CS7641-Project-Page

Intro:

We want to perform sentiment analysis on tweets. Although the dataset we use is a general twitter dataset, we can apply a similar type of classification on tweets under specific categories to determine the overall sentiment towards a certain topic.

Literature Review:

Sentiment Analysis of Tweets Using Machine Learning Approach

- Inspired by this paper that uses TF-IDF vectorization as features in classification [3].

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

- This paper introduces BERT vector embedding which uses a transformer architecture to capture semantic realtionships between text [1]. We will incorporate this into our classification to create features for the text.

Dataset:

https://www.kaggle.com/datasets/kazanova/sentiment140

This dataset contains 1.6 million tweets annotated with the polarity of the tweet being 0(negative) or 1(positive), date, user, and flag. We use a random sample of 30,000 of the 1.6 million tweets for efficiency.

Problem:

This project is to conduct sentiment analysis on tweets to classify the overall sentiment of a tweet regarding specific topics as either positive or negative. A “0” signifies a negative view of the topic, whereas a “1” represents the opposite.

Motivation:

People naturally convey emotions through text. Text is difficult to classify due to its semantic nature, however with the large amounts of textual data available, we need a way to quantify emotions and opinions expressed through text, to better understand public sentiment regarding a certain topic.

Our motivation is to accurately classify tweets as “positive” or “negative” sentiment to help aid decision making processes in various domains. Sentiment analysis has various applications, from gauging customer satisfaction to easily addressing complaints or complications. Developing sentiment analysis would ultimately lead to more informed decision making and better well-being.

Methods:

Data Pre-Processing:

- Drop unused columns
  - These columns have no effect on the sentiment (extra noise)
  - We dropped the following columns:
  - id: Used to index the tweets in dataset, but we don’t need to consider any ordering of tweets to predict sentiment
  - date: Dropped date as we want to predict sentiment independent of time
  - flag: The flag represents the query used to retrieve the tweet, but this dataset had no flags included so we dropped it
  - user: Username is not necessary because it does not directly affect sentiment of the tweet and thus would only add noise.
- Clean the tweets
  - Removed mentions (@’s) to usernames to remove noise since usernames can include unrelated words.
  - Removed urls since they do not usually provide meaningful content to the tweet.
- Use Bert vector embeddings to convert text to Vector embeddings
  - Bert embeddings preserve the semantic relationship between different sentences, allowing us to capture relationships through classification models
  - We used the bert-base-uncased model from transformers library
  - This turned each tweet into a 768 element vector

- We split this vector by element into 768 features

ML Agorithms/Models:

- K-Means Clustering (Decided Not To Implement)
  - Initially, we decided to create a feature from clustering to use in our supervised learning with K-Means.
  - After actually implementing this code and evaluating the results, we found that it did not have much of an impact on our evaluation metrics for our supervised learning algorithms.
  - Therefore, we decided, that we would use a different unsupervised learning method. We decided to implement Isolation Forest.
- Isolation Forest
  - We use this unsupervised learning algorithm in order to detect outliers and anomalies in our data.
  - Removing outliers helps the model generalize better rather than fitting to anomalies
  - `from sklearn.ensemble import IsolationForest`

```
from sklearn.ensemble import IsolationForest
from sklearn.model_selection import train_test_split

x = df.drop(columns=['target'])
y = df['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state=42)




iso = IsolationForest(contamination=0.1, random_state=42)
outlier_labels = iso.fit_predict(x_train)

x_train_cleaned = x_train[outlier_labels == 1]
y_train_cleaned = y_train[outlier_labels == 1]

x_train_cleaned.shape, y_train_cleaned.shape
```

  - Works by randomly selecting a feature and a split value. Performs random recursive partitioning to isolate a sample. The number of splits/path length is a measure of normality. We detect anomalies by outcomes with shorter paths.
  - With this processing of our dataset, we were able to achieve better accuracy on our random forest and SVM classifier models outlined below.

