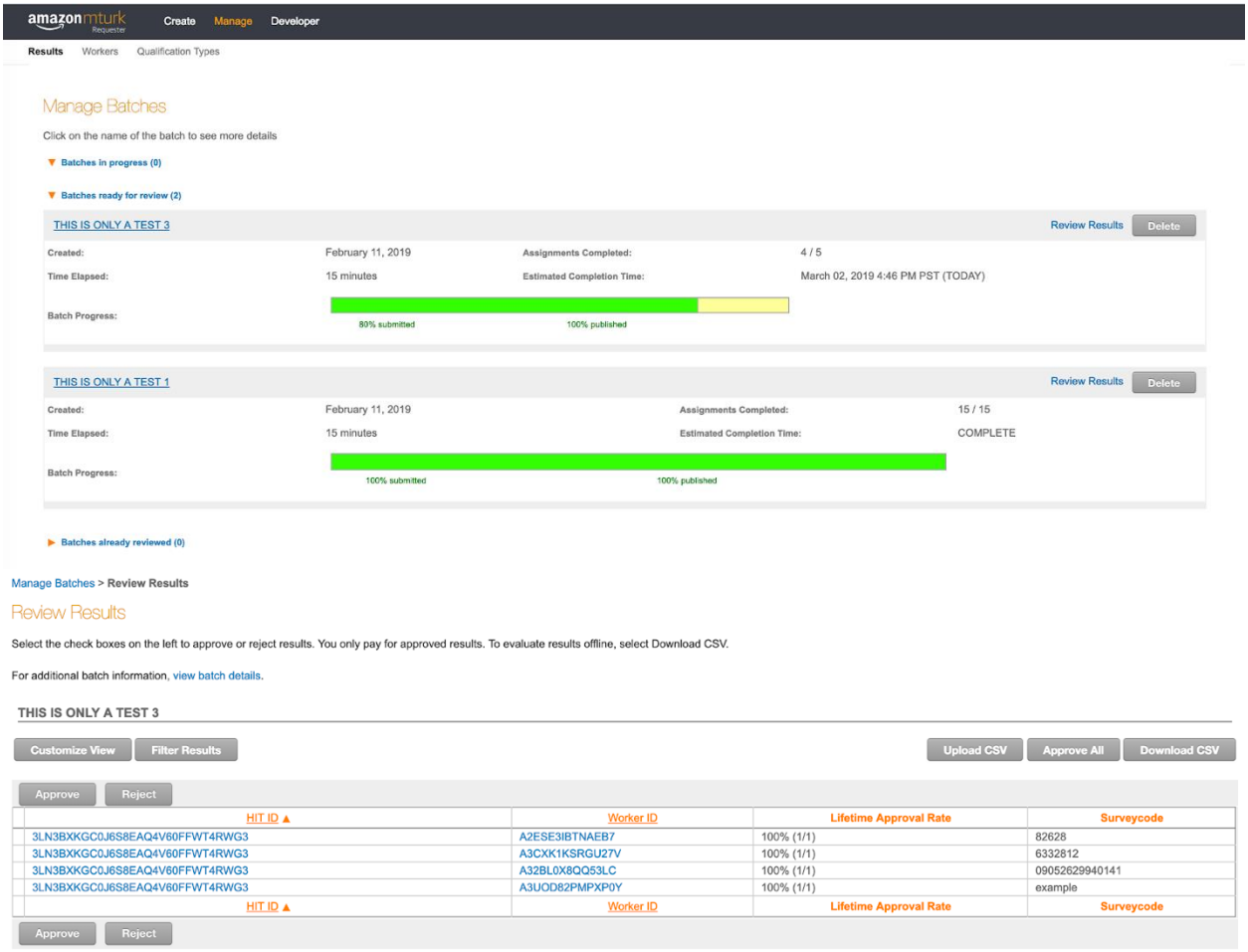


Figure 29: Manage Page

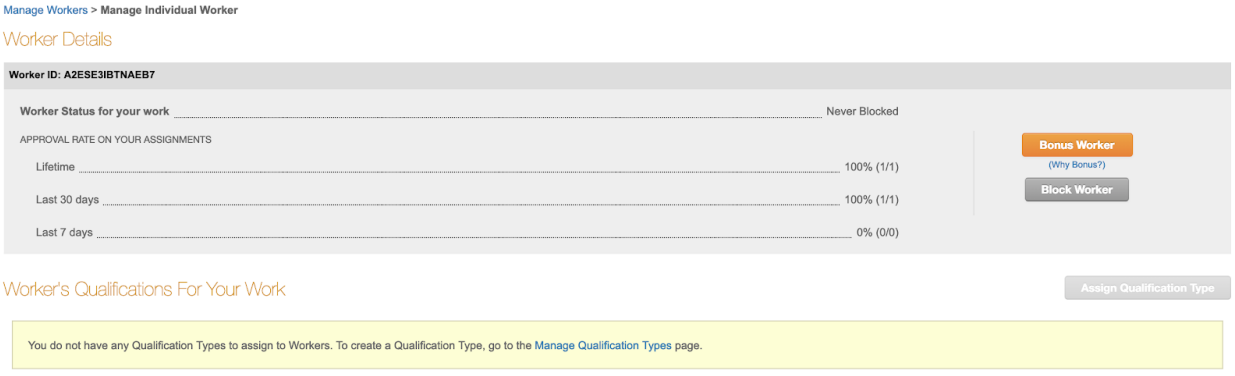


Figure 30: Bonus Page

6.1.3 Publishing Another New Batch

You can also publish a new batch in the publish page to get more participants.

6.2 Mobile Application

An Android application was developed for collecting data. The Android platform was mainly selected because it allows third party applications access to more user data (after requesting permission from the user). It also increases the number of users we can gather data from. Android is more prevalent than other competing platforms, it currently consisting of 74% of mobile devices worldwide.

The application we developed is called EMU Survey. The application takes users through a series of pages that collect data while keeping track of the reward we will pay to the user. The application collects demographics information, phq9, gad7, basic phone data (which includes text messages, call logs, calendar, and contacts), Google Maps location data, voice recordings, Instagram posts and tweets from Twitter. These modalities were selected after running an investigative study that explored which data modalities people would willingly share with us for our study.

The application has a consistent design format. Two bars are displayed on the top and bottom of each screen. The bar on top informs users which section of the data collection process they are in, which also displaying the amount of money they would be paid once finishing the current survey session. The bar at the bottom is a button that submits data and guides users to the next section of the data collection process. It also displays the amount of money users should expect to add if they submit data from the current page. The section in between these bars is scrollable and its content changes for each screen. The following shows the order and content of each screen in the app.

1. **Home Page** –

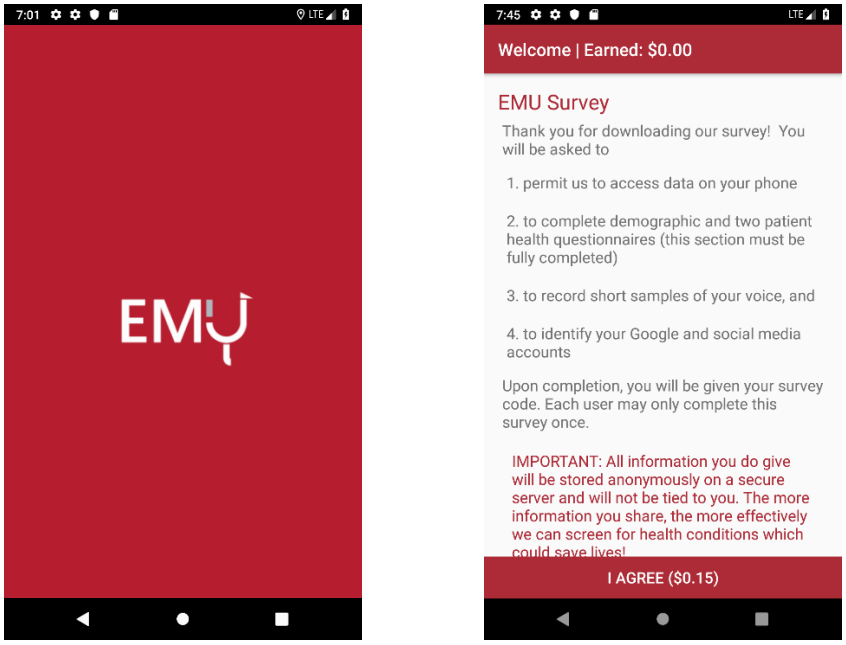


Figure 31: Home Page

This page informs the user about how the app works. It also explains that their data is handles securely and anonymously. Data collection does not start in this page. However, this page does check if there is an internet connection before sending the user to the next page. It displays a message of there is no internet connection and blocks the user from continuing the survey. If the user clicks the button on the bottom of this page and if there is a working internet connection, a connection to our server is established. If the user has already previously completed a survey session, then a message is displayed on the phone informing the user that they can't proceed to create a new session.

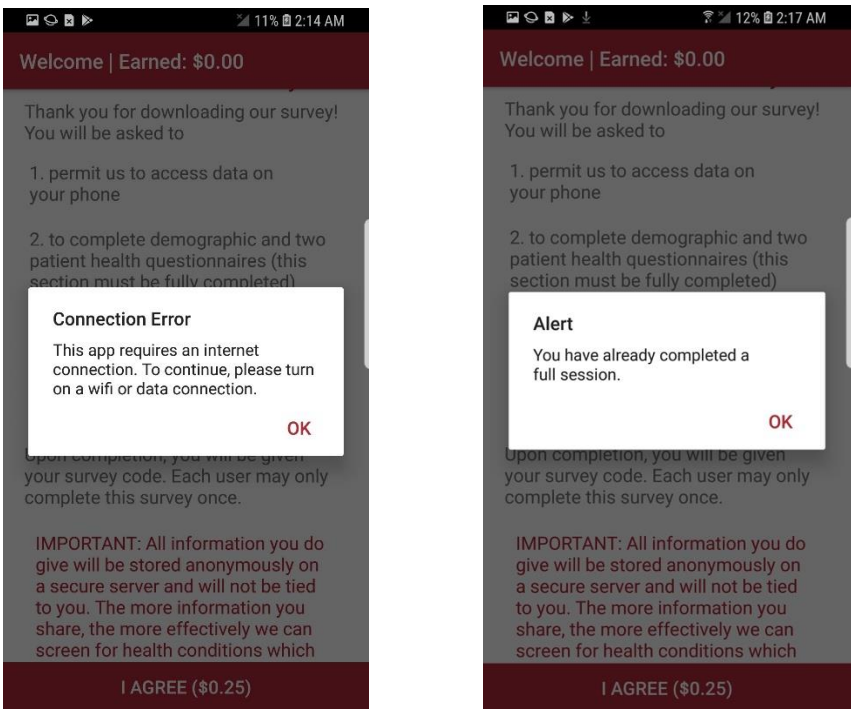


Figure 32: Error and Alert Pages

If the user has not completed a session previously, the server generated a unique session code and a new session is started on the device.

2. **Demographics Page** – The very first thing that happens on this page is not related to demographics data. Instead a series of permission requests pop out in order to collect basic phone data.

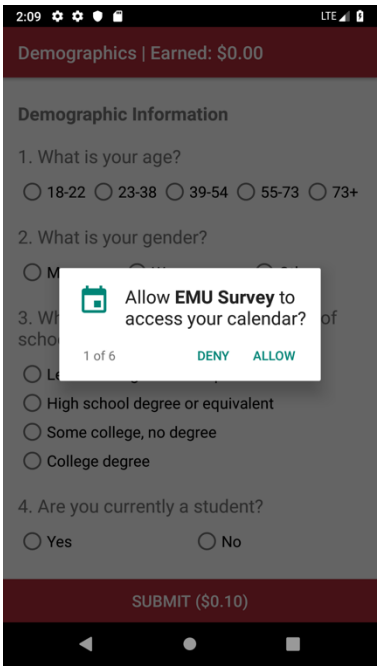


Figure 33: Permission Page

Once these permissions are accepted and the collected data has been sent to the server. The reward amount is updated on top bar of the page and then users can enter demographics data. Demographics data is entered through a series of questions that allow input through radio buttons. Users cannot proceed to the next screen if all questions have not been answered.

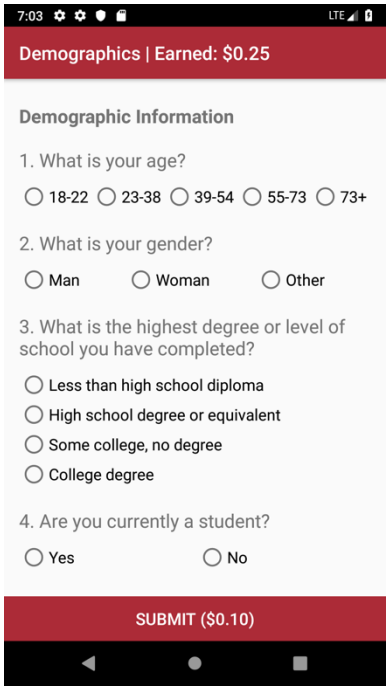


Figure 34: Demographics Page

Once users have answered all questions they can press the submit button.

3. **Fill out the PHQ-9** – This page contains a PHQ9 questionnaire form. It allows user input in the same format as the demographics page, however, each choice is a number from 0 to 3. An immovable section right below the top bar displays the meaning of the numbers next to each radio button.

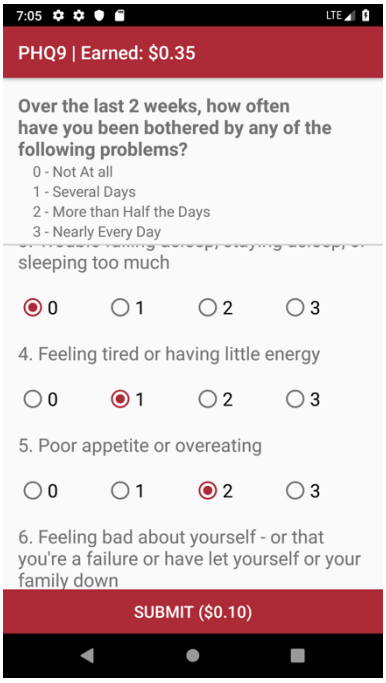


Figure 35: PHQ-9 Page

4. **Fill out the GAD-7** – This page contains a GAD7 questionnaire form. The design of this page is exactly the same as the PHQ9 page.

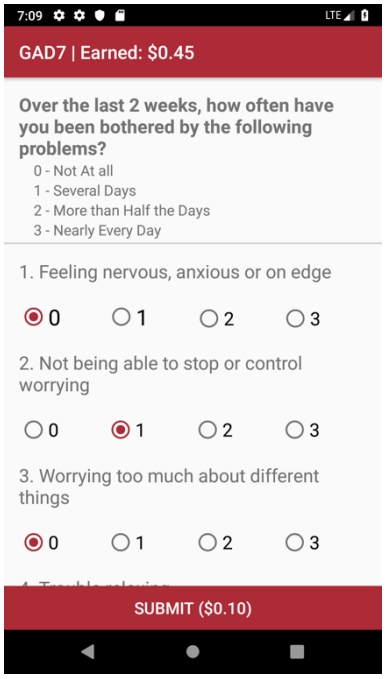


Figure 36: GAD-7 Page

5. **Recording a Voice Sample** – This page requires users to read from a prompt while the phone records their voice through its microphone. It contains a button that starts the recording process. The button also requests audio recording permission if it has not already been granted. Once the user is done reading the prompt, they press the stop button and record again or press submit to go to the next page.

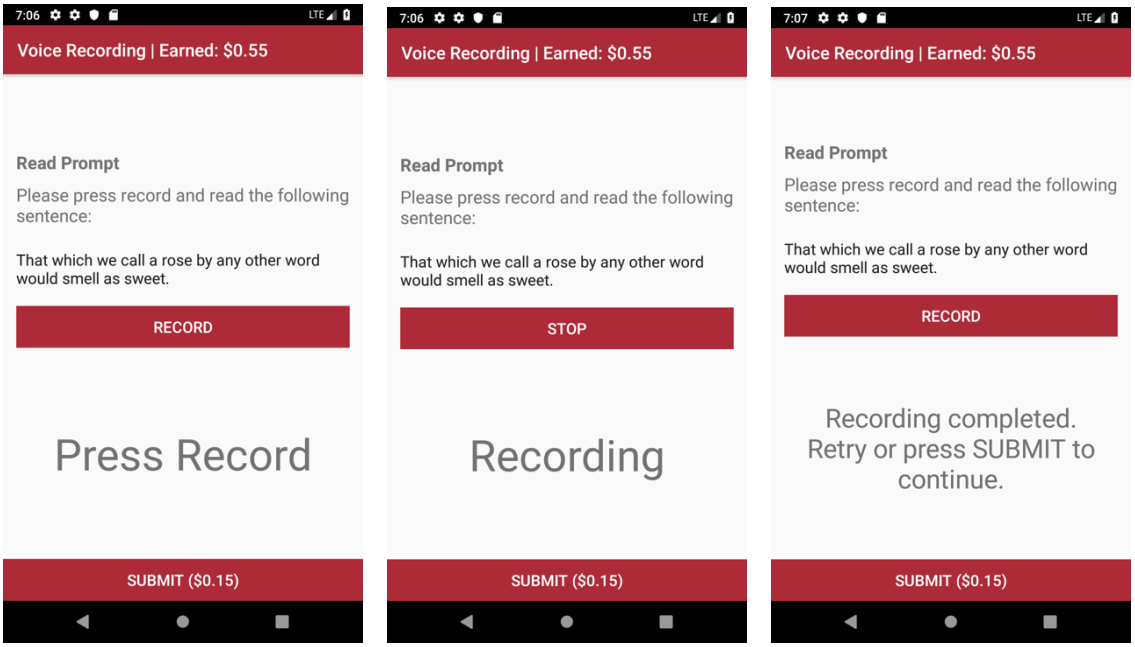


Figure 37: Voice Page

6. **Record an Opened Ended Voice Sample** – We need two different type of audio recordings. The first one requires the user to read a prompt. This one asks users an open ended question in order to get a sample of their natural speaking pattern. This page also sets a timer to how long users can speak (currently 30 seconds). Users cannot stop the recording until the timer reaches at least half of the allotted time (currently 15 seconds).

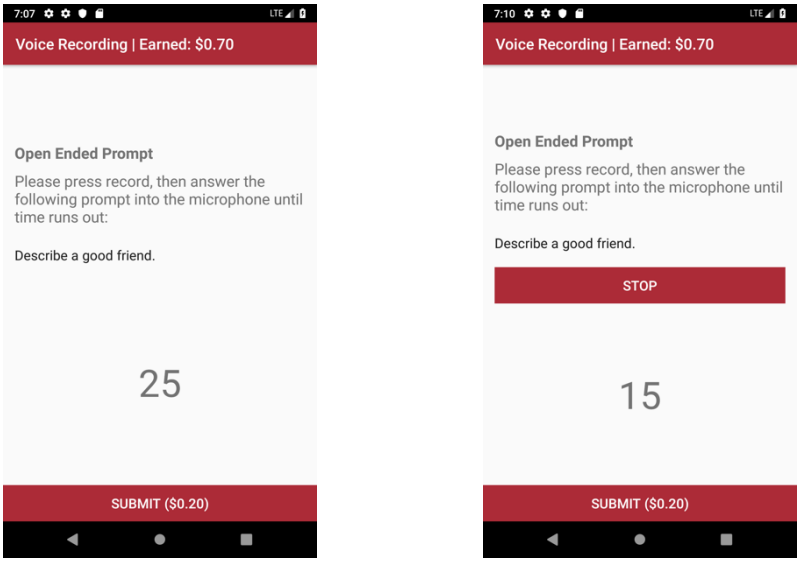


Figure 38: Voice Page 2

7. **Google Maps** –Google continuously collects location information through GoogleMaps in Android phones. This page allows us to access this data. The user is presented with a Google login form. After a user logs in, GoogleMaps KML data is collected for the last 2 weeks and sent to the server.

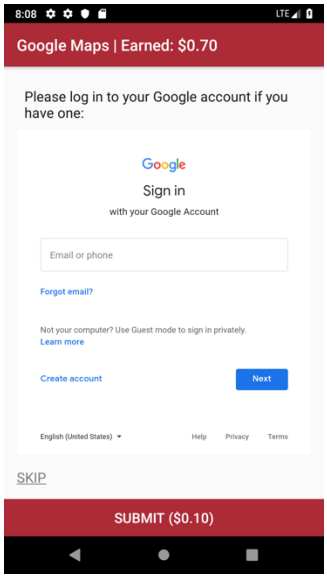


Figure 39: Google Page

The user can skip this or any of the next data collection methods through a skip button shown on the bottom left part of the screen (just above the bottom bar).

8. **Twitter** – This page contains a simple textbox that asks the user for a twitter username. The app checks if the user has put in a valid username and then sends it to the server. The server then accesses tweets from the last two weeks for this user and stores the text data.

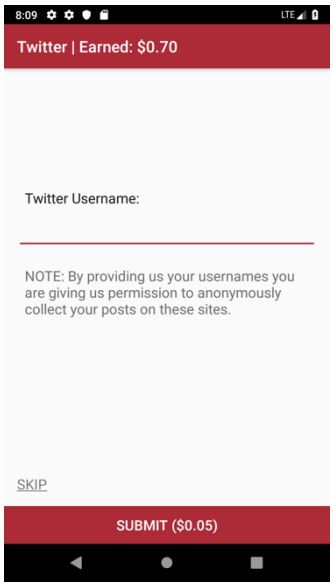


Figure 40: Twitter Page

9. **Instagram** – We are not analyzing image data for this project, but we wanted to collect image data for people that might work on this project in the future. The app presents an Instagram login form to the user. Once the user has logged in, metadata of their posts are accessed and sent to the server. The metadata contains links to the images on Instagram's/Facebook's server.

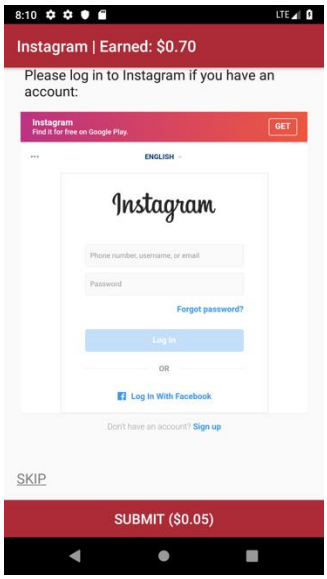


Figure 41: Instagram Page

10. **Thank You and Code Screen** – Once the app is done collecting data for all modalities, the user is presented with this final screen. The screen reveals the session ID the app has been using to communicate with the server on this page. Users are told that this session ID is the code they need to enter on MTurk in order to get their compensation. Once the user is displayed this page, the server will recognize that this phone has completed a full survey session and will block any attempts to start a new session in the future.

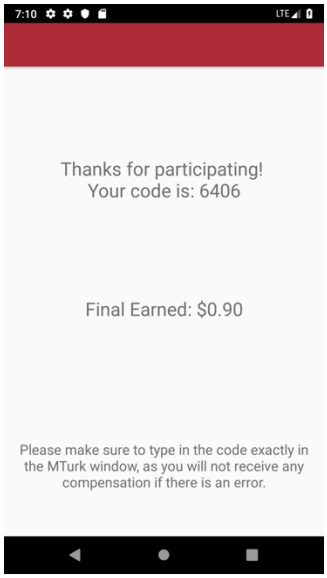


Figure 42: Thank You Page

6.3 EMU Server

The EMU server takes care of the information received by the study participants. All of the participants' responses from the mobile Android application are sent here and properly inserts the data into an SQLite database. We can then read from this database and begin feature extraction and machine learning analysis, without ever looking at the data itself. To avoid a single user participating multiple times, we've restricted duplication by enforcing unique phone IDs.

6.3.1 SQLite Database

An SQLite database is used to house all of the data received. Two tables were used, one for the actual data, and one used to check user compensation. We can then use simple queries to access more specific information (or just view the tables in their entireties). The two tables are broken down below:

Table	*Column*	*Type*
data	id	number
	type	string
	content	JSON
ids	sessionid	number
	phoneid	string
	date	number
	compensation	number
	paid	number

Table 10: Database Structure

6.3.2 Raw Data

The two subsequent subsections break down the specifics of each table.

6.3.2.1 Data Table

1. id - This matches up with the sessionid of the ids table (broken down in section 6.3.2.2). Each time the app is launched, a new session is created. All values of sessionid are unique in the other table, but there may be multiple instances of a single id in this table due to the way the data is stored.
2. type - This represents the type of data that appears in the content column. This falls into one of:
    - current_location (current location)
    - demographic (answers to demographics questions)

- calendar (calendar data)
- log (call logs)
- text (text messages)
- contact (contacts data)
- phq (answers to PHQ-9 questions)
- gad (answers to GAD-7 questions)
- audio (response to closed audio prompt)
- audio_open (response to open audio prompt)
- twitter_username (Twitter username)
- tweets (Twitter data)
- gps (GPS data)
- Instagram (Instagram profile information)
- Instagram media (Instagram data)

3. content - This is where the actual data is. It is a string representation of a JSON object, and is the data we use during feature extraction and machine learning analysis! The contents of this column contain sensitive data and we do not need to look at them to proceed.

6.3.2.2 IDs Table

1. sessionid - This matches up with the id of the data table. Each time the app is launched, a new session is created. All values of sessionid are unique. Each session has different types of data associated with it. For each type of a participant's data, there is a row in the database for it.
2. phoneid - This is an identifier for the device being used. Ideally, there is only one record for each phone so that any given device only participates in the study a single time.
3. date - A string representation of when the session took place.
4. compensation - The amount of money to be paid to each participant.

5. paid - A ternary value representing previous sessions by a user, if any (0), the final and accepted session of a user (1), and if the user was paid (2).

6.3.3 Server Hosting

The server is accessible via SSH with login information only available to EMU personnel (student researchers and advisors). After gaining access, the database is able to be found in the form of a *.db file, which can be opened and viewed by any supported programs and plugins (for instance, DB Browser for SQLite). Participant privacy and security is of *major* importance to us, and thus having WPI host it for us was certainly the way to go.

7. Experimental Evaluation and Analysis

The study involves a dataset was collected by a previous team, which has 335 instances with at least one data modality. While collecting a new dataset, we used the old dataset to test the reliability of our features and the performance of our machine learning methods. Then we used the tuned classifiers to run machine learning experiments on the new data.

Before we started our experiments, we generated a histogram (Figure 43) to show the distribution of the participants' PHQ-9 scores. The horizontal axis measures the PHQ-9 scores and the vertical axis is the number of participants with that score. Although it seems right-skewed, scoring from 10 through 27 means that the depression severity is moderate to severe, and scoring below 10 means the depression severity is none to mild. 10 is also the median of this dataset, so we decided to mark 10 as an import cutoff. Accordingly, most of our experiments are designed on the merit of such an ideology.

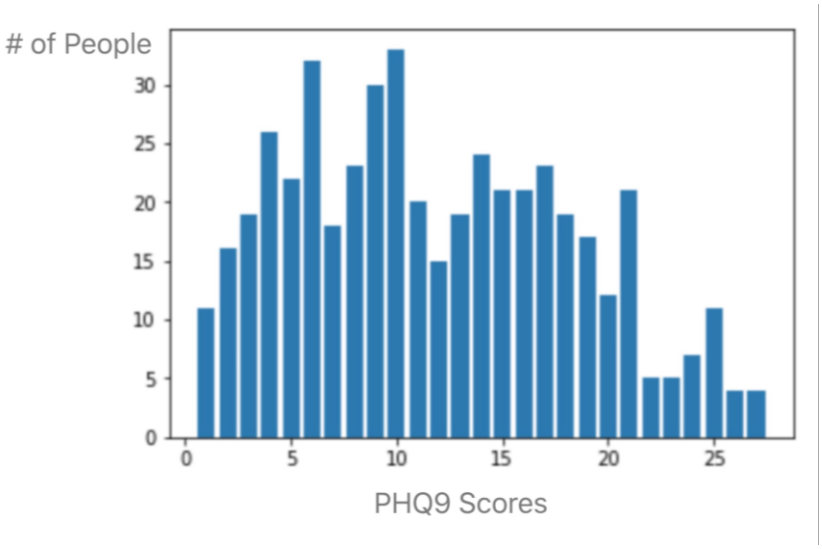


Figure 43: PHQ9 Scores Distribution

7.1 Text Features

We designed a series of experiments following this order: First we ran an experiment to find out how to obtain the best feature subsets, and in the later experiments we will use the selected tool to extract features. Next, we ran experiments to select the best classifier and its

parameters for text features. Then we ran experiments between features selected in different ways. Finally, we concluded the best combination we found in the previous steps and applied it to the new dataset.

7.1.1 Performance Analysis on Different Feature Subsets

7.1.1.1 Spam Removal

This experiment is designed to test out the effectiveness of spam removal in the data processing step. We only used keywords method to remove spam in our project. We used support vector machines (SVC) as our machine learning model and ran the experiment repeatedly at cutoff 10, yielding the data shown in the figure below. We ran it 100 times in order to simulate a true random stimulation. The spam removal did increase the stability for prediction; however, it also decreased the F1 score. Considering spam removal does not effectively increase the overall performance of the prediction score, we decided not to include it in future trials.

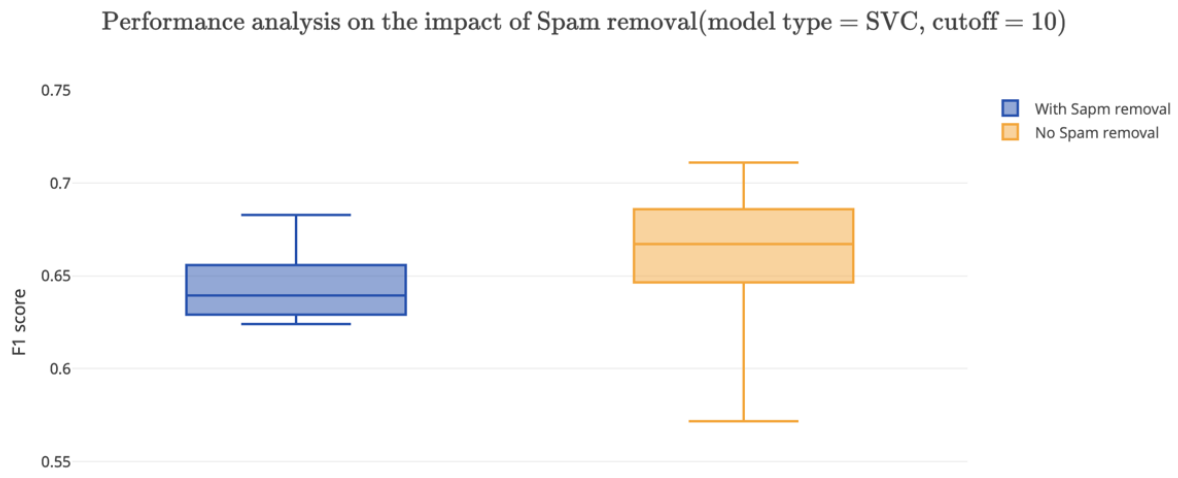


Figure 44: Boxplots of Spam and No Spam Performance

	With Spam removal	No Spam removal
Max	0.682744	0.7109374
Median	0.639403	0.667089
Min	0.6240352	0.571744

Table 11: With Spam and No Spam Performance (F1 Score)

7.1.1.2 Feature Extraction Tools

This is an experiment designed to analyze the impact of features generated by each tool. We decided to use SVC as the model type. We ran it 100 times in order to simulate a true random stimulation at cutoff 10. The result is shown in the figure below. LIWC has an outstanding performance against other tools; this is why we consider it to be worth purchasing. Apart from LIWC, TextBlob is a tool that is not only free, but also reliable.

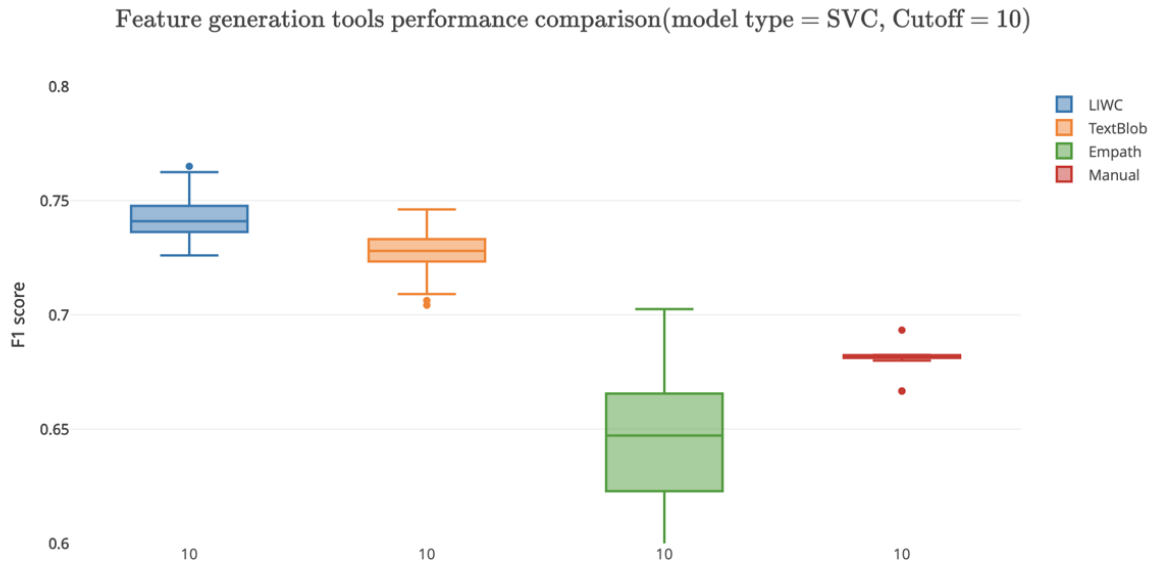


Figure 45: Boxplots of Different Tools Performance

	LIWC	TextBlob	Empath	Manual
Max	0.765017	0.746144	0.7025184	0.693333
Median	0.740984	0.727982	0.647189	0.681176
Min	0.726009	0.704219	0.546751	0.533333

Table 12: Different Tools Performance (F1 Score)

7.1.2 Performance Analysis on Different Classifiers

This part of the experiment is set up aiming to find the most ideal classifiers for machine learning results. In this way, we first need to fine-tune each classifying algorithm and then compare them across each other. We utilized the following seven baseline algorithms:

- Support Vector Machine (SVC)
- k-Nearest Neighbours (kNN)
- Random Forests (RF)
- XGBoost (XGB)
- Logistic Regression (LR)
- Neural Network (NN)
- Gaussian Process (GP)

For hyperparameter optimization, we implemented an exhaustive grid search. The general metric we used to decide the final parameters is to run it 100 times. For each run, we call a built-in grid search function that returns the combination of parameters with the highest F1 score. After that, we selected the combination of parameters of each algorithm that has the highest frequency. The control variables in this experiment are the cutoff (at 10) and no feature selections. To obtain the best combination of each algorithm, we utilized pivot tables (see Appendix C). Accordingly, this mechanism would be applied for text, audio, and GPS features.

Below are the classifiers with their optimized parameters for text features:

- SVC: C:1, kernel:rbf, gamma:10
- kNN: n_neighbors: 9, leaf_size: 1
- RF: max_depth: 3, min_samples_leaf: 1, min_samples_split: 5, n_estimators: 500
- XGB: min_child_weight: 1, gamma: 5, subsample: 0.7, colsample_bytree: 0.6, max_depth: 4
- LR: C: 0.1, solver: ‘lbfgs’, multi_class ‘ovr’, random_state: 0
- GP: max_iter_predict: 1 ,n_jobs: 1
- NN: max_iter:50, learning_ratez:’adaptive’,solver: ‘sgd’,init = 2

After obtaining these tuning sets, we entered into the next step of this experiment, where we ran the machine learning pipeline with the aforementioned classifiers and parameters; 100 times each. The results can be seen in Figure 46 and Table 13. Random forest has shown the best performance across all seven baseline algorithms in terms of both scores and stability. Having a maximum F1 score of 0.75, a mean F1 score of 0.72 and a minimum F1 score of 0.69, RF outperforms SVC (second place) in all three measurements. So overall, fine-tuned RF has the best performance among all seven classifying algorithms within the scope of a cutoff of 10, and not doing any feature selection. Nonetheless, feature selection is crucial to improving machine learning results. Ideally, to find the most optimized hyperparameter, it is best to do feature selection and grid search tuning simultaneously. This is not entirely feasible for our pipeline and computational power, so we decided to optimize a general tuned classifier first, followed by feature reduction to further boost the results (the same is true with audio and GPS features).

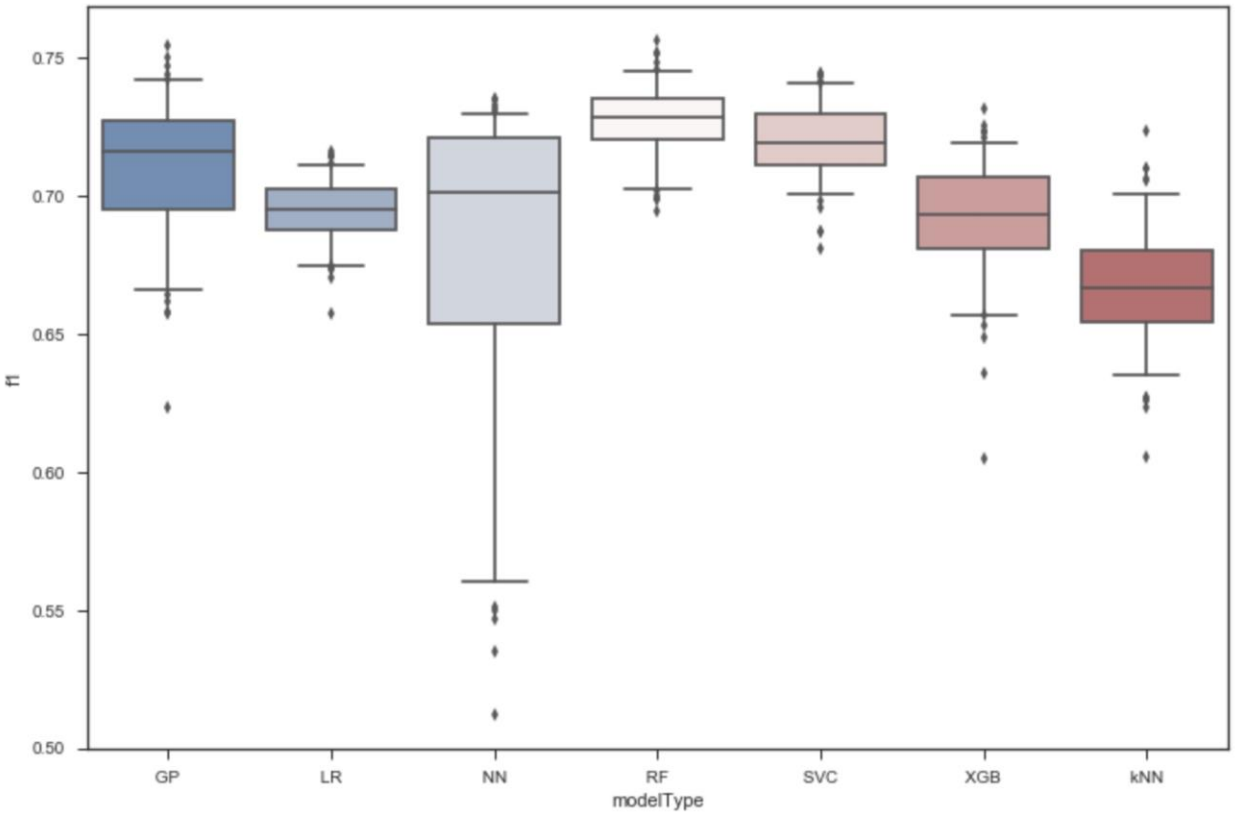


Figure 46: Tuned Classifiers For Text Features Comparison

modelType	Min f1	Mean f1	Max f1
Gaussian Process(GP)	0.623154	0.710575	0.753917
Logistic Regression(LR)	0.657408	0.694446	0.715872
Neural Network(NN)	0.512148	0.677941	0.734928
Random Forests(RF)	0.694245	0.726743	0.756249
Support Vector Machine(SVC)	0.680601	0.719592	0.744080
XGBoost(XGB)	0.604582	0.691365	0.731362
k-Nearest Neighbours(k-N N)	0.605650	0.667866	0.723210

Table 13: Statistic Summary of Tuned Classifiers For Text Features Comparison

7.1.3 Performance Analysis on Different Feature Sets

This part of the experiment aimed to find the most ideal sets of features for machine learning results. To do so, we divided the process into two parts. First, we experiment on various possible combinations of manual features. After finding the feature set with better performance, we then run experiments using different feature reduction mechanisms to further improve the results. The metrics of these experiments will be conducted by the previously determined best-tuned classifier algorithms. The independent variable in this case, would only be the different feature sets.

7.1.3.1 Manual Selection

For the first trial, we derived a combination of five feature sets from our text database. The first one being the features extracted from all text messages, regardless of whether a text is sent or received. The second and the third ones are features for sent text messages and received text messages, respectively. The fourth one is a joint features set of the sent and received texts. Finally, we also transformed tweets and treated it in a similar fashion as text data, which in turns gives us the fifth feature set. The results of machine learning on those features sets are shown in Figure 47 and Table 14. Doubling the text features by separating them into sent and received texts helped boost the F1 score to an average of 0.83. We then applied this insight of doubling text features to the next trial of experiments, where we derived a set of combination features based on the rankings of the results of the first trial. Namely, they are: 1) all texts and tweets feature set, 2) sent texts and tweets feature set, and 3) sent, received, and tweets combined feature set. The result is shown in Figure 48 and Table 15. The result is not as good as the first trial, because by doubling the features without doing feature selection, the classifier will take all of the less effective features into account. Hence, we need to implement a feature selection technique in order to utilize the benefits of simply doubling the features.

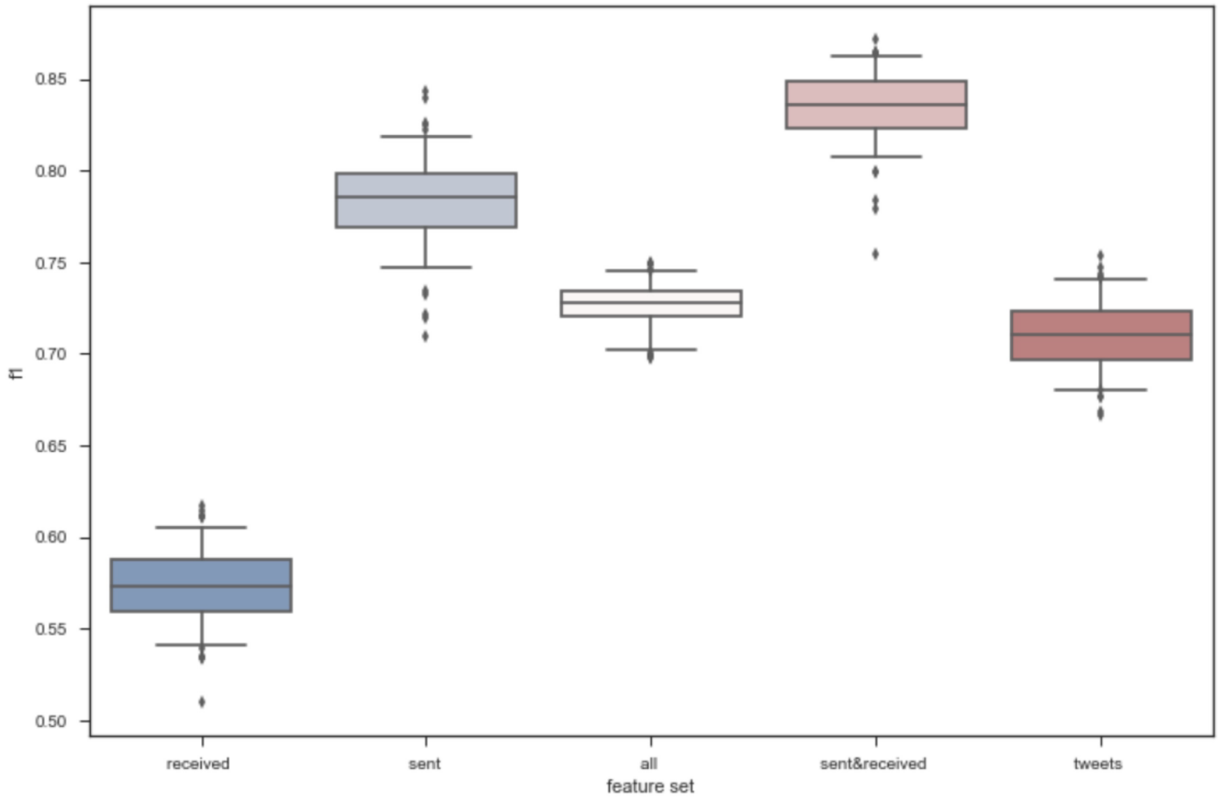


Figure 47: Boxplots of Text Feature Sets Comparison

feature set	Min f1	Mean f1	Max f1
all	0.697356	0.727290	0.749661
received	0.509955	0.572630	0.617415
sent	0.709441	0.782200	0.842779
sent&received	0.754538	0.834453	0.871807
tweets	0.666812	0.710640	0.753804

Table 14: Statistic Summary of Text Feature Sets Comparison

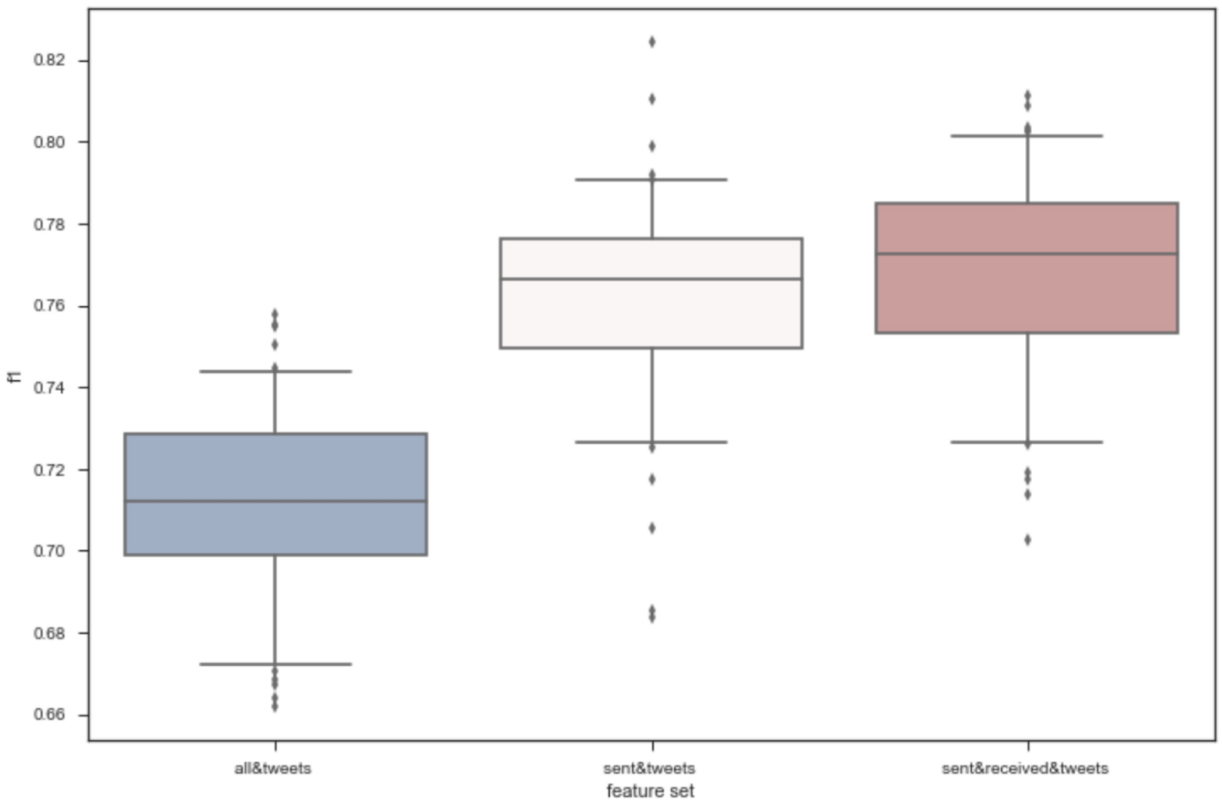


Figure 48: Boxplots of Combined Text Feature Sets Comparison

feature set	Min f1	Mean f1	Max f1
all&tweets	0.661916	0.712493	0.757856
sent&received&tweets	0.702445	0.768124	0.811293
sent&tweets	0.683576	0.762168	0.824485

Table 15: Statistic Summary of Combined Text Feature Sets Comparison

7.1.3.2 Chi-Squared Selection

Here are the results for performing a series of chi-squared feature selections on the combined feature sets. The highest F1 score we obtained from applying feature selection technique is 20 features on the sent, receive and Twitter triple count feature sets. The F1 score ranges from 0.75 to 0.88 with an average of 0.81. In contrast to our hypothesis, this feature selection technique did not significantly improve the F1 score. One explanation we have is that the very first step of hyperparameter tuning optimized the random forest parameters for no feature selection. However, is not optimized for selected features. Still, in conclusion we can say that combining sent, received, and tweets feature sets can significantly improve the machine learning results as long as the number of features for feature selection is controlled under 30.