- Random Forest
  - We implemented a random forest because we believe it would be effective since we are dealing with text, so simpler models probably won't catch the complex relationships with tweets. Also random forest is robust to overfitting and handling high dimensions when we create our feature vectors.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import metrics
clf = RandomForestClassifier(n_estimators = 100, max_depth= 15, random_state=42)
clf.fit(x_train_cleaned, y_train_cleaned)
y_pred = clf.predict(x_test)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7458333333333333
```

  - We used an n_estimators of 100 and a max depth of 15. We did our training on 80% of the dataset and tested using the other 20%.
- Support Vector Machine
  - We implemented an SVM since we believe that SVM's are effective for us since they are good with high-dimensions, they work well with unstructured data, and they are less sensitive to overfitting [2].

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics

clf = SVC(kernel='rbf', C=1.5, gamma='scale', random_state=42)

clf.fit(x_train_cleaned, y_train_cleaned)

y_pred = clf.predict(x_test_dropped)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.785
```

  - Specifically, we decided to use a radial basis function (rbf) as our kernel to allow for higher dimensionality. We used an 80/20 train/test split again.

Overall Plan: We used BERT to create vector embeddings for tweets to use as features [1]. We then used unsupervised learning (isolation forest), to remove outliers and anomalies from our dataset. We then 2 run supervised learning methods. We used Random Forest and SVM Classifier and compared the quantitative metrics between the two.

Quantitative Metrics:

- Recall
- F1 Score
- Precision
- Accuracy

Project Goal:

- Our goal is to develop a sustainable sentiment analysis model that effectively scores tweets as 0 or 1.
  - To tackle sustainability, we will ensure our algorithm works efficiently to minimize the processing power and energy required to train the model
  - To tackle potential ethical dilemmas, we are using a dataset that does not include physical factors of the users.

Expected Results:

- Recall > 75%
- F1 Score > 75%
- Precision > 75%
- Accuracy > 75%

Note: Initially we wanted our scores to be above 80%. This number was obtained from a paper we read that expected similar results. However, that paper trained on significantly more data than we are training on (>1 million vs <30,000). Therefore, we reduced our expected metrics to account for that change.

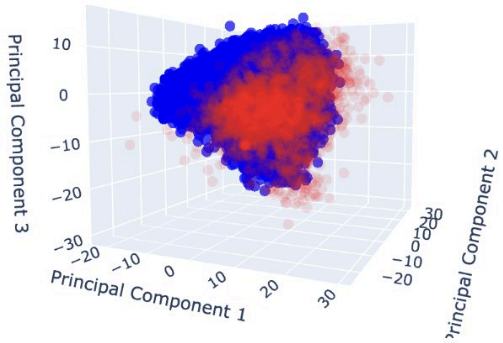
Results

Visualizations

We display our visualizations below. They include confusion matrices, the top 50 features, the random forest tree, 3d Scatter Plots using PCA, .

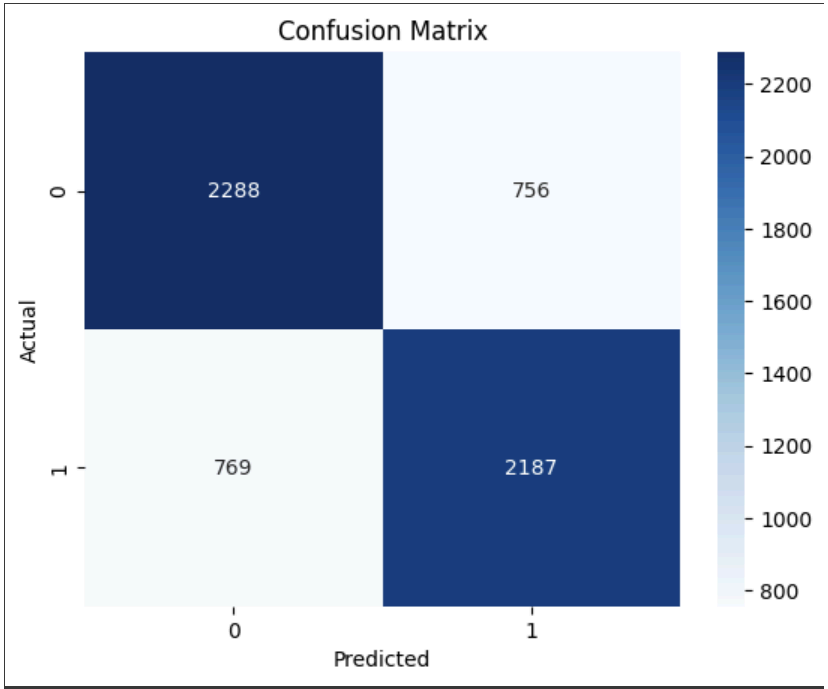
Isolation Forest (Unsupervised Learning) Plot