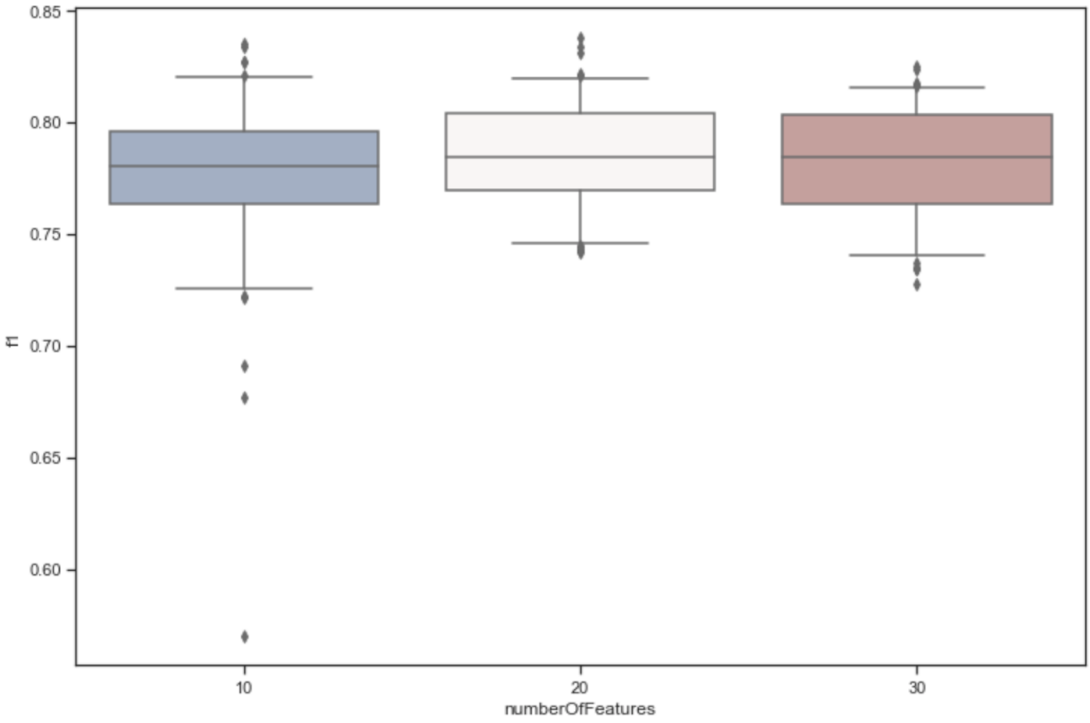


Figure 49: Boxplots of Feature Selection on Sent And Tweets Feature Set

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.570303	0.775615	0.834940
20	0.741485	0.784663	0.837526
30	0.727688	0.781463	0.825044

Table 16: Statistic Summary of Feature Selection on Sent And Tweets Feature Set

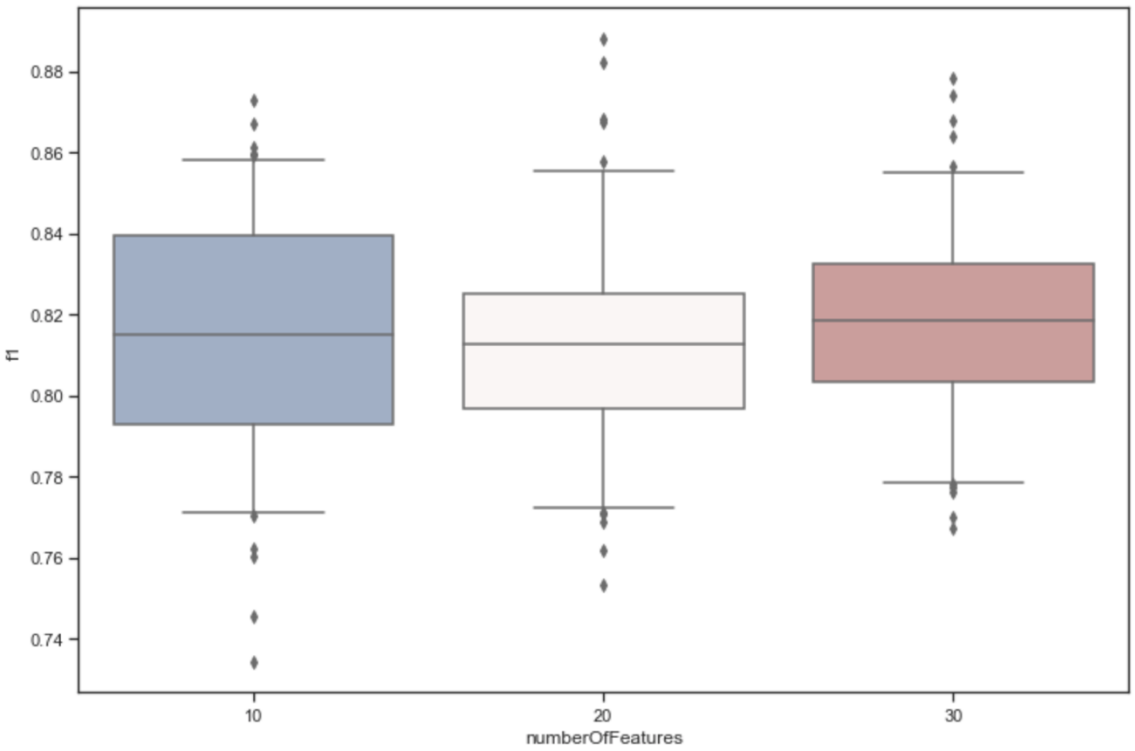


Figure 50: Boxplots of Feature Selection on Sent, Received And Tweets Feature Set

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.734423	0.815522	0.872778
20	0.753427	0.812970	0.887918
30	0.767192	0.818694	0.878156

Table 17: Statistic Summary of Feature Selection on Sent, Received And Tweets Feature Set

7.1.4 Analysis of New Text Data

Here are experiments we conducted for new text data. In these experiments, we reuse the same metrics and procedure for old text data. Although we concluded that Random Forest is our best working classifier algorithm among the seven baseline Algorithms, we decided to include the top three fine-tuned classifiers to test out on the new text data because the new data is not entirely identical to the old one so something might go wrong. In addition, since there are not sufficient participants that fit the best feature selection metrics we had (participants having either sent, receive and tweets or combined are less than 10), we conducted this experiments using features derived from all texts. As shown in figure 51 and table 18, Support Vector Machine (SVC) and Logistic Regression (LR) obtained an average of 0.72. This shows that our machine learning pipeline successfully avoided overfitting and is usable on random test samples. We believe the reason Random Forest does not have a good f1 score is because of that the algorithm itself is needy on the number of training sets, and for the new text data, we only have 100 participants. Given enough text data, random forest can still be promising.

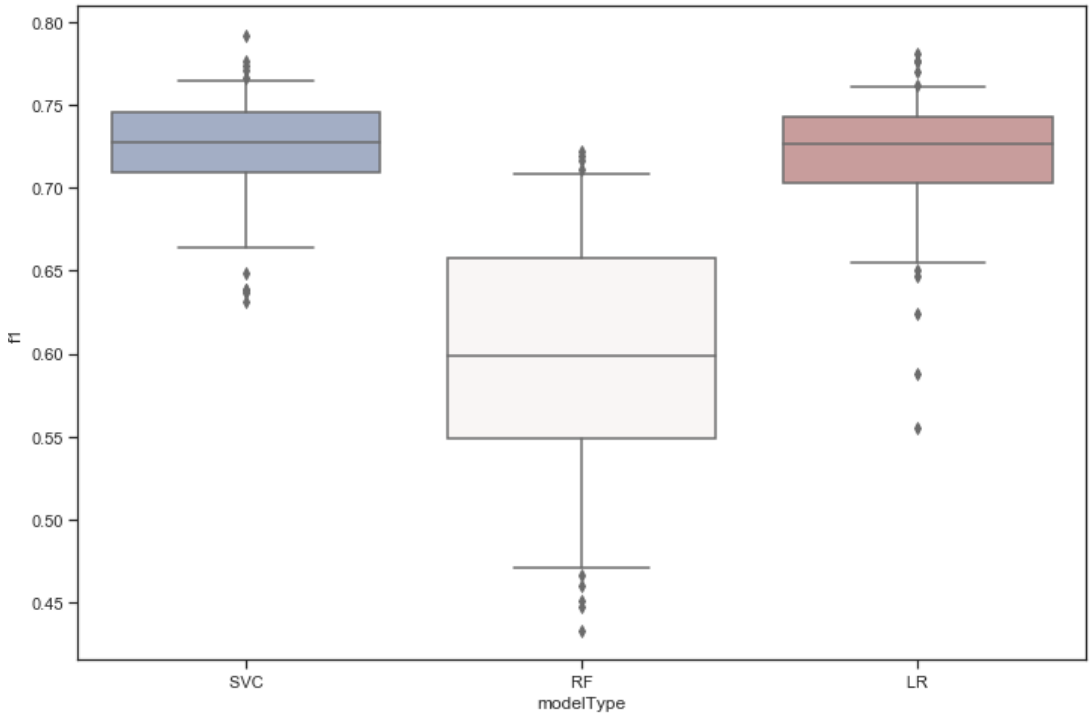


Figure 51: Boxplots of Tuned Classifiers For Text Features Comparison (New Data)

modelType	Min f1	Mean f1	Max f1
Logistic Regression(LR)	0.554762	0.718094	0.781169
Random Forests(RF)	0.433333	0.599542	0.721905
Support Vector Machine(SVC)	0.630952	0.723828	0.791429

Table 18: Statistic Summary of Tuned Classifiers For Text Features Comparison (New Data)

7.1.5 Text Features Discussion

After finishing the two parts of experiments for text features machine learning, the highest average F1 score we managed to obtain is 0.83. Our prediction is that this result may be further increased if we reimplement hyperparameter optimization on random forest for this particular feature set. We repeated the procedure of hyperparameter optimization specifically for RF with 10 feature selections. After that, we ran a various feature reduction algorithm 100 times each with 10 features. The highest result we have is using Recursive Feature Reduction (RFE) with an average of 0.83, which is the same as we had before. However, we managed to increase the lower bound from 0.76 to 0.78 and upper bound from 0.89 to 0.9. However, our experiments did not address on testing out overfitting problem, so accordingly we collected new text data to test out our model. As mentioned in 7.1.4, the model performs well on the new text data and is almost as good as the old ones. In this way, we can safely say that our model refrained from overfitting.

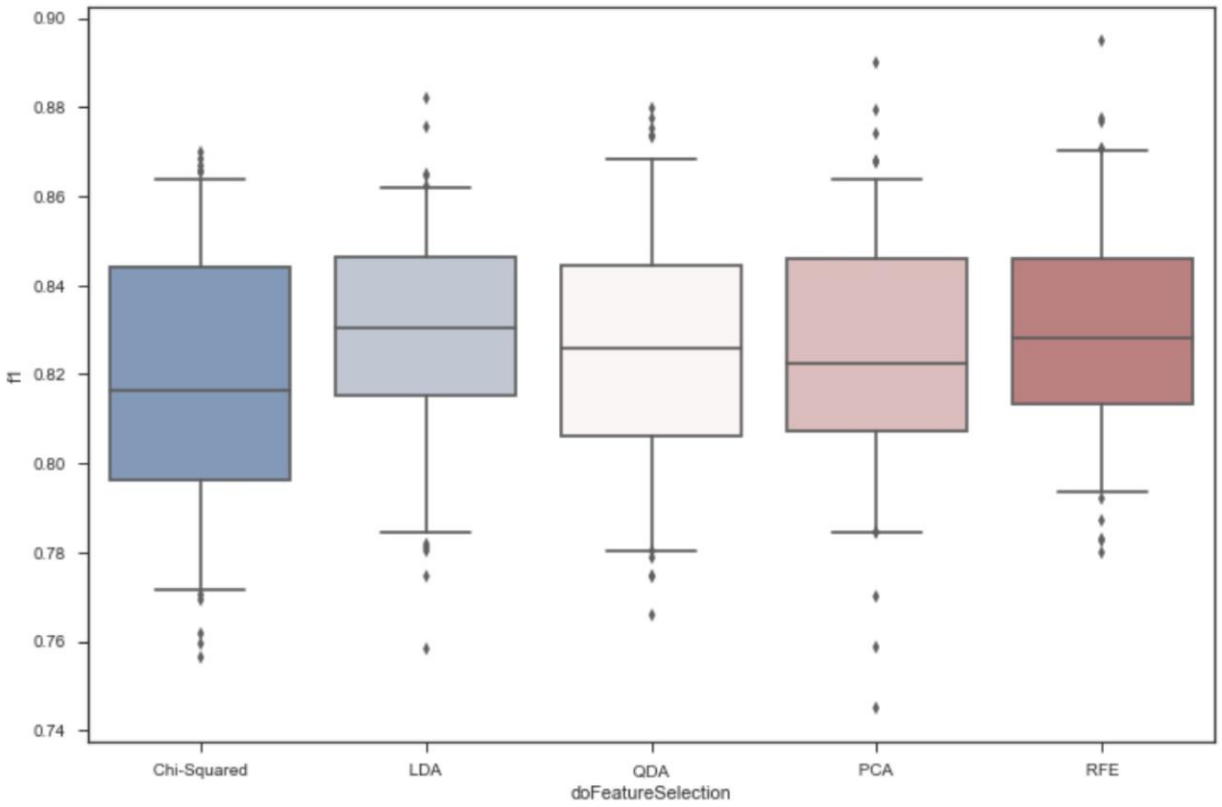


Figure 52: Boxplots of Feature Selection on Sent, Received And Tweets Feature Set

Feature Selection Type	Min f1	Mean f1	Max f1
Chi-Squared	0.756335	0.818840	0.869917
Linear Discriminant Analysis	0.758482	0.828790	0.882123
Principle Component Analysis	0.745046	0.825188	0.890023
Quadratic Discriminant Analysis	0.766110	0.824564	0.879873
Recursive Feature Reduction	0.780018	0.829194	0.895032

Table 19: Statistic Summary of Feature Selection on Sent, Received And Tweets Feature Set

7.2 Audio Features

7.2.1 Performance Analysis on Different Classifiers

Similar to our approach for text features, we first did grid search on seven classifiers and ran experiments on them. Below is the results we got for audio features, and the classifiers with their optimized parameters.

- SVC: C: 0.001, kernel: rbf, gamma: 0.1
- kNN: n_neighbors: 1, leaf_size: 1
- RF: max_depth: 2, min_samples_leaf: 3, min_samples_split: 5, n_estimators: 500
- XGB: min_child_weight: 5, gamma: 0, subsample: 0.8, colsample_bytree: 0.6, max_depth: 2
- LR: C: 100, solver: lbfgs, multi_class: ovr, random_state: 0
- GP: max_iter_predict: 1 ,n_jobs: 1
- NN: solver: ‘sgd’, learning_rate_init: 0.001, max_iter: 100, learning_rate: ‘adaptive’

After obtaining these tuning sets for audio features, we ran the next experiment with classifiers and parameters 100 times each. The results can be seen in Figure 53 and Table 20.

XGBoost has the best performance across all seven baseline algorithms in regards to both scores and stability. Having a maximum F1 score of 0.59, a mean F1 of 0.50 and a minimum F1 of 0.38, XGBoost is slightly better than the other algorithms. So overall, fine-tuned XGBoost has the best performance among all seven classifying algorithms within the scope of audio features, setting the cutoff to 10 and not doing any feature selection.

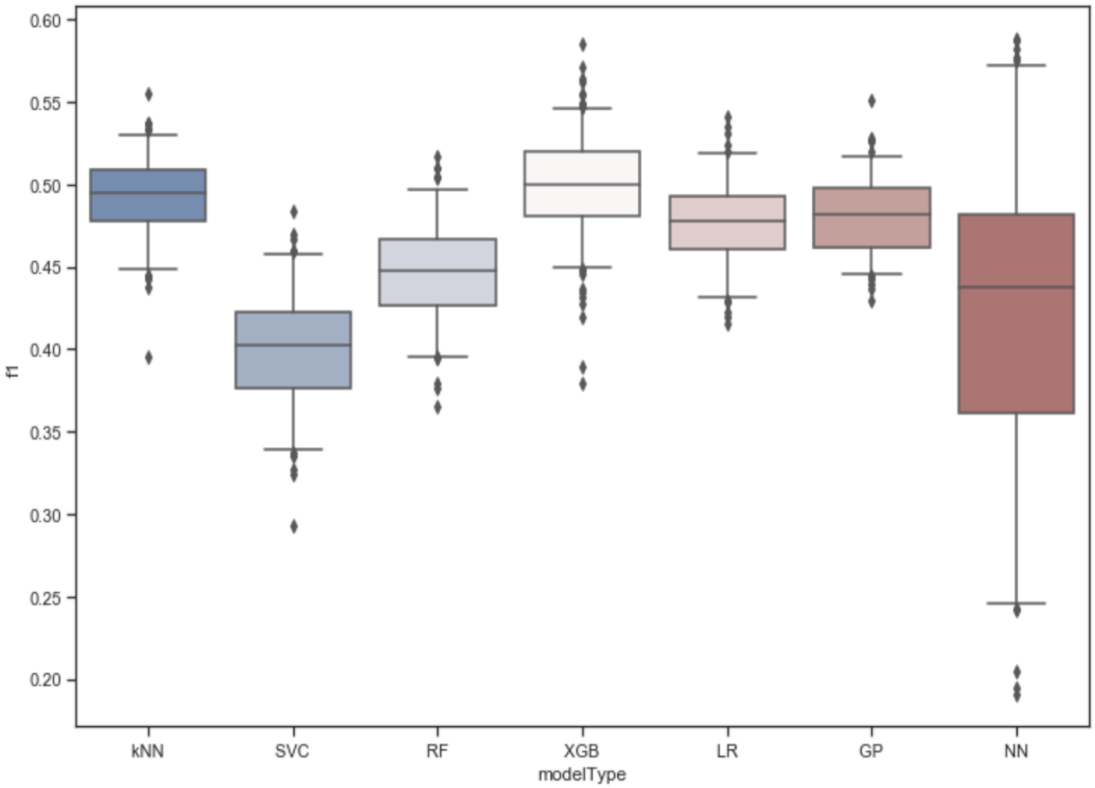


Figure 53: Tuned Classifiers For Audio Features Comparison

modelType	Min f1	Mean f1	Max f1
Gaussian Process(GP)	0.429381	0.481107	0.551396
Logistic Regression(LR)	0.415335	0.477141	0.540885
Neural Network(NN)	0.190934	0.422131	0.588009
Random Forests(RF)	0.365091	0.446387	0.516457
Support Vector Machine(SVC)	0.292959	0.399635	0.483743
XGBoost(XGB)	0.379229	0.499229	0.585513
k-Nearest Neighbours(k-N N)	0.395098	0.492418	0.555178

Table 20: Statistic Summary of Tuned Classifiers For Audio Features Comparison

7.2.2 Performance Analysis on Different Feature Sets

This part of the experiment aimed to find the most ideal sets of features for machine learning results. To do so, we divided the process into two parts. First, we experiment on various possible combinations of manual features. After finding the feature set with better performance, we then run experiments using different feature reduction mechanisms to further improve the results. The metrics of these experiments are conducted by the previously determined best-tuned classifier algorithms. The independent variable in this case, would only be the different feature sets.

7.2.2.1 Manual Selection

Since we used different tools and scripts to extract different kinds of audio features, we finally got five feature sets. These sets are: 1) OpenSmile feature set, 2) Praat signal feature set, 3) Praat pause time feature set, 4) Praat combined feature set, and 5) all audio feature set. The results of machine learning on those features sets are shown in Figure 54 and Table 21. With the exception of the Praat pause time feature set having a slightly better performance, all of the other

feature sets have similar results. Therefore, we need to implement some feature selection techniques in order to improve performance and determine the best feature set.

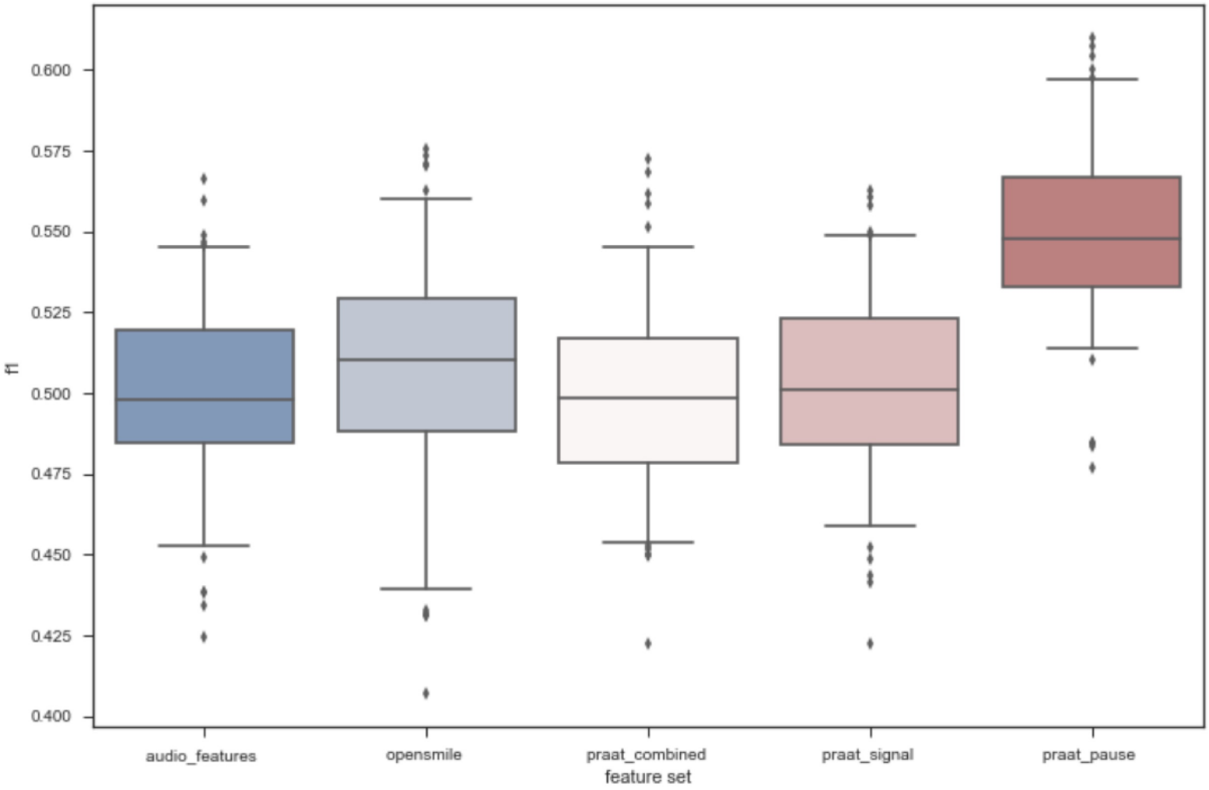


Figure 54: Boxplots of Audio Feature Sets Comparison

feature set	Min f1	Mean f1	Max f1
audio_features	0.424549	0.500289	0.566494
opensmile	0.407006	0.507170	0.575741
praat_combined	0.422421	0.498210	0.572492
praat_pause	0.476670	0.549319	0.610115
praat_signal	0.422492	0.501252	0.562531

Table 21: Statistic Summary of Audio Feature Sets Comparison

7.2.2.2 Chi-Squared Selection

Following are the results for performing a series of chi-squared feature selection on every feature set except for the Praat pause time feature set (Figure 55-60, Table 22-27), which we didn't do feature selection on because it only contains eight features. The best F1 scores we obtained from applying feature selection technique occurred when selecting 50 features from all audio features. The F1 score ranges from 0.55 to 0.67 with an average of 0.62.