This plot contains all of our datapoints and shows which ones were decided to be outliers in a red color. We use PCA with 3 dimensions in order to visualize this data.



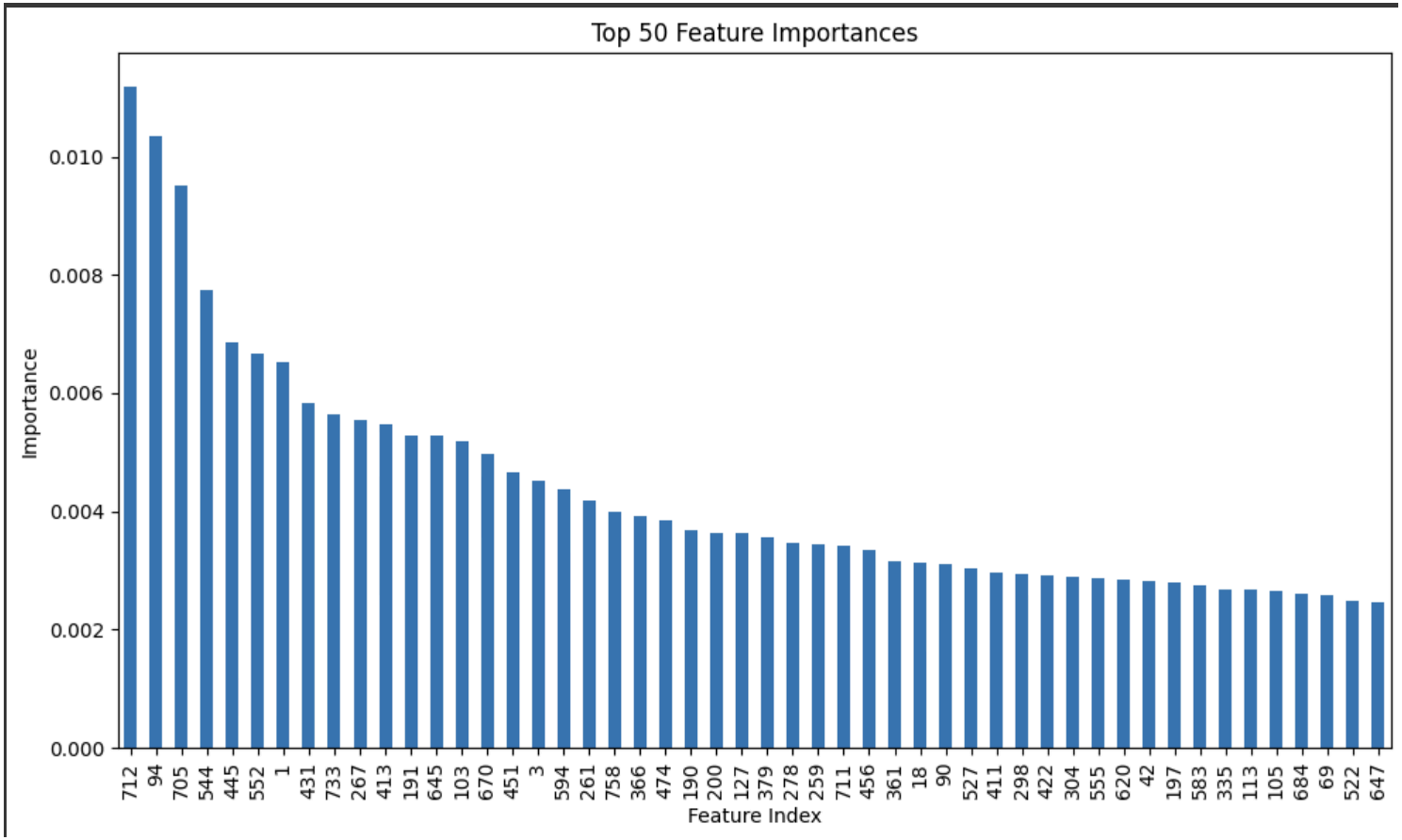
Random Forest Confusion Matrix

The confusion matrix is a matrix that visually shows the number TP, TN, FP, and FN. Our picture shows a heat map of these values. This confusion matrix is for the Random Forest model.