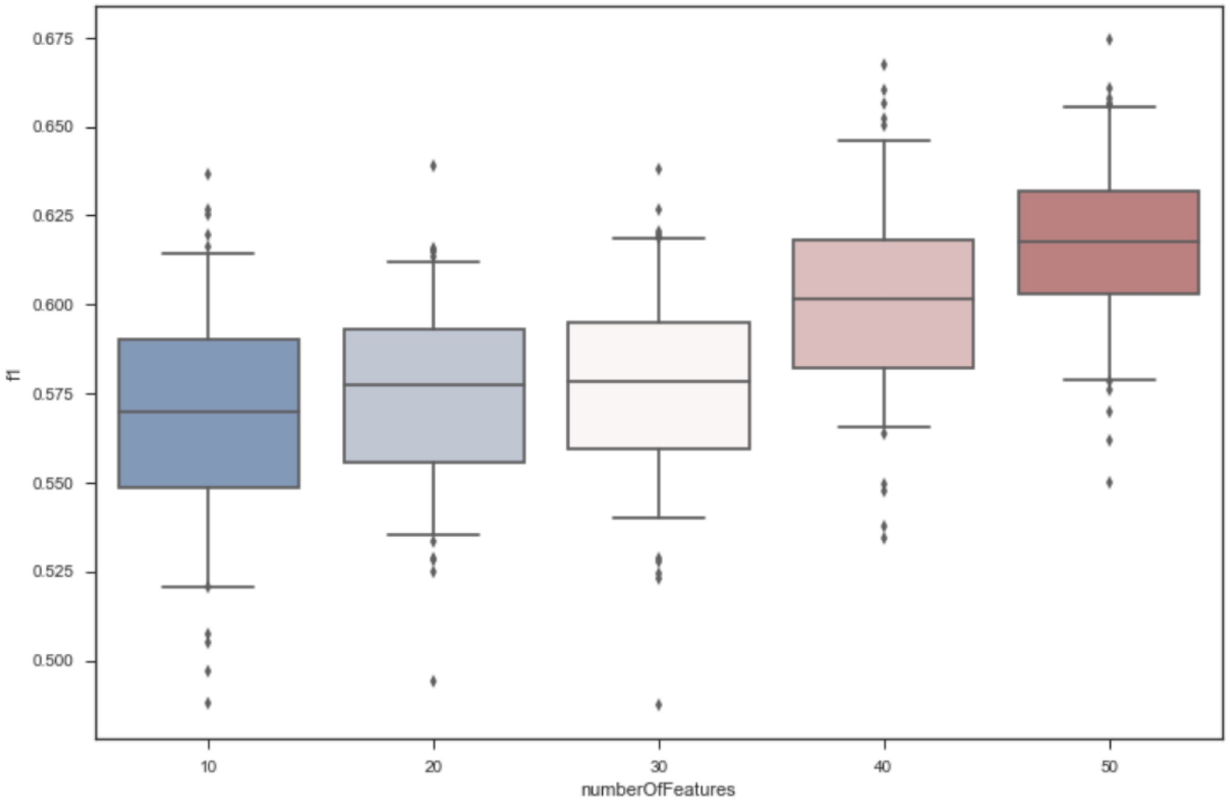


Figure 55: Boxplots of Feature Selection on Audio All Feature Set (# of features <= 50)

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.487724	0.568189	0.636575
20	0.494194	0.573660	0.638961
30	0.487445	0.577875	0.637845
40	0.534536	0.601410	0.667149
50	0.549894	0.617667	0.674601

Table 22: Statistic Summary of Feature Selection on Audio All Feature Set (# of features <= 50)

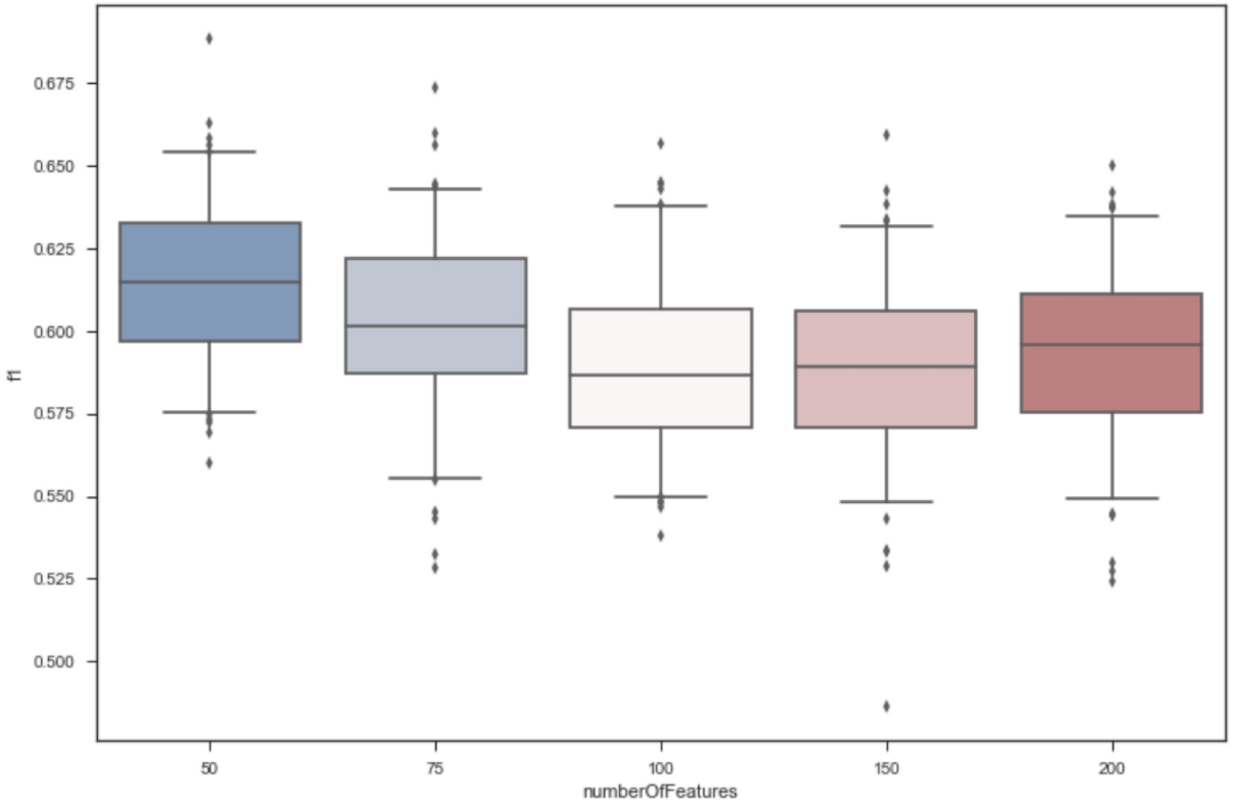


Figure 56: Boxplots of Feature Selection on Audio All Feature Set (# of features >= 50)

numberOfFeatures	Min f1	Mean f1	Max f1
50	0.560219	0.615613	0.688529
75	0.528483	0.602372	0.673661
100	0.538065	0.588790	0.656463
150	0.486372	0.588361	0.658996
200	0.524482	0.593387	0.650012

Table 23: Statistic Summary of Feature Selection on Audio All Feature Set (# of features >= 50)

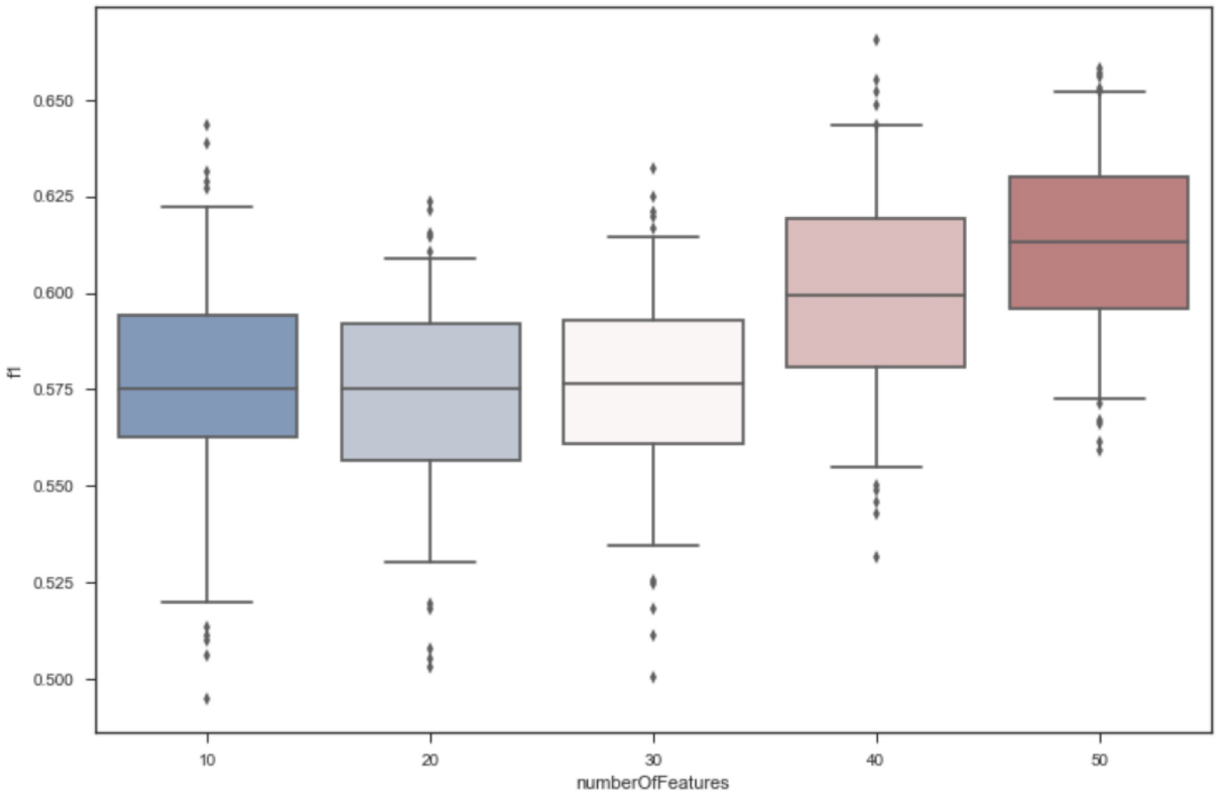


Figure 57: Boxplots of Feature Selection on OpenSmile Feature Set (# of features <= 50)

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.494821	0.574839	0.643497
20	0.503112	0.572653	0.623705
30	0.500338	0.575066	0.632248
40	0.531561	0.600455	0.665617
50	0.559070	0.613854	0.658168

Table 24: Statistic Summary of Feature Selection on OpenSmile Feature Set (# of features <= 50)

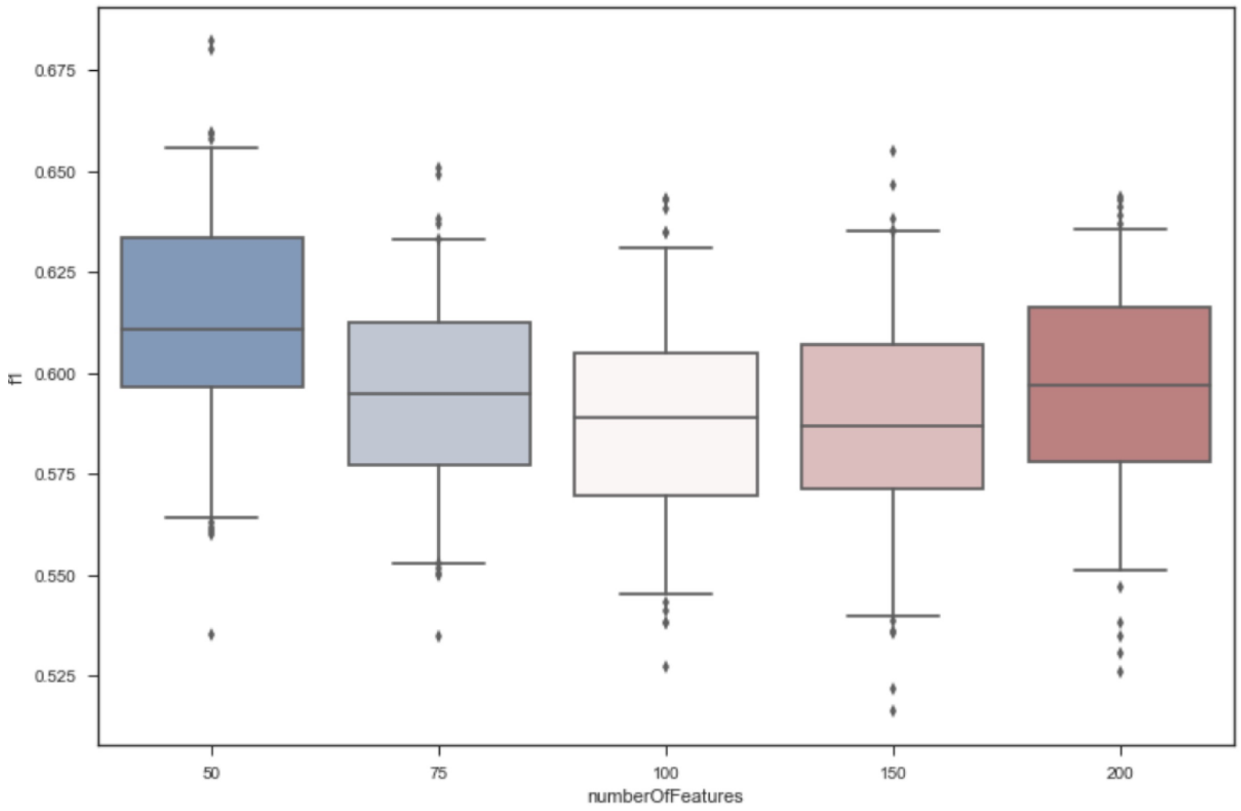


Figure 58: Boxplots of Feature Selection on OpenSmile Feature Set (# of features >= 50)

numberOfFeatures	Min f1	Mean f1	Max f1
50	0.535068	0.613255	0.682371
75	0.534932	0.595054	0.650738
100	0.527149	0.588128	0.643355
150	0.516313	0.588105	0.654787
200	0.525983	0.595509	0.643536

Table 25: Statistic Summary of Feature Selection on OpenSmile Feature Set (# of features >= 50)

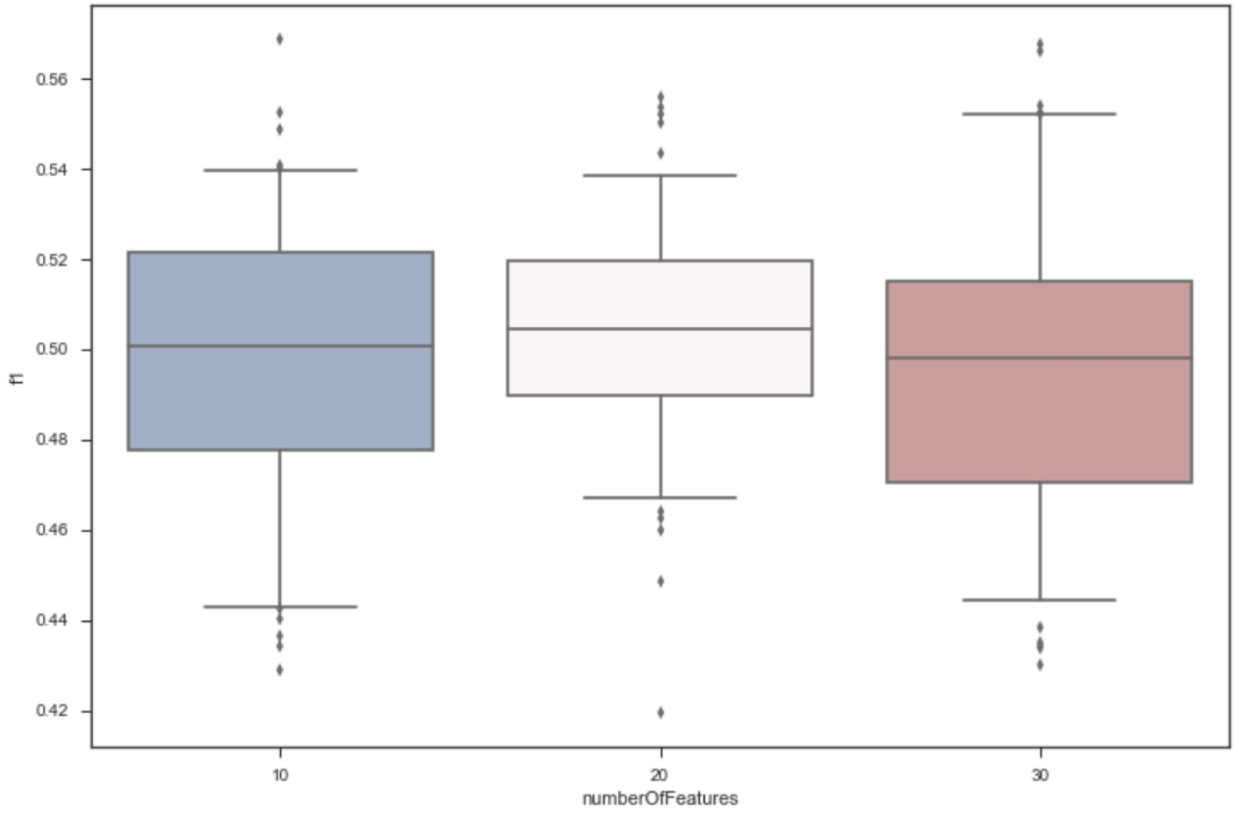


Figure 59: Boxplots of Feature Selection on Praat Combined Feature Set

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.429078	0.497209	0.568725
20	0.419438	0.504181	0.555584
30	0.430030	0.494870	0.567296

Table 26: Statistic Summary of Selection on Praat Combined Feature Set

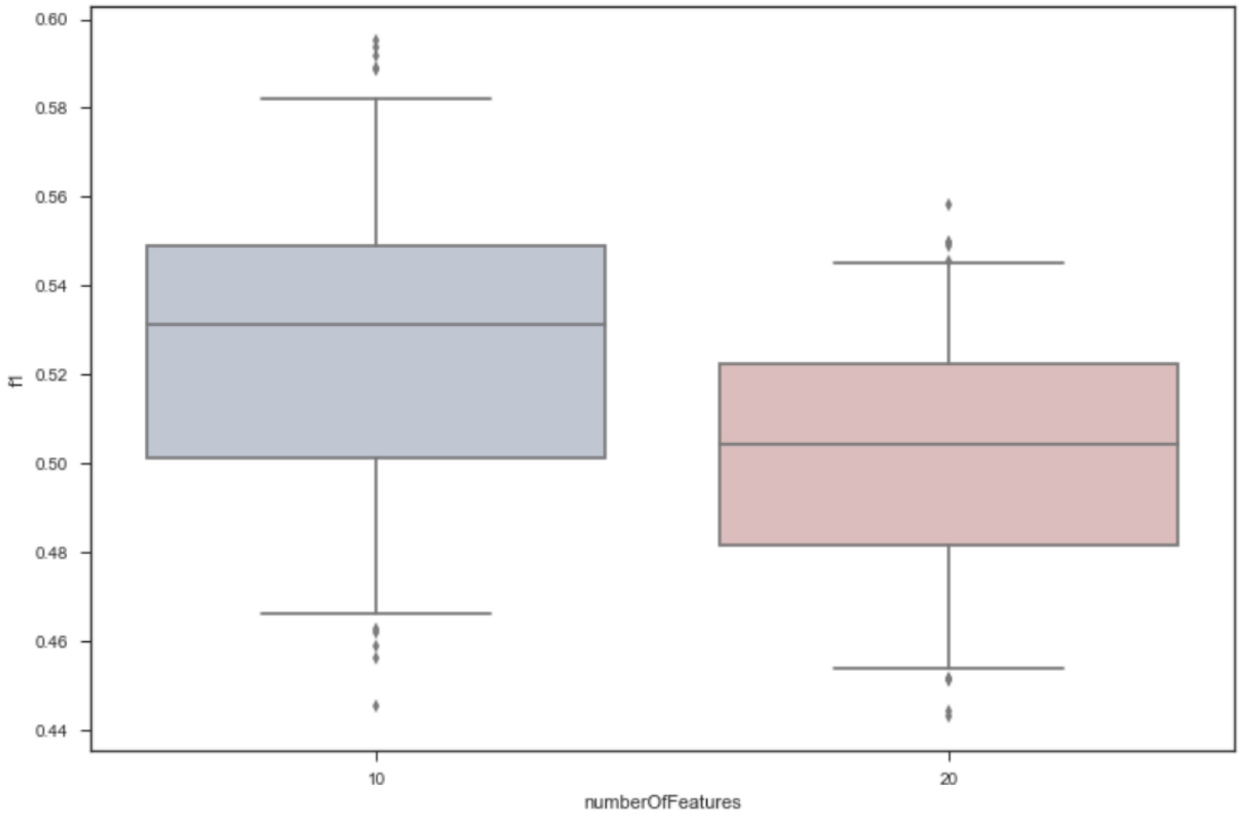


Figure 60: Boxplots of Feature Selection on Praat Signal Feature Set

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.445433	0.527234	0.595328
20	0.443014	0.501267	0.558091

Table 27: Statistic Summary of Feature Selection on Praat Signal Feature Set

7.2.3 Analysis of New Audio Data

Here are a series of experiments we conducted for new audio data. In these experiments, we reused the same metrics and procedure for old audio data. The first part will be hyperparameter optimization for different classifiers. After that, we will compare across to find the best classifiers.

We first did grid search on seven classifiers and ran experiments on them. Below is the results we got for audio features, and the classifiers with their optimized parameters.

- SVC: C: 0.001, kernel: rbf, gamma: 10
- kNN: n_neighbors: 1, leaf_size: 1
- RF: max_depth: 5, min_samples_leaf: 3, min_samples_split: 3, n_estimators: 20
- XGB: min_child_weight: 1, gamma: 0.1, subsample: 0.8, colsample_bytree: 0.6, max_depth: 4
- LR: C: 0.01, solver: lbfgs, multi_class: ovr, random_state: 0
- GP: max_iter_predict: 1 ,n_jobs: 1
- NN: solver: ‘sgd’, learning_rate_init: 0.001, max_iter: 100, learning_rate: ‘adaptive’

After obtaining these tuning sets for audio features, we ran the next experiment with classifiers and parameters 100 times each. The results can be seen in Figure 61 and Table 28. kNN has the best performance across all seven baseline algorithms in regards to both scores and stability. Having a maximum F1 score of 0.63, a mean F1 of 0.55 and a minimum F1 of 0.49, kNN is better than the other algorithms. So overall, fine-tuned kNN has the best performance among all seven classifying algorithms within the scope of audio features, setting the cutoff to 10 and not doing any feature selection.

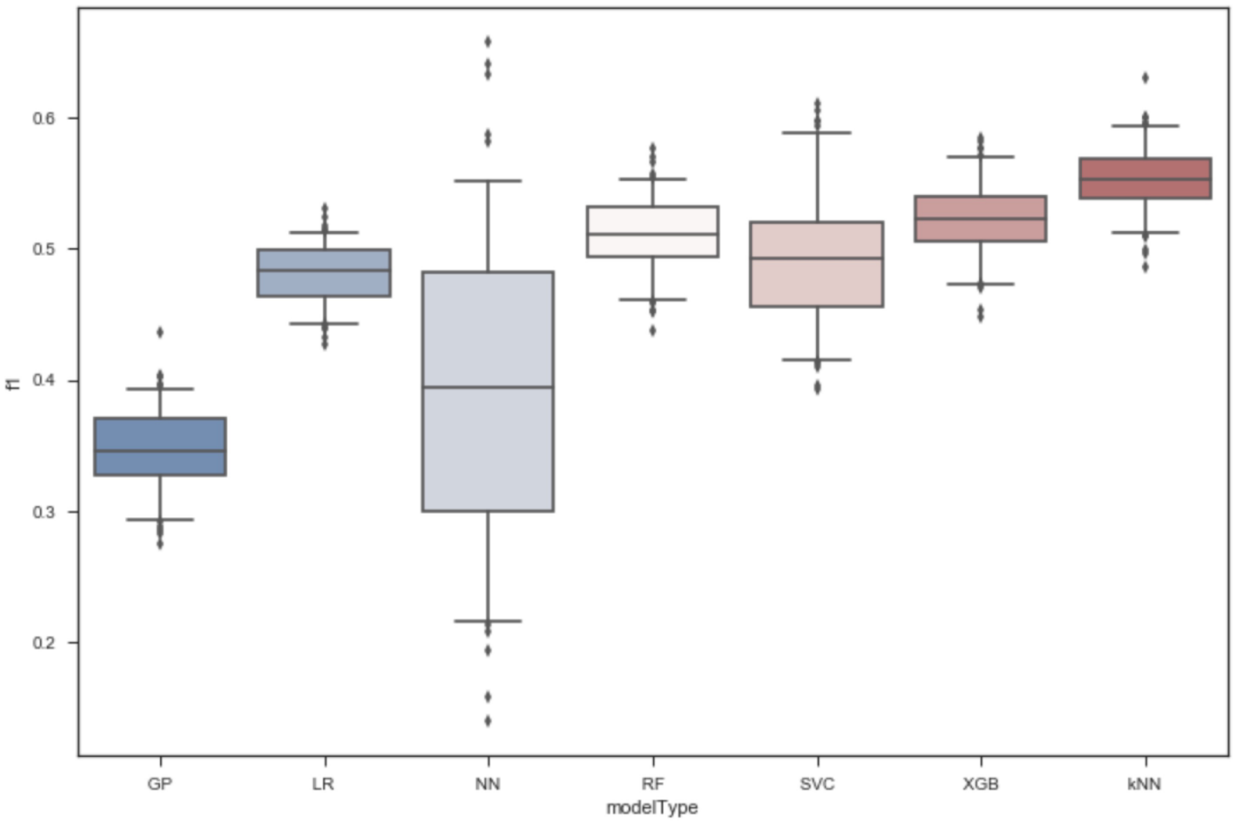


Figure 61: Boxplots of Tuned Classifiers For Audio Features Comparison (New Data)

modelType	Min f1	Mean f1	Max f1
Gaussian Process(GP)	0.274525	0.346430	0.436341
Logistic Regression(LR)	0.426374	0.480121	0.530222
Neural Network(NN)	0.139922	0.389351	0.657882
Random Forests(RF)	0.436923	0.509296	0.576096
Support Vector Machine(SVC)	0.392636	0.493979	0.610448
XGBoost(XGB)	0.448412	0.521338	0.583748
k-Nearest Neighbours(k-N N)	0.485366	0.553492	0.630063

Table 28: Statistic Summary of Tuned Classifiers For Audio Features Comparison (New Data)

The second part will be the experiments on finding the best feature sets. We have six manual feature sets for the new audio data:

- d1: Old audio data (reading speech)
- d2: New audio data (reading speech)
- d3: New audio data (open response)
- d1traind2test: d1 as the training set, d2 as the testing set
- Stackedd1d2: Combined audio data (without open response)
- Joinedd2d3: New audio data (includes both reading speech and open response) from the same participant, in other words, double the number of features by extracting two sets of features from two audio files.

We used kNN to run experiments on each dataset for 100 times. As shown in Figure 62 and Table 29 stacked d1 and d2 has the best performance in that the max f1 is 0.61, min f1 is 0.51 and the average is 0.56. In this way, we managed to keep the f1 above 0.5. We can conclude that stacking d1 and d2 improves the f1 score.

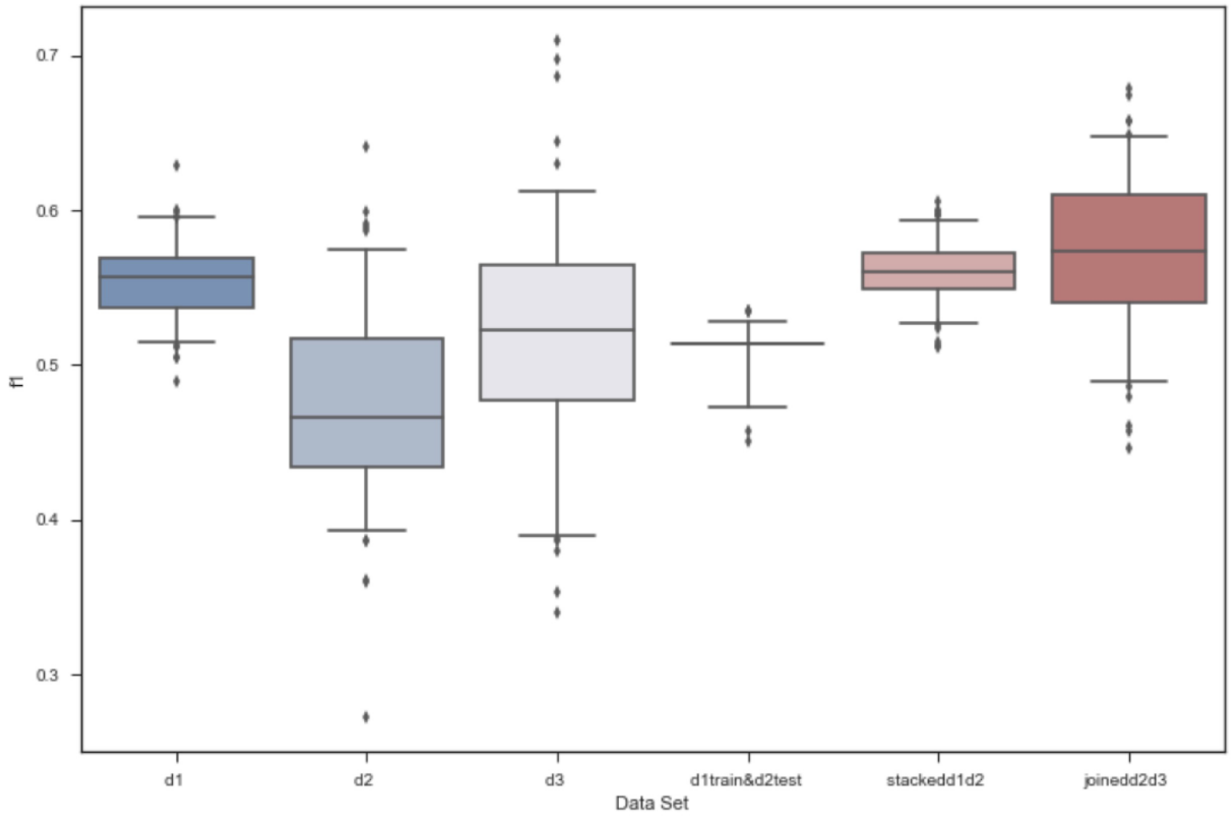


Figure 62: Boxplots of Audio Data Sets Comparison (New Data)

Data Set	Min f1	Mean f1	Max f1
d1	0.489578	0.554289	0.628926
d1train&d2test	0.450704	0.509472	0.535211
d2	0.272140	0.474323	0.640693
d3	0.339394	0.516957	0.710000
joinedd2d3	0.445758	0.572369	0.678392
stackedd1d2	0.511715	0.559028	0.605123

Table 29: Statistic Summary of Audio Data Sets Comparison (New Data)

7.2.4 Audio Features Discussion

In conclusion, experiments show that the best performance for machine learning on audio features occurs under the following conditions: using XGBoost as the algorithm, 50 features for feature selection, and using the dataset that is generated by OpenSMILE. Since we did grid search on all audio features, which mainly consisted of OpenSMILE features, the tuned classifiers might be not very fit for Praat features. Still, the result for pause time features generated by Pratt is not optimum as the manual recording of participants reading one sentence has too many limitations. In order to have better results, we want data that includes more than just one sentence, and preferably is spontaneous. Therefore we modified our data collection methods on the mobile application. Accordingly, we reimplemented the whole hyperparameter tuning and manual feature selection process for Praat features on the new data. This turned out to be satisfying as the f1 is improved from an average of 0.46 to 0.56 with only 91 participants. It is promising to further collect spontaneous speech or open response.

7.3 GPS Features

7.3.1 Performance Analysis on Different Classifiers

Similar to our approach for text and audio features, we first did gird search on seven classifiers and ran experiments on them. Below is the results we got for GPS features, and the classifiers with their optimized parameters.

- SVC: C: 0.001, kernel: rbf, gamma: 0.001
- kNN: n_neighbors: 9, leaf_size: 1
- RF: max_depth: 3, min_samples_leaf: 3, min_samples_split: 2, n_estimators: 500
- XGB: min_child_weight: 5, gamma: 5, subsample: 0.8, colsample_bytree: 0.8, max_depth: 2
- LR: C: 0.01, solver: ‘lbfgs’, multi_class: ‘ovr’, random_state: 0
- GP: max_iter_predict: 1 ,n_jobs: 1
- NN: solver: ‘sgd’, learning_rate_init: 0.001, max_iter: 50, learning_rate: ‘constant’

After obtaining these tuning sets for GPS, we ran the next experiment with classifiers and parameters 100 times each. The results can be seen in Figure 63 and Table 30. SVM has the best performance across all seven baseline algorithms in terms of both scores and stability. Having a maximum F1 score of 0.67, a mean F1 of 0.63 and a minimum F1 of 0.58, RF significantly outperforms other algorithms in all three measurements. Overall, fine-tuned SVC has the best performance among all seven classifying algorithms within the scope of GPS features, setting the cutoff to 10, and not doing any feature selection.

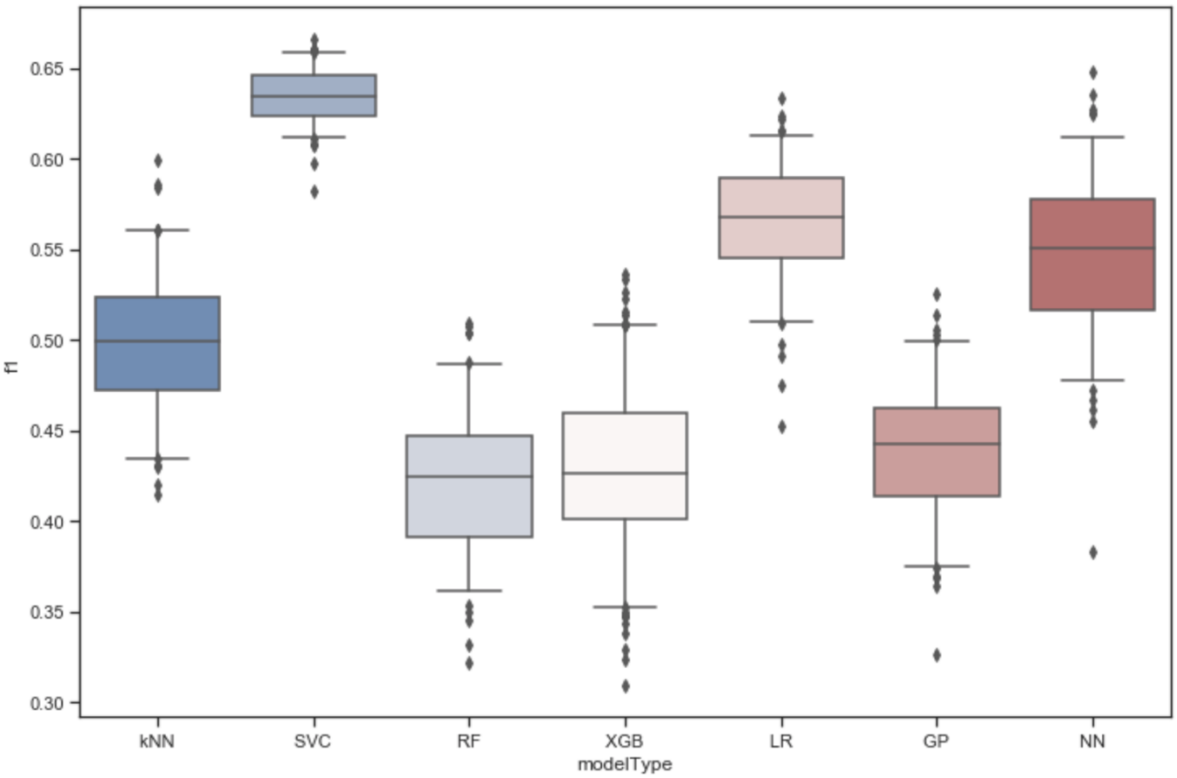


Figure 63: Tuned Classifiers For GPS Features Comparison

modelType	Min f1	Mean f1	Max f1
Gaussian Process(GP)	0.326678	0.438645	0.525659
Logistic Regression(LR)	0.452311	0.565791	0.633225
Neural Network(NN)	0.383203	0.547470	0.647825
Random Forests(RF)	0.321543	0.420601	0.509405
Support Vector Machine(SVC)	0.581757	0.634118	0.665705
XGBoost(XGB)	0.309634	0.429783	0.536516
k-Nearest Neighbours(k-N N)	0.414804	0.497717	0.599319

Table 30: Statistic Summary of Tuned Classifiers For GPS Features Comparison

7.3.2 Performance Analysis on Different Feature Sets

This part of the experiment aimed to find the most ideal sets of features for GPS machine learning results. Identical to the previous experiment sets, we divided the process into two parts. First, we experiment on various possible combinations of manual features. Secondly, we run experiments using different feature reduction mechanisms to further improve the results. The metrics of these experiments will be conducted by the previously determined best-tuned classifier algorithms. The independent variable, in this case, would only be the different feature sets.

7.3.2.1 Manual Selection

For the first trial, we derived a combination of three feature sets from our GPS database; the first one being the features extracted from raw GPS data. The second was the activity features we created based on the raw data. The third one is a joint feature sets of the first two. The results of machine learning on those features sets are shown in Figure 64-67 and Table 31-3. Unlike text features, simply doubling the feature does not boost the F1 score but instead, it stabilized it for a bit. We can see that while having similar average scoring, the upper and lower bounds of the joint feature set is steadier. We presume the reason for raw data features not having an optimum outcome to be not having an accurate timestamp for each coordinate pair. Instead, we did some manual calculation to add uniform timestamps based on each activity's time interval. Next, we implemented feature selection techniques in order to further test these feature sets.

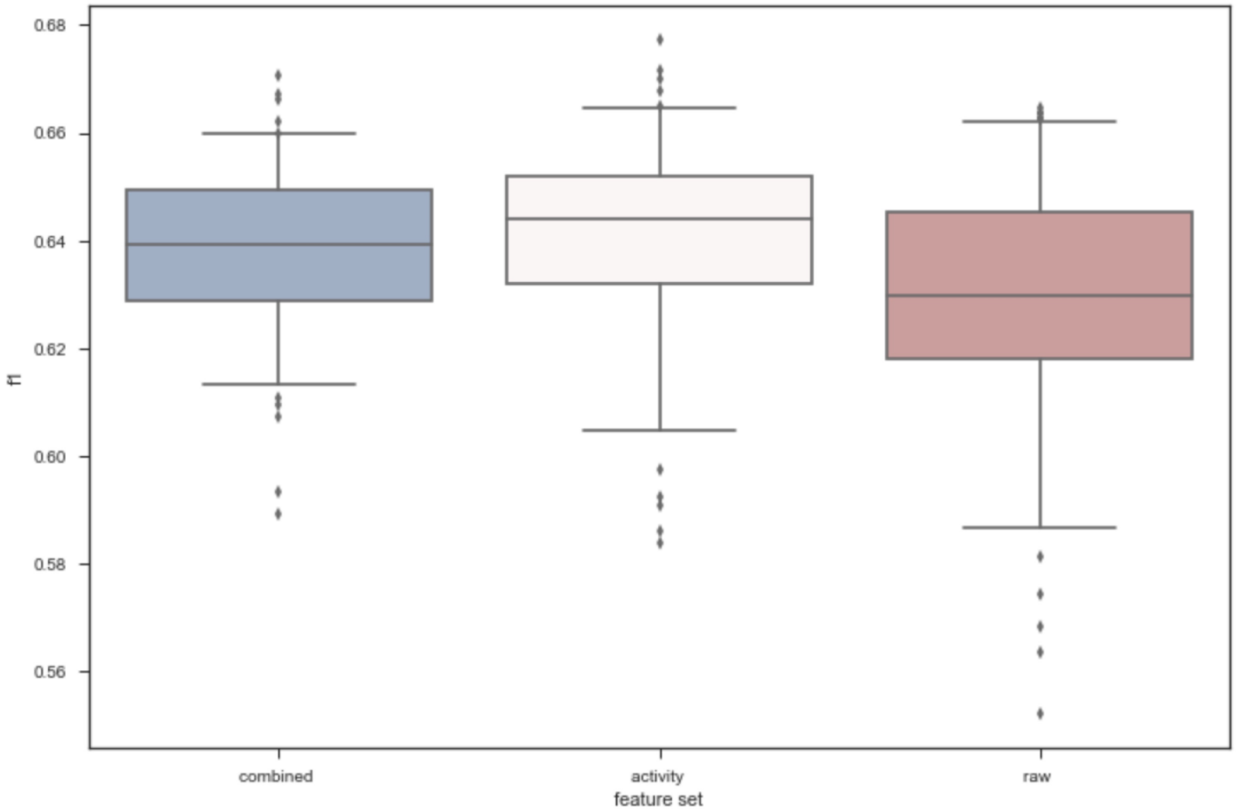


Figure 64: Boxplots of GPS Feature Sets Comparison

feature set	Min f1	Mean f1	Max f1
activity	0.583906	0.641197	0.677403
combined	0.589389	0.636903	0.670671
raw	0.552231	0.628471	0.664581

Table 31: Statistic Summary of GPS Feature Sets Comparison

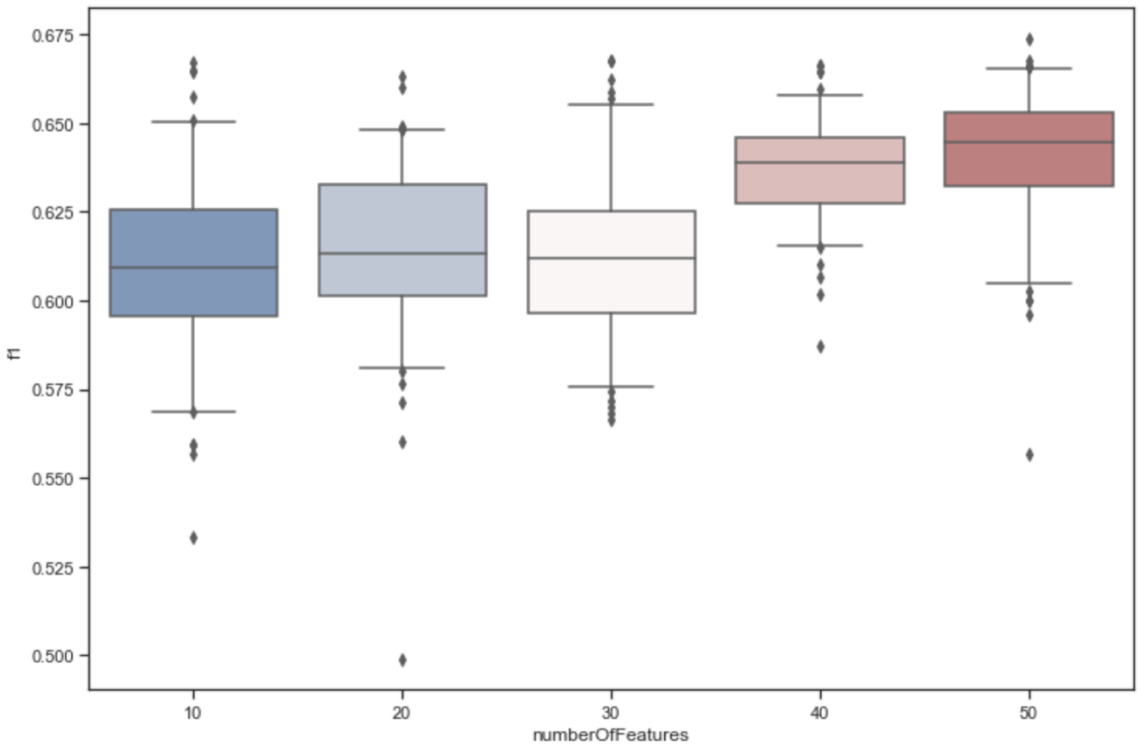


Figure 65: Boxplots of Feature Selection on Activity Feature Set

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.533202	0.611181	0.667329
20	0.499031	0.614829	0.663187
30	0.566646	0.612166	0.667576
40	0.587369	0.636879	0.666478
50	0.556900	0.641166	0.673718

Table 32: Statistic Summary of Feature Selection on Activity Feature Set

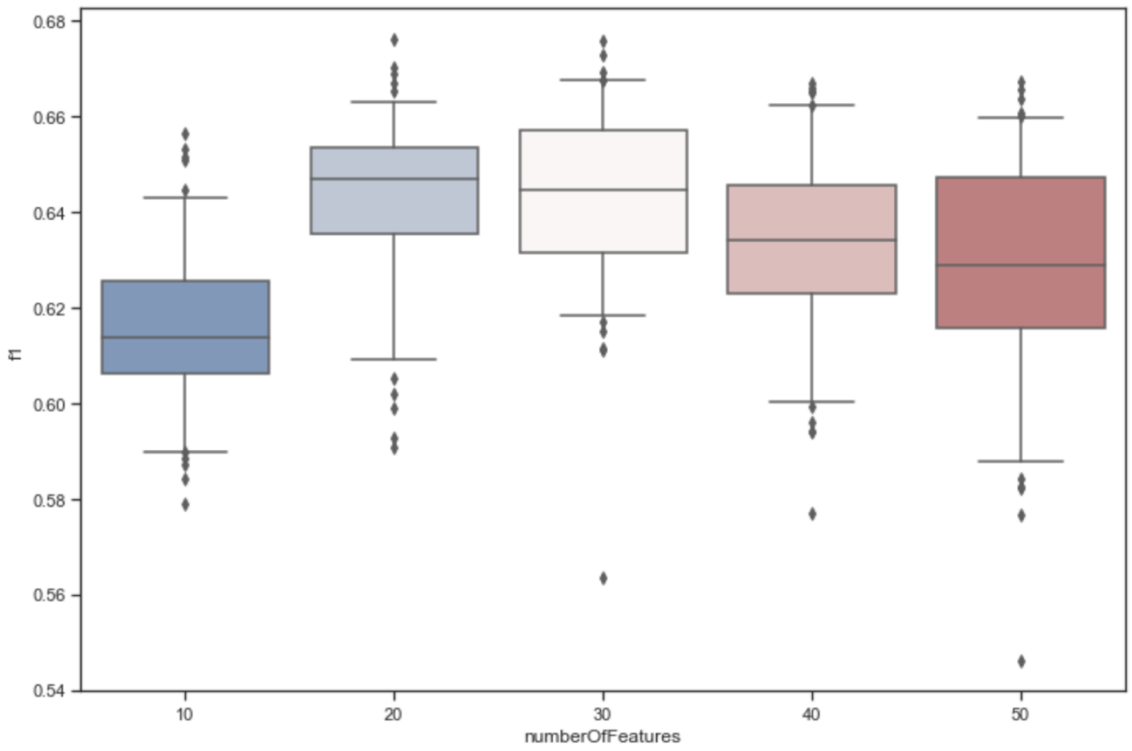


Figure 66: Boxplots of Feature Selection on Raw GPS Feature Set

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.578827	0.615772	0.656450
20	0.590768	0.643049	0.676030
30	0.563445	0.643683	0.675891
40	0.577122	0.633770	0.667032
50	0.546265	0.628282	0.667114