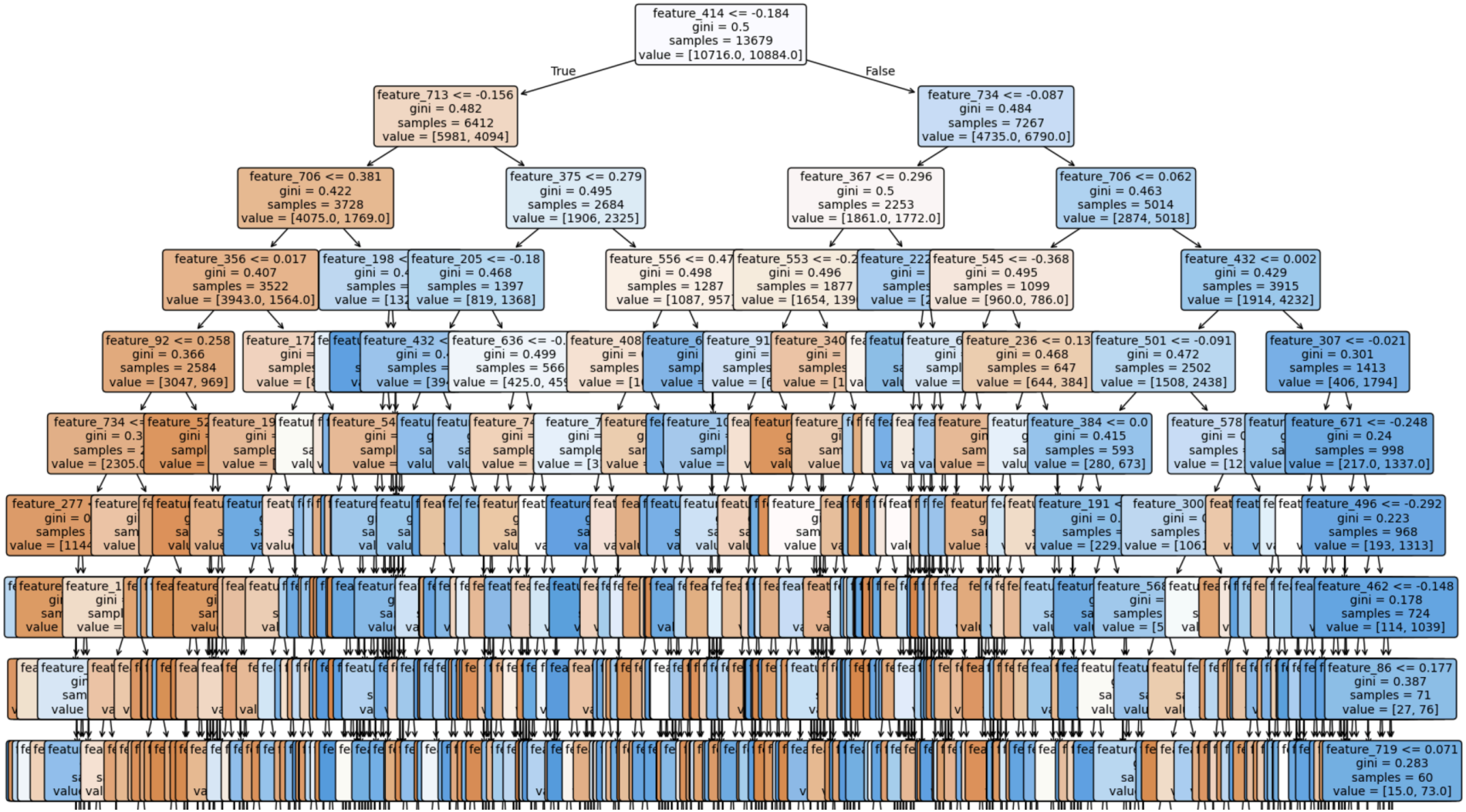
Top 50 Features

The random forest generated several features based on the vector embeddings from the Bert Model. We have displayed the top 50 features sorted for importance.