Table 33: Statistic Summary of Feature Selection on Raw GPS Feature Set

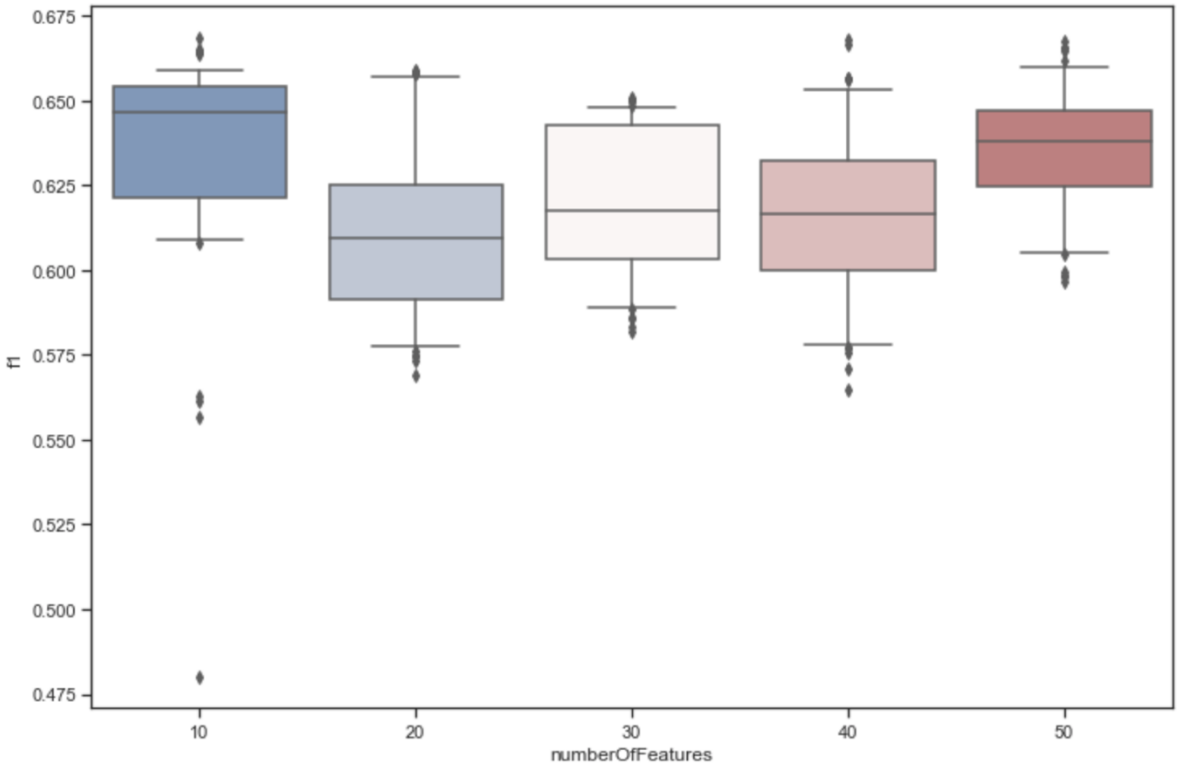


Figure 67: Boxplots of Feature Selection on Combined GPS Feature Set

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.480238	0.637397	0.668369
20	0.568955	0.611876	0.658825
30	0.581984	0.619348	0.650732
40	0.564953	0.616031	0.667924
50	0.596484	0.635820	0.667488

Table 34: Statistic Summary of Feature Selection on Combined GPS Feature Set

7.3.2.2 Chi-Squared Selection

Here are the results for performing a series of chi-squared feature selections on the combined feature sets. The highest F1 score we obtained from applying feature selection techniques is 40 features on the activity feature sets. The F1 score ranges from 0.58 to 0.66 with

an average of 0.61. In contrast to our hypothesis, the feature selection technique did not significantly improve the F1 score. In fact, the best score comes from no feature selection for the combined feature set where the average of F1 is 0.63 in a range of 0.58 to 0.67. This is very likely due to GPS not having features that are distinctly better than others.

7.3.3 GPS Features Discussion

After finishing the two parts of experiments for GPS feature machine learning, the highest average F1 score we managed to obtain is 0.63. This result may be further increased if the raw data was a little more accurate with respect to timestamp and coordinate pairs. If we can find a better way to get the timestamp for the coordinates, we would be able to get more accurate features. Also, reimplementing hyperparameter optimization on SVC for this particular feature set might also increase the results. Since MTurk's policy did not allow a further collection of GPS data, there won't be more new data for our machine learning experiments at this moment.

7.4 All Features

For the last part of this experiment, we decided to test out the possibility of combining all features together. We set up an all feature file with all IDs, making a total of 1881 feature columns. Firstly, since out of all the feature files, we have text having the highest F1 score on average, we decided to use the parameter setting for text features in this experiment. Secondly, we chose to experiment using SVC with rbf kernel because it is the strongest classifier considering its performance in all three feature sets. Lastly, we set up this experiment in two parts. The former to find a number of features that have higher performance, and the latter to perform various feature reduction techniques to squeeze as much as possible. We ran each trial 100 times, and as shown in Table 35, 36, and 37, the F1 score is highest when we do a chi-square top 60 feature selection, ranging from 0.60 to 0.78 with an average of 0.70. Considering this result overlooks everything in audio and GPS, we decided to proceed to the next step of this part of the experiment.

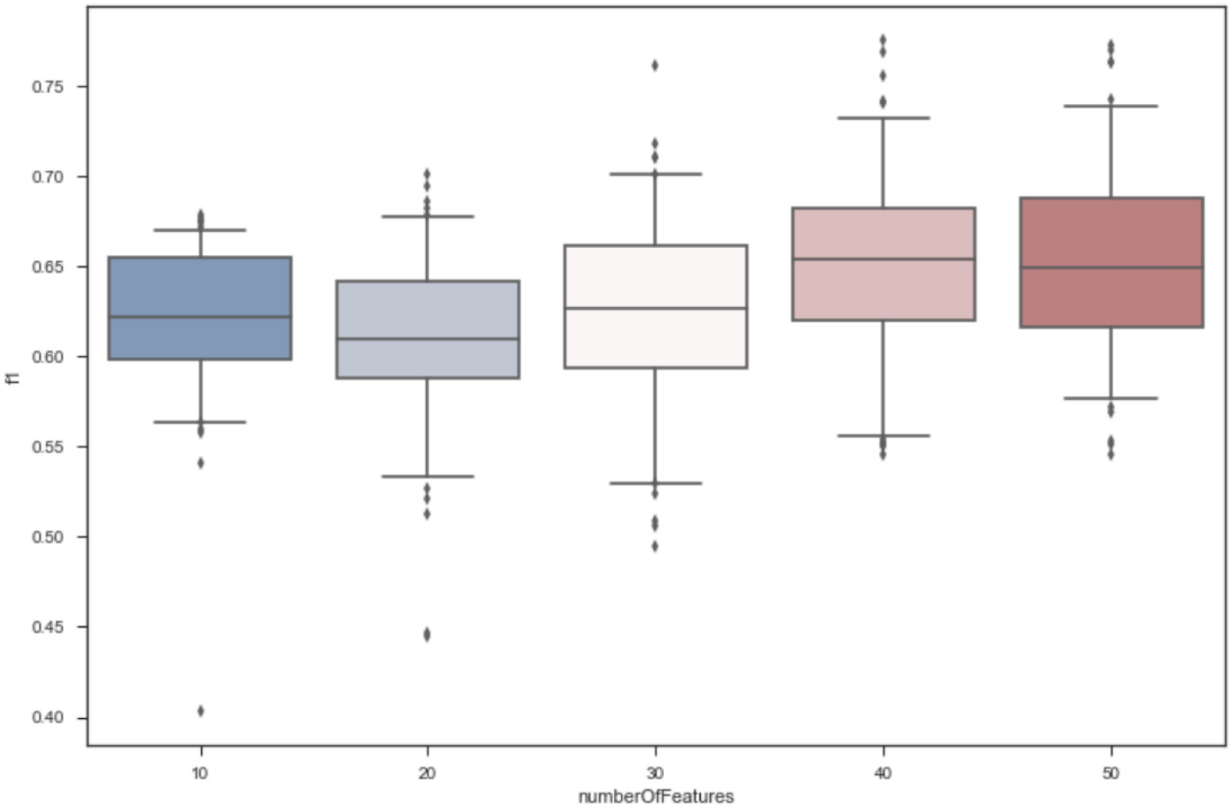


Figure 68: Boxplots of Feature Selection on All Features Using SVC (10-50)

numberOfFeatures	Min f1	Mean f1	Max f1
10	0.403361	0.621542	0.678160
20	0.444385	0.610561	0.701197
30	0.494987	0.624993	0.760955
40	0.545784	0.651534	0.775549
50	0.545599	0.653387	0.772711

Table 35: Statistic Summary of Feature Selection on All Features Using SVC (10-50)

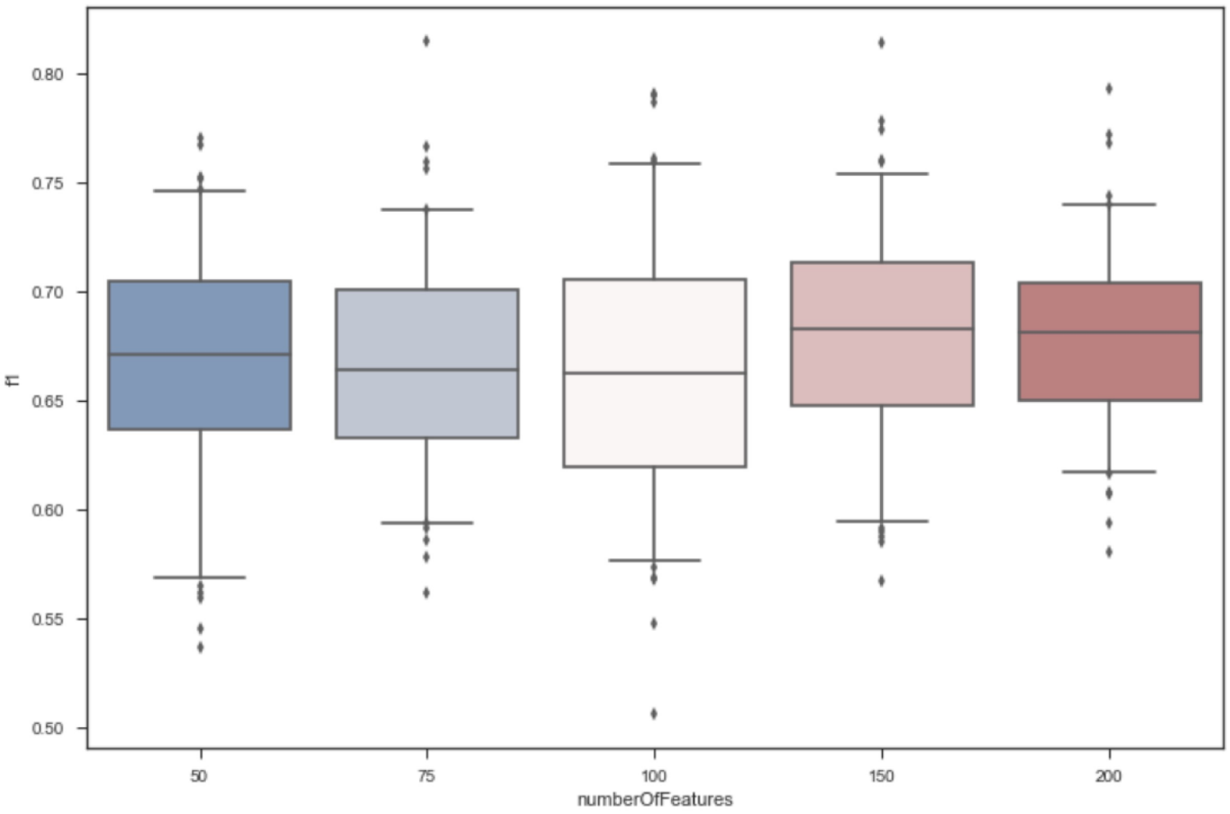


Figure 69: Boxplots of Feature Selection on All Features Using SVC (50-200)

numberOfFeatures	Min f1	Mean f1	Max f1
50	0.536508	0.665614	0.770317
75	0.561616	0.668382	0.815070
100	0.506139	0.664509	0.790700
150	0.567112	0.679038	0.814121
200	0.580220	0.678126	0.792857

Table 36: Statistic Summary of Feature Selection on All Features Using SVC (50-200)

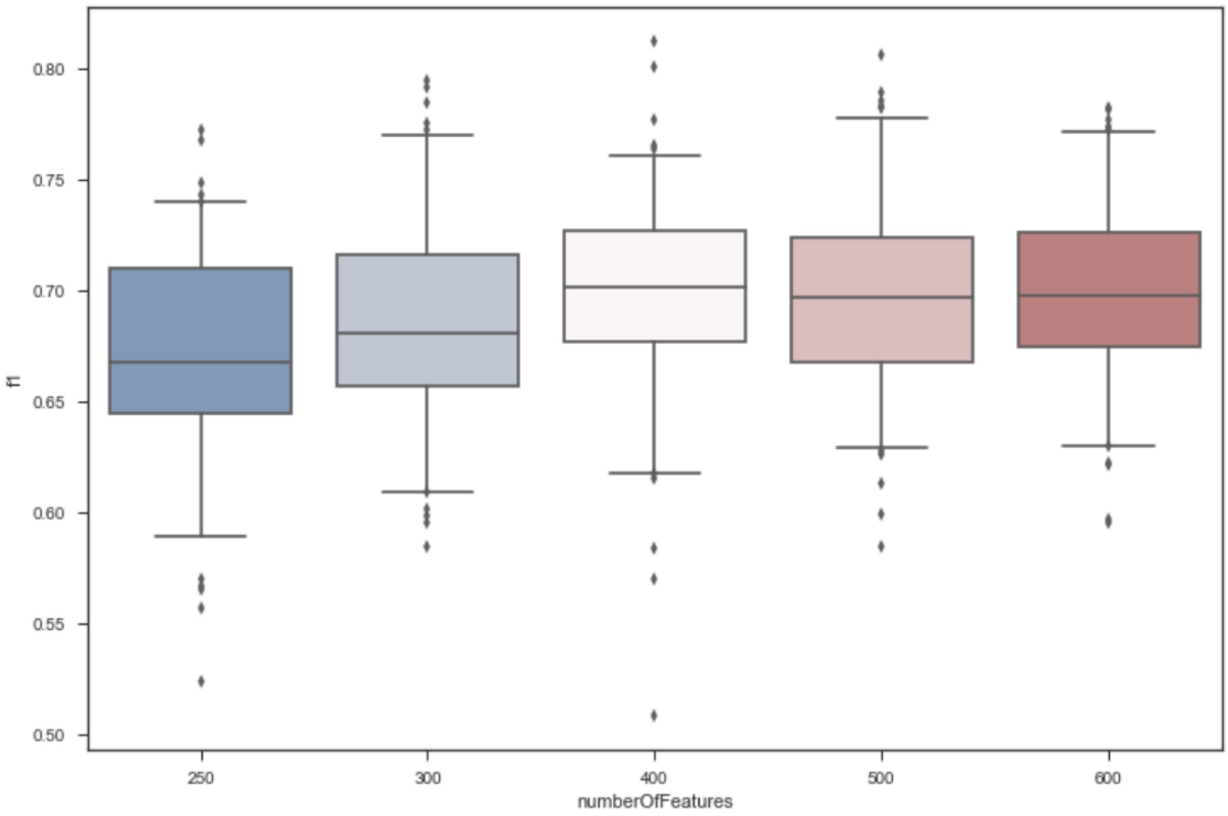


Figure 70: Boxplots of Feature Selection on All Features Using SVC (250-600)

numberOfFeatures	Min f1	Mean f1	Max f1
250	0.523974	0.671849	0.772698
300	0.584900	0.686673	0.795275
400	0.508615	0.699154	0.812880
500	0.584809	0.697467	0.806172
600	0.595379	0.699107	0.782801

Table 37: Statistic Summary of Feature Selection on All Features Using SVC (250-600)

In the second part of this experiment, we ran different feature reduction techniques 100 times each, with the desired number of features set to 600. As shown in Figure 71 and Table 38, recursive feature elimination significantly outperforms any other feature reduction algorithms.

Ranging from 0.79 to 0.96 with an average of 0.88, we managed to obtain the highest F1 score ever. RFE has better performance in larger feature sets since this algorithm recursively eliminates the least important feature until it reaches the desired number of features. So, it is almost guaranteed that, when using a bottom-up approach, the chosen features have better performance than the eliminated ones.

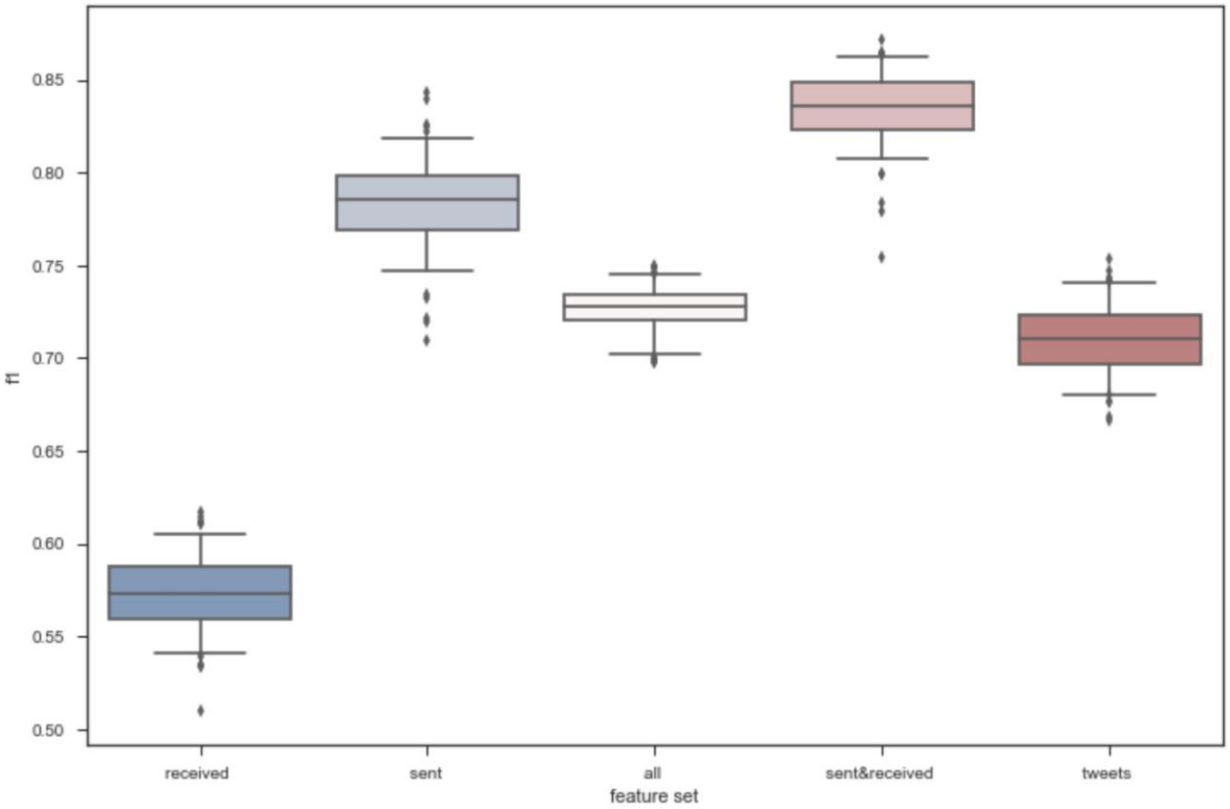


Figure 71: Boxplots of Different Feature Selection Methods on All Features Using SVC

Feature Selection Type	Min f1	Mean f1	Max f1
Chi-Squared	0.573333	0.696144	0.795000
Linear Discriminant Analysis	0.331360	0.474142	0.586181
Principle Component Analysis	0.340963	0.480463	0.601425
Quadratic Discriminant Analysis	0.362491	0.488346	0.603641
Recursive Feature Reduction	0.787943	0.879043	0.958095

Table 38: Statistic Summary of Different Feature Selection Methods on All Features Using SVC

7.5 Conclusion and Discussion

The experiments results showed detecting depression from personal data is very promising. In this project we made an effort to improve the quality of participants on Amazon Mechanical Turk (MTurk), since MTurk users may be biased in terms of representing the general public. There is simply too much uncertainty within this particular population. For future studies, we suggest acquiring experimental data from other sources (this might be costly but it is beneficial to compare across different samples). Our results indicate that text and tweets (written content) can best predict depression. The only source of this kind of data right now is text and Twitter, however for future studies, perhaps more social media data can be included to further improve the potential of text features. Furthermore, our experiments also showed that LIWC as a sentiment analysis library is significantly helpful for feature engineering processes and we have various literature to actually support this tool. Even though it is not free, it is definitely worth using if we want to take text feature machine learning to the next level. For other modalities, which may not be as powerful as text features, still contribute very much in the joint feature set. Future studies could focus more on digging out possibilities from combining various feature sets.

8. Website

We need a way to clearly and concisely display our findings. This is where the idea of a dataset website comes into play. With such a site, we're able to show off several things, including background information on the project, who was involved, and the data itself! This section will go into detail about all components of the website and why they are important. The EMU website is hosted on WPI's domain and can be accessed at https://emu.wpi.edu.

8.1 Pages

Also referred to as the landing page, the "Home" page is the first page that most users will see (unless they have a direct link to any of the other pages). It contains a brief overview of why and how we pursued the notion of predicting depressive behavior. This is important as going straight to the data might be a little overwhelming without any kind of previous knowledge about the project, say from an extensive research paper!

The "About Us" page holds information about everyone involved throughout the entire project. This is including the current team, past teams who inspired this project, advisors, and honorable mentions. Student information includes WPI degree and class year, while faculty and honorable mention information includes position and field of interest.

The main point of the website is for us to cleanly show the results of our study, so it would make sense to have a "Dataset" page dedicated to this. It consists of a citation (referring to this paper!) for those who plan to use information from our dataset to support future research in related fields. This is followed by an explanation of how the data is organized, and then delves depth-first into each of the types of data. It also contains information about our study and results of feature extraction and machine learning analysis. A quick glimpse of this page can be seen below.

Figure 72: EMU Website - Dataset Page Preview

In order to properly give credit to all sources that made this website and dataset possible, there is a dedicated "Publications" page that does just that. Each reference has a citation and link that goes straight to the report. It's important to give credit to those who both inspired us and supplied us with previous research in similar areas and disciplines, which is why publications receive their own page.