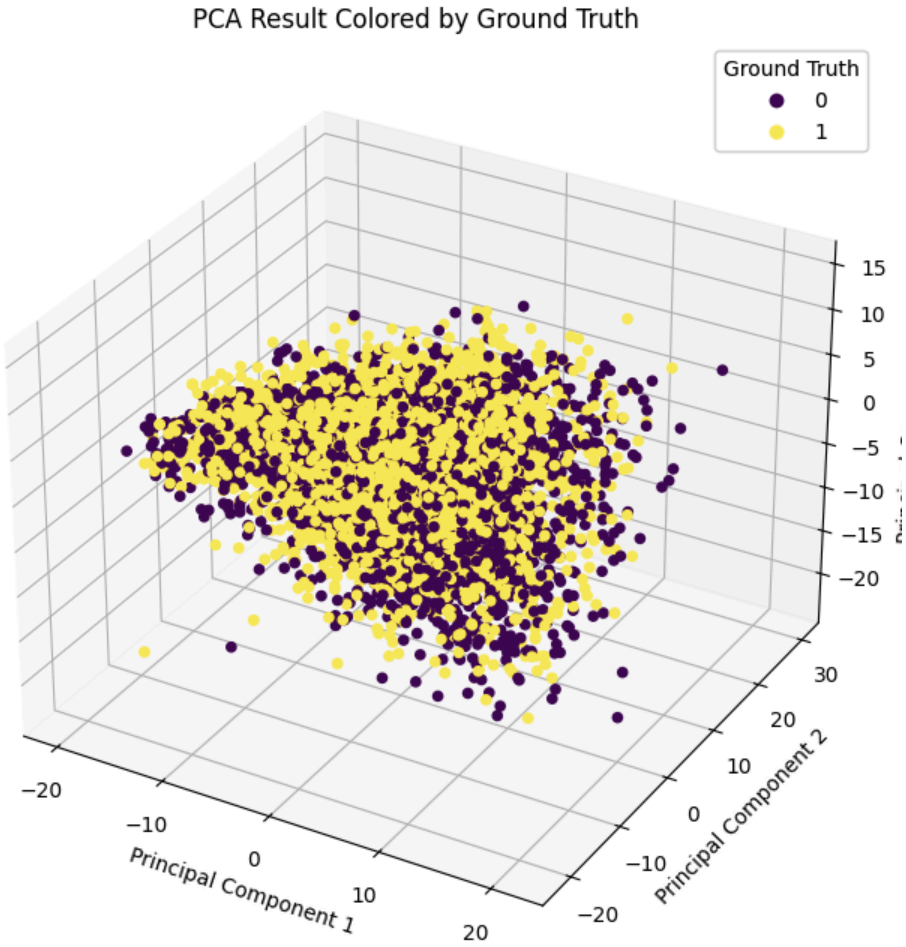
Random Forest Tree

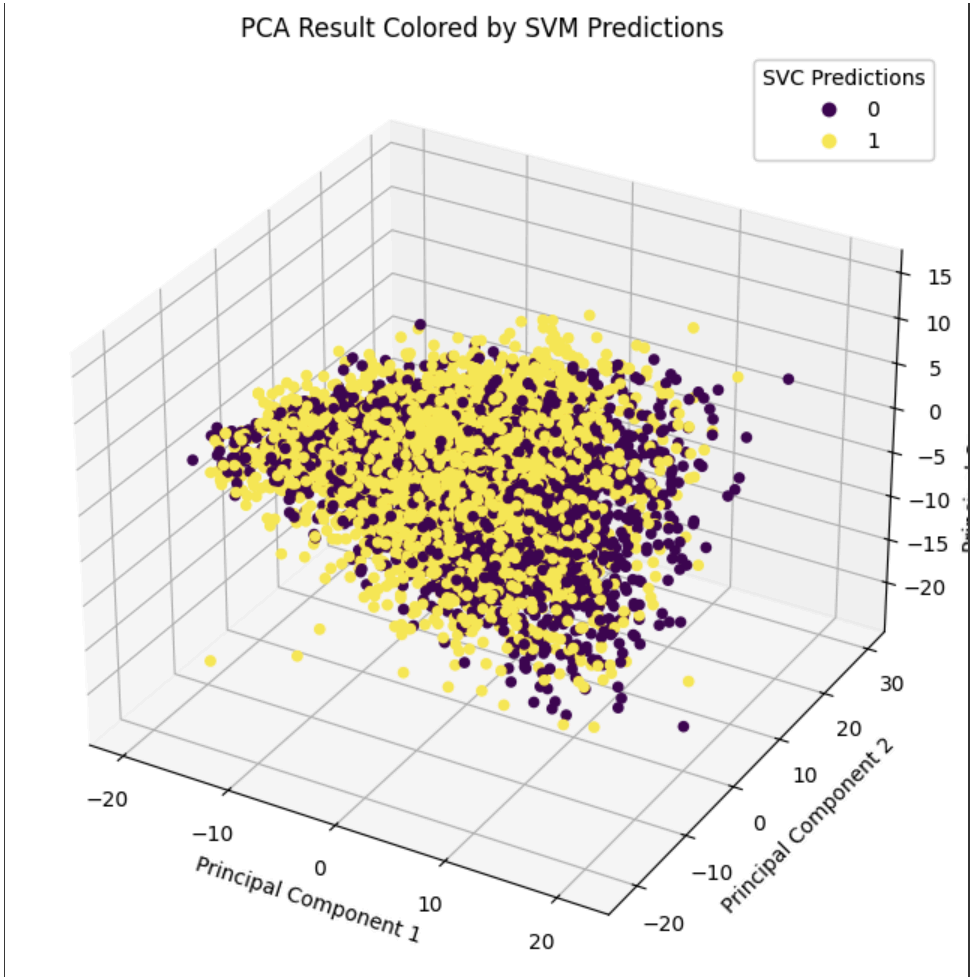
This visualizations attempts to graph the tree produced from our model. This shows the splitting criterion for the node, the gini impurity or measure of the node's purity, the number of training samples that can reach this node, and the value distribution or which class is dominant at each node.