8.2 Compatibility

Being that on-the-go technology has become more and more prevalent over the years, it only makes sense to provide functionality supporting compatibility for more than just desktop and laptop computers. That said, the EMU website takes this into consideration and is beautifully displayed on all mobile and tablet devices. See some screenshots below:

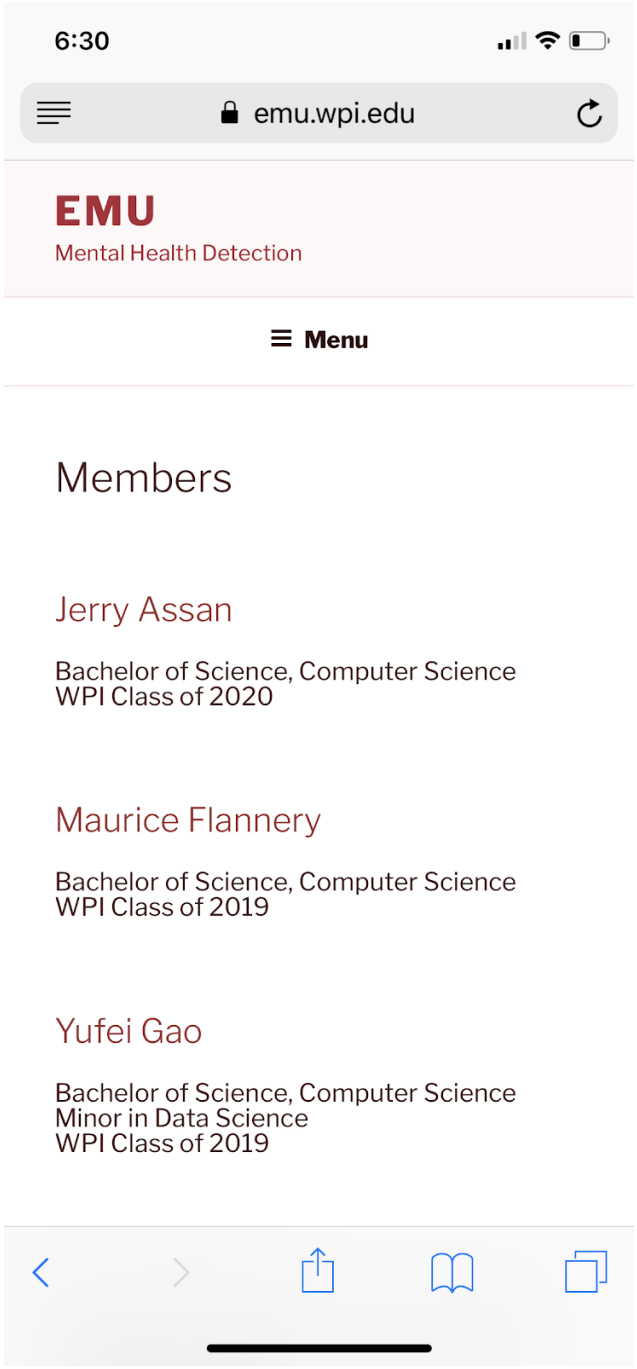


Figure 73: EMU Website - Taken on iPhone X

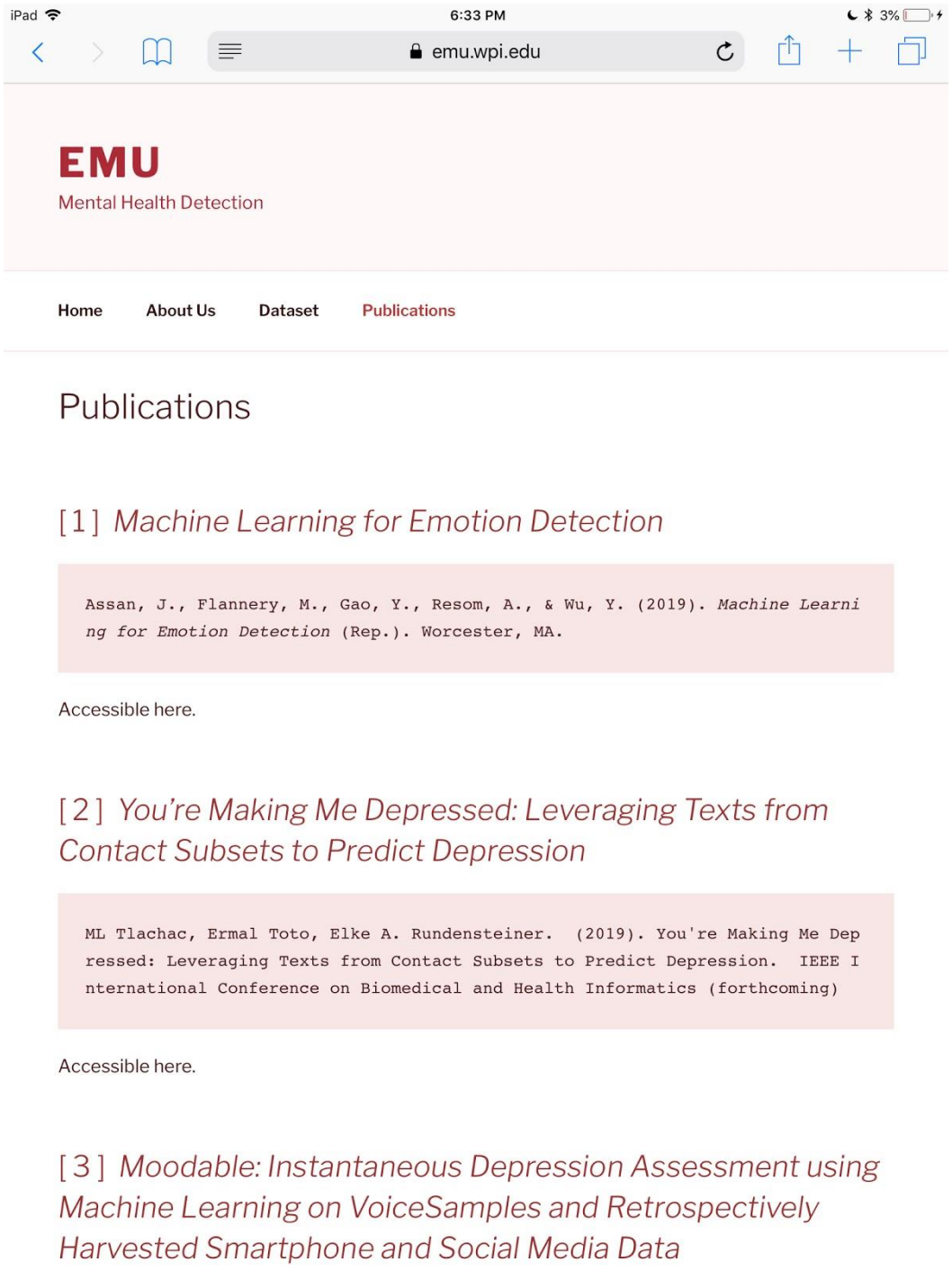


Figure 74: EMU Website - Taken on iPad Pro 10.5-inch

As seen above, mobile and tablet devices have slightly different looks. This is required as screen sizes are different, and so things need to be adjusted accordingly to optimize user

experience, appearance, and performance. The provided compatibility of the EMU site across platforms allows all users to navigate comfortably and efficiently, while being able to receive the exact same information as PC users.

8.3 Other Features

This subsection covers a few smaller features of the site that aren't necessarily worthy of their own subsection: the search functionality and footer buttons.

At the bottom of each page, a search bar is present that allows users to type in some text. Using the entered search terms, the website will return all pages that contain the given words. For instance, searching "audio" will return the "Home" and "Dataset" pages. If there's nothing found, no results will be displayed. This makes it simple for users who are looking for anything in particular, and takes them straight to it.

A piece of functionality implemented solely for convenience, the footer buttons allow quick and easy access to the project members as well as a link to this paper. They have a very simplistic design, and their icons get their points across quite nicely:

Figure 75: EMU Website - Footer Buttons

The email button (left) will open up a new email message, pre-filled with the appropriate email address and subject line to properly address the team and reason for communication. Anything can be entered in the body of the message and all personnel will receive said message. The button linking to the paper (right) will, in a new tab, take users directly to this paper should they want to learn about the more technical aspects of, say, our methodology and reasoning.

8.4 Alterations

Credentials to access the back-end of the website have been passed on to those who will carry on further research based off this project (or to make any other changes they deem necessary). With these privileges, *any* aspect of the website are able to be changed. This is particularly useful if there is new data to share, more team members to add, or any design changes to make (this list is not exhaustive). All websites are different, and our particular implementation is not useful, informative, nor relevant to this project, so further details on how changes themselves are made will be excluded.

9. Conclusion

To wrap up, this final section contains a brief summary of the project and any closing remarks regarding our study, along with improvements that could be made by future members.

9.1 Summary of the Project

Using the EMU mobile Android application in conjunction with various machine learning algorithms, we were successfully able to predict levels of depression on our dataset. The app collected SMS messages along with open- and closed-ended audio prompt responses from real people on Amazon Mechanical Turk (MTurk), which made up part of our dataset. We used several tools and machine learning algorithms to see which combinations performed the best, and were able to obtain results with satisfyingly high accuracies. Of course, we do hope overall accuracies in this field of interest will be improved by those doing similar studies as our results are not perfect. Nonetheless, this is a process that makes it easier for someone to receive an estimate of depression severity, and a type of diagnosis that we hope has potential to reduce the number of people who may have depression but are unaware of it.

9.2 Future Work

The EMU app has more potential than what it has been used for in this project. In addition to text and audio, it can collect the following types of data:

- Call logs
- Contacts
- Calendar information
- GPS (via Google account)
- Twitter
    - Username
    - Tweets
- Instagram
    - Username

- Posts

Some of these kinds of information have been collected, but were not used. We strongly suggest that future studies encourage the collection of either or all of these modalities. Ideally, each participant in the study will provide all of the modalities that can be used, and we recommend that members to come dedicate a good amount of time ensuring that this happens.

We also want to note that although the Linguistic Inquiry and Word Count (LIWC) tool is a paid one, we believe it is worth utilizing as it showed very promising results when compared to other tools. Its outstanding performance allows text analysis to be taken to the next level, and in the end could satisfy our goal of maximizing the number of people who are aware that they show signs of depression.

References

Alpert, M., Pouget, E. R., & Silva, R. R. (2001). Reflections of depression in acoustic measures of the patient's speech. *Journal of affective disorders*, *66*(1), 59-69.

Alghowinem, S., Goecke, R., Wagner, M., Epps, J., Breakspear, M., & Parker, G. (2013). *Detecting depression: A comparison between spontaneous and read speech*. Paper presented at the 7547-7551. doi:10.1109/ICASSP.2013.6639130 Retrieved from https://ieeexplore.ieee.org/document/6639130

Bailey, Katherine. "*Gaussian Processes for Dummies*." Matrix Factorization, Katherine Bailey, 9 Aug. 2016, katbailey.github.io/post/gaussian-processes-for-dummies/.

Ball, D., Dogrucu, A., Isaro, A., & Perucic, A. (2018). *Sensing Depression* (Undergraduate Major Qualifying Project No. E-project-032218-173040). Retrieved from Worcester Polytechnic Institute Electronic Projects Collection: https://web.wpi.edu/Pubs/E-project/Available/E-project-032218-173040/unrestricted/MQP_Final_Report.pdf

Behshad, H., Mohammad, H., Moradi, & Reza, R. (2012). *Classifying depression patients and normal subjects using machine learning techniques and nonlinear features from EEG signal. Computer Methods and Programs in Biomedicine, 109*(3), 339-345. doi:10.1016/j.cmpb.2012.10.008

Bernstein, M. S., Chen, B., & Fast, E. (2016). *Empath: Understanding Topic Signals in Large-Scale Text*.

Brownlee, J. (2016). *Overfitting and underfitting with machine learning algorithms*. Retrieved from https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms

Cai, H., Chen, Y., Han, J., Zhang, X., & Hu, B. (2018). *Study on feature selection methods for depression detection using three-electrode EEG data. Interdisciplinary Sciences: Computational Life Sciences, 10*(3), 558-565. doi:10.1007/s12539-018-0292-5

Canzian L, Musolesi M. *Trajectories of depression*. Proceedings of the 2015 ACM International Joint Conference on pervasive and ubiquitous computing. Sep 7, 2015:1293-1304.

Chen, T., & Guestrin, C. (2016). *XGBoost*. Paper presented at the 785-794. doi:10.1145/2939672.2939785 Retrieved from http://dl.acm.org/citation.cfm?id=2939785

CLiPS Research Center. (2010). *Penn treebank II tag set | CLiPS*. Retrieved from http://www.clips.ua.ac.be/pages/mbsp-tags

Cummins, N., Sethu, V., Epps, J., Schnieder, S., & Krajewski, J. (2015). Analysis of acoustic space variability in speech affected by depression. *Speech Communication, 75*, 27-49. doi:10.1016/j.specom.2015.09.003

de Jong, Nivja & Wempe, Ton. (2009). Praat script to detect syllable nuclei and measure speech rate automatically. Behavior research methods. 41. 385-90. 10.3758/BRM.41.2.385.

Donges, N. (2018, February 22). The Random Forest Algorithm – Towards Data Science. Retrieved from https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

Eyben, F., Weninger, F., Gross F., Schuller, B. (2013). *Recent Developments in openSMILE, the Munich Open-Source Multimedia Feature Extractor*, In Proc. ACM Multimedia (MM), Barcelona, Spain, ACM, ISBN 978-1-4503-2404-5, pp. 835-838, October 2013. doi:10.1145/2502081.2502224

Farhan, A. A., Yue, C., Morillo, R., Ware, S., Lu, J., Bi, J., . . . Wang, B. (2016). *Behavior vs. introspection: Refining prediction of clinical depression via smartphone sensing data*. 2016 IEEE Wireless Health (WH). doi:10.1109/wh.2016.7764553

Gholipour Shahraki, A. (2015). *Emotion mining from text* doi:10.7939/R3C53F63N Retrieved from http://www.dx.doi.org/10.7939/R3C53F63N

Ghose, A. (2017). *Support vector machines tutorial*. Retrieved from https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93

Holczer, B. (2018, October 11). Random Forest Classifier - Machine Learning. Retrieved from http://www.globalsoftwaresupport.com/random-forest-classifier-bagging-machine-learning/

Hsu, C., Chang, C., & Lin, C. (2016). *A practical guide to support vector classification* Retrieved from https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Hutto, C. (2015). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. Retrieved from https://www.researchgate.net/publication/275828927_VADER_A_Parsimonious_Rule-based_Model_for_Sentiment_Analysis_of_Social_Media_Text

K. Vicsi, D. Sztahó and G. Kiss, "Examination of the sensitivity of acoustic-phonetic parameters of speech to depression," *2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*, Kosice, 2012, pp. 511-515. doi:10.1109/CogInfoCom.2012.6422035

Lenzo, K. (n.d.). *The CMU Pronouncing Dictionary*. Retrieved from http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Liu, Z., Kang, H., Feng, L., & Zhang, L. (2017). *Speech pause time: A potential biomarker for depression detection.* 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). doi:10.1109/bibm.2017.8217971

Mayo Clinic Staff (2018). *Depression (major depressive disorder)*. Retrieved from https://www.mayoclinic.org/diseases-conditions/depression/symptoms-causes/syc-20356007

Mitchell, A. J., Vaze, A., & Rao, S. (2009). *Clinical diagnosis of depression in primary care: A meta-analysis* doi://doi.org/10.1016/S0140-6736(09)60879-5

Mitra, V., & Shriberg, E. (2015). *Effects of feature type, learning algorithm and speaking style for depression detection from speech.* Paper presented at the 4774-4778. doi:10.1109/ICASSP.2015.7178877 Retrieved from https://ieeexplore.ieee.org/document/7178877

Morales, M. R. (2018, May). *Multimodal Depression Detection: An Investigation of Features and Fusion Techniques for Automated Systems*. Retrieved from https://academicworks.cuny.edu/cgi/viewcontent.cgi?article=3607&context=gc_etds

Morales, M., & Levitan, R. (2016). *Speech vs. text: A comparative analysis of features for depression detection systems. In Spoken Language Technology Workshop (SLT), 2016 IEEE* (pp. 136–143). IEEE. https://doi.org/10.1109/SLT.2016.7846256

Morales, M., Scherer, S., & Levitan, R. (2017). A cross-modal review of indicators for depression detection systems. In Proceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology–-From Linguistic Signal to Clinical Reality (pp. 1-12).

Mundt, J. C., Snyder, P. J., Cannizzaro, M. S., Chappie, K., & Geralts, D. S. (2007). *Voice acoustic measures of depression severity and treatment response collected via interactive voice response (IVR) technology*. Journal of Neurolinguistics, 20(1), 50–64. http://doi.org/10.1016/j.jneuroling.2006.04.001

Mundt, J. C., Vogel, A. P., Feltner, D. E., & Lenderking, W. R. (2012). *Vocal Acoustic Biomarkers of Depression Severity and Treatment Response.* Biological Psychiatry, 72(7), 580-587. doi:10.1016/j.biopsych.2012.03.015

The National Alliance on Mental Illness (2018). *Types Of Mental Illness*. Retrieved from https://namica.org/resources/mental-illness/types-mental-illness/

The National Institute of Mental Health (2018). *Depression*. Retrieved from https://www.nimh.nih.gov/health/topics/depression/index.shtml

Quatieri, T.F., & Malyska, N. (2012). Vocal-Source Biomarkers for Depression: A Link to Psychomotor Activity. INTERSPEECH.

Rude, S., Gortner, E.-M., & Pennebaker, J. (2004). *Language use of depressed and depression-vulnerable college students*. Cognition and Emotion, 18(8), 1121–1133. https://doi.org/10.1080/02699930441000030

Saeb, S., Lattie, E. G., Kording, K. P., & Mohr, D. C. (2017). *Mobile Phone Detection of Semantic Location and Its Relationship to Depression and Anxiety*. JMIR mHealth and uHealth, 5(8), e112. doi:10.2196/mhealth.7297

SauceCat. "*Boosting Algorithm: AdaBoost – Towards Data Science*." Towards Data Science, Towards Data Science, 29 Apr. 2017, towardsdatascience.com/boosting-algorithm-adaboost-b6737a9ee60c.

Scherer, S., Lucas, G. M., Gratch, J., Rizzo, A. S., & Morency, L. (2016). *Self-reported symptoms of depression and PTSD are associated with reduced vowel space in screening interviews.* IEEE Transactions on Affective Computing, 7(1), 59-73. doi:10.1109/TAFFC.2015.2440264

Sharma, Ishan. "*Introduction to Decision Tree Learning – Heartbeat*." Heartbeat, Heartbeat, 26 Apr. 2018, heartbeat.fritz.ai/introduction-to-decision-tree-learning-cd604f85e236.

sklearn.ensemble.RandomForestClassifier (n.d.). Retrieved October 14, 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Stricker, D. (n.d.). 2D Image Processing: Introduction to object recognition & SVM. Retrieved from https://ags.cs.uni-kl.de/fileadmin/inf_ags/2dip-ss16/2DIP_SS2016_lec09_Recognition_SVM.p

StudentLife Dataset. (n.d.). Retrieved from http://studentlife.cs.dartmouth.edu/dataset.html

Sundar, A. (2018). *An end-to-end guide to understand the math behind XGBoost*. Retrieved from https://medium.com/analytics-vidhya/an-end-to-end-guide-to-understand-the-math-behind-xgboost-72c07acb4afb

Swaminathan, Saishruthi, and Saishruthi Swaminathan. "*Logistic Regression - Detailed Overview – Towards Data Science*." Towards Data Science, Towards Data Science, 15 Mar. 2018, towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc.

Synced. (2017). *Tree boosting with XGBoost — why does XGBoost win "Every" machine learning competition?* Retrieved from https://medium.com/syncedreview/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition-ca8034c0b283

Tausczik, Y., & Pennebaker, J. (2009). The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods. *Journal of Language and Social Psychology*, *29*(1), 24–54. https://doi.org/10.1177/0261927X09351676

Vaizman, Y., & Ellis, K. (2016). The ExtraSensory Dataset. Retrieved from http://extrasensory.ucsd.edu

Wolters, M. K., Ferrini, L., Farrow, E., Tatar, A. S., & Burton, C. D. (2015). TRACKING DEPRESSED MOOD USING SPEECH PAUSE PATTERNS. *In ICPhS*. Retrieved from https://pdfs.semanticscholar.org/2ba7/1b43d2e50878a0ef8ef5348a0bfc3cf2c3b1.pdf

World Health Organization (2018). *Depression*. Retrieved from http://www.who.int/news-room/fact-sheets/detail/depression

Zimmermann, J., Brockmeyer, T., Hunn, M., Schauenburg, H., & Wolf, M. (2017). *First-person pronoun use in spoken language as a predictor of future depressive symptoms: Preliminary evidence from a clinical sample of depressed patients. Clinical Psychology & Psychotherapy, 24*(2), 384-391. doi:10.1002/cpp.2006

Josh. (2015). Everything you need to know about artificial neural networks. Retrieved from https://medium.com/technology-invention-and-more/everything-you-need-to-know-about-artificial-neural-networks-57fac18245a1

Appendices

Appendix A - Table of Division of Labor

Chapter	Section	Authors	Editors
Introduction	-	Yufei	Grammar & Sentence Structure: Maurice Format & Citation: Yufei
Background	Summary of the Previous MQP	Yuxin, Yufei, Jerry	
	Background on Depression	Yuxin, Jerry	
	Detecting Depression from Text	Maurice, Yuxin	
	Detecting Depression from Voice	Yufei, Adonay	
	Detecting Depression from GPS Data	Yufei	
Feature Engineering Methodology	Text Features	Maurice, Yuxin	
	Audio Features	Adonay	
	GPS Features	Adonay, Yufei	
Machine Learning Methodology	K-Nearest Neighbors	Yuxin, Yufei	
	Support Vector Machine	Jerry, Maurice	
	Random Forest	Adonay	
	XGBoost	Yufei, Jerry	

	AdaBoost	Yuxin	
	Decision Tree	Yuxin	
	Logistic Regression	Yuxin	
	Artificial Neural Networks	Yuxin	
	Voting	Yuxin	
	Gaussian Process	Yuxin	
	Grid Search	Yuxin	
EMU Pipeline	All Subsections	Jerry, Yufei	
Experimental Methodology	Amazon Mechanical Turk	Yuxin	
	Mobile Application	Anoday	
	EMU Server	Maurice	
Experimental Evaluation and Analysis	All Subsections	Jerry, Yufei	
Website	All Subsections	Maurice	
Conclusion	All Subsections	Maurice	

Appendix B - Table of Literature Papers and Features

#	Features	Title of Paper	Link to Paper	Type
1	Reduced vowel space	Self-reported symptoms of depression and PTSD are associated with reduced vowel space in screening interviews	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7117386	Audio
2	Pause time Speaking rates	Voice acoustic measures of depression severity and treatment response collected via interactive voice response (IVR) technology	https://www.sciencedirect.com/science/article/pii/S0911604406000303	Audio
3	LIWC, NRC	Emotion mining from text	http://www.dx.doi.org/10.7939/R3C53F63N	Text
4	Number of words, word frequency, the number of syllables, the word case (capitalization), and punctuation usage	Speech vs. text: A comparative analysis of features for depression detection systems	https://doi.org/10.1109/SLT.2016.7846256	Text
5	Word case, punctuation	A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text	https://www.researchgate.net/publication/275828927_VADER_A_Parsimonious_Rule-based_Model_for_Sentiment_Analysis_of_Social_Media_Text	Text
6	Self-references	Language use of depressed and depression-vulnerable college students	https://doi.org/10.1080/02699930441000030	Text
7	Social words	The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods	https://doi.org/10.1177/0261927X09351676	Text

8	Switching pause Vocal fundamental frequency (F0)	Detecting Depression Severity from Vocal Prosody	https://www.pitt.edu/~jeffcohn/biblio/tac_prosody.pdf	Audio
9	To be determined	A Depression Detection Model Based on Sentiment Analysis in Micro-blog Social Network	https://link.springer.com/content/pdf/10.1007%2F978-3-642-40319-4_18.pdf	Social Media
10	Number of First-Person Pronoun	First-person Pronoun Use in Spoken Language as a Predictor of Future Depressive Symptoms: Preliminary Evidence from a Clinical Sample of Depressed Patients.	https://onlinelibrary.wiley.com/doi/epdf/10.1002/cpp.2006	Text
11	Number of First-Person Pronoun	The Psychology of Word Use in Depression Forums in English and in Spanish: Testing Two Text Analytic Approaches	http://www.aaai.org/Papers/ICWSM/2008/ICWSM08-020.pdf	Text
12	Speaking style (read versus spontaneous) on depression prediction	Effects of feature type, learning algorithm and speaking style for depression detection from speech	https://ieeexplore.ieee.org/document/7178877	Audio
13	Number of First-Person Pronoun	Me, myself, and I: self-referent word use as an indicator of self-focused attention in relation to depression and anxiety	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4598574/pdf/fpsyg-06-01564.pdf	Text
14	Features derived from the formant and power	Acoustical Properties of Speech as Indicators of Depression and Suicidal Risk	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=846676	Audio

	spectral density measurements			
15	Degree of breathiness, jitter and shimmer	Speech Features for Depression Detection	https://www.isca-speech.org/archive/Interspeech_2016/pdfs/1566.PDF	Audio
16	Doc2Vec of the Python Gensim library COVAREP features	Detecting Depression with Audio/Text Sequence Modeling of Interviews	https://groups.csail.mit.edu/sls/publications/2018/Alhanai_Interspeech-2018.pdf	Audio & Text
17	Acoustic Space Variability in Speech	Analysis of Acoustic Space Variability in Speech Affected by Depression	https://www.researchgate.net/publication/281719239_Analysis_of_Acoustic_Space_Variability_in_Speech_Affected_by_Depression	Audio
18	High jitter Low shimmer	Characterising Depressed Speech for Classification	https://pdfs.semanticscholar.org/63d5/28976b69e122b6e494cb4c50dd83b2dac3d4.pdf	Audio
19	Speaking style, jitter, shimmer, energy and loudness feature groups	DETECTING DEPRESSION:A COMPARISON BETWEEN SPONTANEOUS AND READ SPEECH	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6639130	Audio
20	Location variance, Time Spent in Moving, Total distance, Number of Unique Locations	Behavior vs. introspection: refining prediction of clinical depression via smartphone sensing data	https://ieeexplore.ieee.org/document/7764553	GPS
21	Location variance	Mobile Phone Detection of Semantic Location and Its Relationship to Depression	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5571235/	GPS

		and Anxiety		
22	Total distance covered and the number of different places visited	Trajectories of Depression: Unobtrusive Monitoring of Depressive States by means of Smartphone Mobility Traces Analysis	https://userpages.umbc.edu/~nroy/courses/shhasp18/papers/Depressive_State%20Monitoring_Ubicomp15.pdf	GPS

Appendix C – Pivot Tables of Grid Search Results For Text Features

SVC:

Count of C	Column Labels				
Row Labels	'gamma': 0.001	'gamma': 0.01	'gamma': 1	'gamma': 10	Grand Total
{'C': 0.001				**23**	**23**
'kernel': 'rbf'}				23	23
{'C': 1	**3**		**3**	**69**	**75**
'kernel': 'linear'}	3				3
'kernel': 'rbf'}			3	69	72
{'C': 10		**1**			**1**
'kernel': 'rbf'}		1			1
{'C': 100		**1**			**1**
'kernel': 'rbf'}		1			1
Grand Total	**3**	**2**	**3**	**92**	**100**

kNN:

Sum of count	Column Labels					
Row Labels	'n_neighbors': 3}	'n_neighbors': 5}	'n_neighbors': 7}	'n_neighbors': 9}	(blank)	Grand Total
{'leaf_size': 1	2	4	14	62		82
{'leaf_size': 10		1		3		4
{'leaf_size': 20		1		1		2
{'leaf_size': 3			3	2		5
{'leaf_size': 5			2	5		7
(blank)						
Grand Total	**2**	**6**	**19**	**73**		**100**

LR:

Count of solver	Column Labels	
	'multi_class': 'ovr'	Grand Total
Row Labels	'random_state': 0	
{'C': 0.1	74	74
{'C': 1	25	25
{'C': 10	1	1
Grand Total	**100**	**100**

GP:

Sum of count	Column Labels	
Row Labels	{'max_iter_predict': 1	Grand Total
'n_jobs': 1}	100	100
Grand Total	**100**	**100**

RF:

Count of n_estimators	Column Labels			
Row Labels	{'max_depth': 2	{'max_depth': 3	{'max_depth': 4	Grand Total
'min_samples_leaf': 1	**9**	**22**	**12**	**43**
'min_samples_split': 2	5	3	6	14
'min_samples_split': 3	1	6	4	11
'min_samples_split': 5	3	13	2	18
'min_samples_leaf': 2	**10**	**21**	**13**	**44**
'min_samples_split': 2	7	5	3	15
'min_samples_split': 3	1	7	6	14
'min_samples_split': 5	2	9	4	15
'min_samples_leaf': 3	**2**	**7**	**4**	**13**
'min_samples_split': 2	1	2	2	5
'min_samples_split': 3	1	4		5
'min_samples_split': 5		1	2	3
Grand Total	**21**	**50**	**29**	**100**

NN:

Sum of count	Column Labels			
Row Labels	'max_iter': 100	'max_iter': 200	'max_iter': 50	Grand Total
{'learning_rate': 'adaptive'	**19**	**10**	**17**	**46**
'solver': 'sgd'}	**19**	**10**	**17**	**46**
'learning_rate_init': 0.001	7	1		8
'learning_rate_init': 1	4	2	6	12
'learning_rate_init': 2	8	7	11	26
{'learning_rate': 'constant'	**16**	**19**	**16**	**51**
'solver': 'sgd'}	**16**	**19**	**16**	**51**
'learning_rate_init': 0.001	8	7	8	23
'learning_rate_init': 1	4	6	5	15
'learning_rate_init': 2	4	6	3	13
{'learning_rate': 'invscaling'	**1**	**1**	**1**	**3**
'solver': 'sgd'}	**1**	**1**	**1**	**3**
'learning_rate_init': 1	1	1	1	3
Grand Total	**36**	**30**	**34**	**100**

XGB:

Sum of count	Column Labels				
Row Labels	'gamma': 0	'gamma': 0.0015	'gamma': 0.1	'gamma': 5	Grand Total
{'colsample_bytree': 0.6	**13**	**1**	**12**	**35**	**61**
'max_depth': 2	**2**		**2**	**9**	**13**
'min_child_weight': 1			2	2	4
'subsample': 0.6}				2	2
'subsample': 0.7}			1		1
'subsample': 0.8}			1		1
'min_child_weight': 2				6	6
'subsample': 0.6}				3	3
'subsample': 0.8}				3	3
'min_child_weight': 5	2			1	3
'subsample': 0.6}				1	1
'subsample': 0.8}	2				2
'max_depth': 3	**3**	**1**	**7**	**11**	**22**
'min_child_weight': 1	2		3	4	9
'subsample': 0.6}	2		1	1	4
'subsample': 0.7}			1	2	3
'subsample': 0.8}			1	1	2
'min_child_weight': 2	1	1	3	4	9
'subsample': 0.6}	1		2		3
'subsample': 0.7}			1	2	3
'subsample': 0.8}		1		2	3
'min_child_weight': 5			1	3	4
'subsample': 0.7}			1		1
'subsample': 0.8}				3	3
'max_depth': 4	**8**		**3**	**15**	**26**
'min_child_weight': 1	5		2	9	16
'subsample': 0.6}	3			1	4
'subsample': 0.7}	1			6	7
'subsample': 0.8}	1		2	2	5
'min_child_weight': 2	3		1	3	7
'subsample': 0.6}				1	1
'subsample': 0.7}				1	1
'subsample': 0.8}	3		1	1	5
'min_child_weight': 5				3	3
'subsample': 0.7}				2	2
'subsample': 0.8}				1	1
{'colsample_bytree': 0.8	**8**	**1**	**6**	**24**	**39**
'max_depth': 2				**4**	**4**
'min_child_weight': 1				2	2
'subsample': 0.7}				1	1
'subsample': 0.8}				1	1
'min_child_weight': 2				1	1
'subsample': 0.8}				1	1
'min_child_weight': 5				1	1
'subsample': 0.8}				1	1
'max_depth': 3	**4**		**2**	**6**	**12**
'min_child_weight': 1			1	3	4
'subsample': 0.6}			1		1
'subsample': 0.7}				1	1
'subsample': 0.8}				2	2
'min_child_weight': 2	4			2	6
'subsample': 0.6}	1				1
'subsample': 0.7}	3				3
'subsample': 0.8}				2	2
'min_child_weight': 5			1	1	2
'subsample': 0.7}			1		1
'subsample': 0.8}				1	1
'max_depth': 4	**4**	**1**	**4**	**14**	**23**
'min_child_weight': 1	1	1	2	5	9
'subsample': 0.6}		1	2	4	7
'subsample': 0.7}	1				1
'subsample': 0.8}				1	1
'min_child_weight': 2	3		2	5	10
'subsample': 0.6}	1		1	1	3
'subsample': 0.7}	1		1	1	3
'subsample': 0.8}	1			3	4
'min_child_weight': 5				4	4
'subsample': 0.7}				1	1
'subsample': 0.8}				3	3
Grand Total	**21**	**2**	**18**	**59**	**100**

Appendix D – Pivot Tables of Grid Search Results For Audio Features

SVC:

Sum of count	Column Labels				
Row Labels	'gamma': 0.001	'gamma': 0.1	'gamma': 1	'gamma': 10	Grand Total
'kernel': 'linear'}	**1**				**1**
{'C': 1	1				1
'kernel': 'rbf'}		**77**	**17**	**5**	**99**
{'C': 0.001		77	7	4	88
{'C': 0.01			1		1
{'C': 0.1				1	1
{'C': 1			3		3
{'C': 10			6		6
Grand Total	**1**	**77**	**17**	**5**	**100**

RF:

Sum of count	Column Labels			
Row Labels	'min_samples_leaf': 1	'min_samples_leaf': 2	'min_samples_leaf': 3	Grand Total
'n_estimators': 500}	**32**	**36**	**32**	**100**
'min_samples_split': 2	**10**	**14**	**8**	**32**
{'max_depth': 2	3	5	2	10
{'max_depth': 3	2	4	4	10
{'max_depth': 4	5	5	2	12
'min_samples_split': 3	**10**	**10**	**11**	**31**
{'max_depth': 2	2	4	6	12
{'max_depth': 3	3	5	3	11
{'max_depth': 4	5	1	2	8
'min_samples_split': 5	**12**	**12**	**13**	**37**
{'max_depth': 2	2	4	7	13
{'max_depth': 3	6	4	2	12
{'max_depth': 4	4	4	4	12
Grand Total	**32**	**36**	**32**	**100**

kNN:

Sum of count	Column Labels				
Row Labels	'n_neighbors': 1}	'n_neighbors': 3}	'n_neighbors': 5}	'n_neighbors': 9}	Grand Total
{'leaf_size': 1	66	28	1	5	100
Grand Total	**66**	**28**	**1**	**5**	**100**

LR:

Count of random_state	Column Labels	
Row Labels	'random_state': 0	Grand Total
'solver': 'lbfgs'}	**100**	**100**
'multi_class': 'ovr'	**100**	**100**
{'C': 0.01	10	10
{'C': 0.1	7	7
{'C': 1	17	17
{'C': 10	25	25
{'C': 100	41	41
Grand Total	**100**	**100**

GP:

Sum of count	Column Labels		
Row Labels	{'max_iter_predict': 1	{'max_iter_predict': 5	Grand Total
'n_jobs': 1}	99	1	100
Grand Total	**99**	**1**	**100**

NN:

Sum of count	Column Labels			
Row Labels	'max_iter': 100	'max_iter': 200	'max_iter': 50	Grand Total
'solver': 'sgd'}	**28**	**35**	**37**	**100**
{'learning_rate': 'adaptive'	**10**	**14**	**13**	**37**
'learning_rate_init': 0.001	7	5	5	17
'learning_rate_init': 1	3	6	6	15
'learning_rate_init': 2		3	2	5
{'learning_rate': 'constant'	**10**	**7**	**16**	**33**
'learning_rate_init': 0.001	3	3	5	11
'learning_rate_init': 1	6	4	5	15
'learning_rate_init': 2	1		6	7
{'learning_rate': 'invscaling'	**8**	**14**	**8**	**30**
'learning_rate_init': 0.001	1	5	4	10
'learning_rate_init': 1	3	7	1	11
'learning_rate_init': 2	4	2	3	9
Grand Total	**28**	**35**	**37**	**100**

XGB:

Sum of count	Column Labels				
Row Labels	'gamma': 0	'gamma': 0.0015	'gamma': 0.1	'gamma': 5	Grand Total
{'colsample_bytree': 0.6	**19**	**2**	**5**	**24**	**50**
'min_child_weight': 1	**1**	**1**	**3**	**6**	**11**
'max_depth': 2			1	1	2
'subsample': 0.6}			1	1	2
'max_depth': 3			1	3	4
'subsample': 0.6}			1		1
'subsample': 0.7}				2	2
'subsample': 0.8}				1	1
'max_depth': 4	1	1	1	2	5
'subsample': 0.6}			1		1
'subsample': 0.7}				2	2
'subsample': 0.8}	1	1			2
'min_child_weight': 2	**3**	**1**	**2**	**10**	**16**
'max_depth': 2				4	4
'subsample': 0.6}				2	2
'subsample': 0.7}				2	2
'max_depth': 3	3	1	1	4	9
'subsample': 0.6}	1			1	2
'subsample': 0.7}	1	1		2	4
'subsample': 0.8}	1		1	1	3
'max_depth': 4			1	2	3
'subsample': 0.6}				1	1
'subsample': 0.7}				1	1
'subsample': 0.8}			1		1
'min_child_weight': 5	**15**			**8**	**23**
'max_depth': 2	7			4	11
'subsample': 0.6}	2			2	4
'subsample': 0.7}				1	1
'subsample': 0.8}	5			1	6
'max_depth': 3	4			3	7
'subsample': 0.6}	3			1	4
'subsample': 0.7}	1			1	2
'subsample': 0.8}				1	1
'max_depth': 4	4			1	5
'subsample': 0.6}				1	1
'subsample': 0.7}	1				1
'subsample': 0.8}	3				3
{'colsample_bytree': 0.8	**22**	**2**	**6**	**20**	**50**
'min_child_weight': 1	**4**	**1**	**2**	**3**	**10**
'max_depth': 2	2			2	4
'subsample': 0.7}				2	2
'subsample': 0.8}	2				2
'max_depth': 3	1			1	2
'subsample': 0.6}	1				1
'subsample': 0.7}				1	1
'max_depth': 4	1	1	2		4
'subsample': 0.6}	1				1
'subsample': 0.7}		1			1
'subsample': 0.8}			2		2
'min_child_weight': 2	**4**	**1**	**4**	**4**	**13**
'max_depth': 2	1		2	3	6
'subsample': 0.6}	1			2	3
'subsample': 0.7}			1		1
'subsample': 0.8}			1	1	2
'max_depth': 3	1		1		2
'subsample': 0.6}			1		1
'subsample': 0.8}	1				1
'max_depth': 4	2	1	1	1	5
'subsample': 0.6}	1			1	2
'subsample': 0.7}		1			1
'subsample': 0.8}	1		1		2
'min_child_weight': 5	**14**			**13**	**27**
'max_depth': 2	7			6	13
'subsample': 0.6}	3			2	5
'subsample': 0.7}	2			2	4
'subsample': 0.8}	2			2	4
'max_depth': 3	2			4	6
'subsample': 0.7}	1			3	4
'subsample': 0.8}	1			1	2
'max_depth': 4	5			3	8
'subsample': 0.6}	2				2
'subsample': 0.7}	1			2	3
'subsample': 0.8}	2			1	3
Grand Total	**41**	**4**	**11**	**44**	**100**

Appendix E – Pivot Tables of Grid Search Results For GPS Features

SVC:

Sum of counts	Column Labels		
Row Labels	'kernel': 'linear'}	'kernel': 'rbf'}	Grand Total
{'C': 0.001	**37**	**41**	**78**
'gamma': 0.001	37	6	43
'gamma': 0.01		35	35
{'C': 0.01	**11**		**11**
'gamma': 0.001	11		11
{'C': 1		**5**	**5**
'gamma': 0.01		5	5
{'C': 10		**6**	**6**
'gamma': 0.001		6	6
Grand Total	**48**	**52**	**100**

RF:

Sum of count	Column Labels			
Row Labels	'min_samples_leaf': 1	'min_samples_leaf': 2	'min_samples_leaf': 3	Grand Total
'n_estimators': 500}	**23**	**34**	**43**	**100**
{'max_depth': 2	**5**	**7**	**11**	**23**
'min_samples_split': 2		1	4	5
'min_samples_split': 3	1	3	6	10
'min_samples_split': 5	4	3	1	8
{'max_depth': 3	**4**	**9**	**14**	**27**
'min_samples_split': 2		3	11	14
'min_samples_split': 3	3	3	1	7
'min_samples_split': 5	1	3	2	6
{'max_depth': 4	**14**	**18**	**18**	**50**
'min_samples_split': 2	2	6	6	14
'min_samples_split': 3	8	5	7	20
'min_samples_split': 5	4	7	5	16
Grand Total	**23**	**34**	**43**	**100**

kNN:

Sum of count	Column Labels	
Row Labels	{'leaf_size': 1	Grand Total
'n_neighbors': 1}	7	7
'n_neighbors': 3}	3	3
'n_neighbors': 5}	2	2
'n_neighbors': 7}	24	24
'n_neighbors': 9}	64	64
Grand Total	**100**	**100**

LR:

Sum of count	Column Labels	
	'random_state': 0	
Row Labels	'solver': 'lbfgs'}	
{'C': 0.01		100
'multi_class': 'ovr'		100
Grand Total		**100**

GP:

Sum of count	Column Labels	
Row Labels	'n_jobs': 1}	Grand Total
{'max_iter_predict': 1	80	80
{'max_iter_predict': 5	20	20
Grand Total	**100**	**100**

NN:

Sum of count	Column Labels			
Row Labels	'learning_rate_init': 0.001	'learning_rate_init': 1	'learning_rate_init': 2	Grand Total
'solver': 'sgd'}	**55**	**8**	**37**	**100**
{'learning_rate': 'adaptive'	**20**	**1**		**21**
'max_iter': 100	10			10
'max_iter': 200	2			2
'max_iter': 50	8	1		9
{'learning_rate': 'constant'	**22**	**4**	**25**	**51**
'max_iter': 100	7		8	15
'max_iter': 200	3		9	12
'max_iter': 50	12	4	8	24
{'learning_rate': 'invscaling'	**13**	**3**	**12**	**28**
'max_iter': 100	5		4	9
'max_iter': 200	2		5	7
'max_iter': 50	6	3	3	12
Grand Total	**55**	**8**	**37**	**100**

XGB:

Sum of count	Column Labels			
Row Labels	'min_child_weight': 1	'min_child_weight': 2	'min_child_weight': 5	Grand Total
{'colsample_bytree': 0.6	**21**	**10**	**6**	**37**
'gamma': 0	**8**	**4**	**2**	**14**
'subsample': 0.6}	1		1	2
'max_depth': 2			1	1
'max_depth': 4	1			1
'subsample': 0.7}	4	1		5
'max_depth': 3	2	1		3
'max_depth': 4	2			2
'subsample': 0.8}	3	3	1	7
'max_depth': 2			1	1
'max_depth': 3	1			1
'max_depth': 4	2	3		5
'gamma': 0.0015	**1**			**1**
'subsample': 0.6}	1			1
'max_depth': 4	1			1
'gamma': 0.1	**7**	**3**	**1**	**11**
'subsample': 0.6}	2	1		3
'max_depth': 2		1		1
'max_depth': 4	2			2
'subsample': 0.7}	2	1	1	4
'max_depth': 2		1	1	2
'max_depth': 4	2			2
'subsample': 0.8}	3	1		4
'max_depth': 2	1			1
'max_depth': 3	1	1		2
'max_depth': 4	1			1
'gamma': 5	**5**	**3**	**3**	**11**
'subsample': 0.6}	3	1	1	5
'max_depth': 2	1		1	2
'max_depth': 3	1	1		2
'max_depth': 4	1			1
'subsample': 0.7}	2	2		4
'max_depth': 3	1			1
'max_depth': 4	1	2		3
'subsample': 0.8}			2	2
'max_depth': 2			2	2
{'colsample_bytree': 0.8	**32**	**17**	**14**	**63**
'gamma': 0	**11**	**4**	**5**	**20**
'subsample': 0.6}	2	2	3	7
'max_depth': 2	1		3	4
'max_depth': 4	1	2		3
'subsample': 0.7}	3	1	1	5
'max_depth': 2			1	1
'max_depth': 4	3	1		4
'subsample': 0.8}	6	1	1	8
'max_depth': 2	2		1	3
'max_depth': 3	1			1
'max_depth': 4	3	1		4
'gamma': 0.0015	**3**			**3**
'subsample': 0.6}	2			2
'max_depth': 4	2			2
'subsample': 0.8}	1			1
'max_depth': 4	1			1
'gamma': 0.1	**7**	**6**		**13**
'subsample': 0.6}	2	2		4
'max_depth': 3	1	1		2
'max_depth': 4	1	1		2
'subsample': 0.7}	3	1		4
'max_depth': 2	1			1
'max_depth': 3		1		1
'max_depth': 4	2			2
'subsample': 0.8}	2	3		5
'max_depth': 3	2	1		3
'max_depth': 4		2		2
'gamma': 5	**11**	**7**	**9**	**27**
'subsample': 0.6}	3	3	4	10
'max_depth': 2		2	4	6
'max_depth': 3	3			3
'max_depth': 4		1		1
'subsample': 0.7}	4	3		7
'max_depth': 2	2			2
'max_depth': 3	2	2		4
'max_depth': 4		1		1
'subsample': 0.8}	4	1	5	10
'max_depth': 2	1		5	6
'max_depth': 3	1			1
'max_depth': 4	2	1		3
Grand Total	**53**	**27**	**20**	**100**

Appendix F – Code

1 – Pratt script that was used to generate pause time features

```
To TextGrid (silences)... threshold3 minpause 0.1 silent sounding
textgridid = selected("TextGrid")
silencetierid = Extract tier... 1
silencetableid = Down to TableOfReal... sounding
nsounding = Get number of rows
nsounds = 'nsounding'
npauses = 0
speakingtot = 0
pausetot = 0
beginsound = 0
endsound = 0
beginpause = 0
endpause = 0
for ipause from 1 to nsounds
   if 'endsound' <> 0
            beginpause = 'endsound'
   endif
   beginsound = Get value... 'ipause' 1
       if 'beginpause' <> 0 and 'beginsound' <> 0
           endpause = 'beginsound'
               npauses = npauses + 1
               pausedur = 'endpause' - 'beginpause'
               pausetot = 'pausedur' + 'pausetot'
   endif
   endsound = Get value... 'ipause' 2
   speakingdur = 'endsound' - 'beginsound'
   speakingtot = 'speakingdur' + 'speakingtot'
endfor
```

2 – Pratt script that was used to generate signal features

```
form Test command line cals
    sentence Source_path unknown
    sentence Destination_path reports
    sentence File_name 001
    sentence Source_extension .wav
endform

sound = Read from file: source_path$ + "\" + file_name$ + source_extension$
selectObject: sound
# Initial variables, just to make the parameters clearer
pitch_min = 60
pitch_max = 600
time_step = 0.3
silence_threshold = -25
min_pause = 0.1
min_voiced = 0.1
tier = 1

# Detect silences
sound = selected("Sound")
textgrid = To TextGrid (silences): pitch_min, time_step,
  ... silence_threshold, min_pause, min_voiced, "silent", "sounding"
# The TextGrid is automatically selected
total_intervals = Get number of intervals: tier

# Make the remaining objects
selectObject: sound
pitch = To Pitch: 0, pitch_min, pitch_max
selectObject: sound
pulses = To PointProcess (periodic, cc): pitch_min, pitch_max

report_table_columns$ = "median_pitch mean_pitch sd_pitch min_pitch max_pitch
report_table = Create Table with column names: "report_table", 0, report_tabl

# Find beginning and end of sounding intervals
for i to total_intervals
  selectObject: textgrid
  label$ = Get label of interval: tier, i
  if label$ == "sounding"
    start = Get start point: tier, i
    end   = Get end point: tier, i
```