SVM Results Displayed through PCA

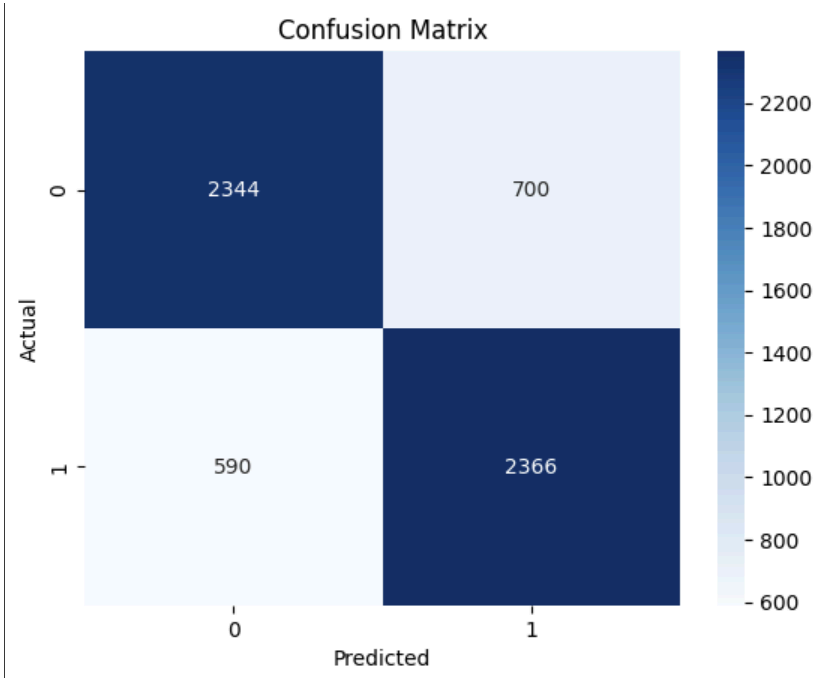
This visualization shows the result of our SVM classifier by performing PCA to reduce to 3 dimensions so we can see the results visually. We show the ground truth scatterplot in comparison to the scatterplot for what the SVM classifier led to.


PCA Result Colored by Ground Truth
Ground Truth
0
1
15
10
5
0
−5
−10
−15
−20
30
20
10
0
−10
−20
Principal Component 2
−20
−10
0
10
20
Principal Component 1



SVM Confusion Matrix

Below is the confusion matrix for the SVM Classifier model.




```
Accuracy: 0.7850
```

Quantitative Metrics

Unsupervised Learning

This table shows the number of points determined to be outliers from the Isolation Forest algorithm. Overall, 10% of the sampeled data was recognized as an outlier.

```
Total points: 24000
Non-outliers: 21600
Outliers (dropped points): 2400
```

Supervised Learning

We display 4 metrics: accuracy, precision, recall, and F1-score. Our results show all of our metrics are around 75% (due to a similar number of false positive and fales negatives).

Accuracy is the overall correctness of the model's predictions. It's a good overall indication of how correct the model is on the test data.

A high precision indicates that the model's predictions of positive instances are usually correct.

A high recall indicates that the model is able to identify a large proportion of the actual positive instances in the dataset.

F1-score provides a balanced measure of model performance, especially useful when classes are imbalanced.

The F1-score ranges from 0 to 1, where 1 represents perfect precision and recall. A high F1-score indicates that the model is performing well in terms of both precession and recall.

Here are our Random Forest Metrics:

	precision	recall	f1-score	support
0	0.75	0.75	0.75	3044
1	0.74	0.74	0.74	2956
accuracy			0.75	6000
macro avg	0.75	0.75	0.75	6000
weighted avg	0.75	0.75	0.75	6000

Here are our SVM Classifier Metrics:

	precision	recall	f1-score	support
0	0.80	0.77	0.78	3044
1	0.77	0.80	0.79	2956
accuracy			0.79	6000
macro avg	0.79	0.79	0.78	6000
weighted avg	0.79	0.79	0.78	6000

Analysis

Initially, we were going to use K-Means to append the cluster of the sample as a feature in supervised learning. We soon realized this was not ideal since we already have over 700 features, and, as expected, it made little to no difference on our metrics to use cluster as a feature. Instead, we decided to use another unsupervised method (Isolation Forest) to remove outliers. From the visualization of Isolation Forest (we used PCA with 3 components in order to plot the data), we can see which points were determined to be outliers. Visually, these outlier points tend to be on the outer shell of the datapoints which makes sense intuitively. Overall, we dropped 2400 samples that were determined to be outliers.

Analyzing our quantitative metrics for supervised models, we see that we achieved a accuracy, precision, recall, and f1-score of around 0.75 for all metrics with the Random Forest Classifier. From the confusion matrix for this we still have a good proportion of false positives and false negatives during our classification, meaning there is room for improvement for our model. We can also see that our True Positive and True negatives are both high, meaning the model is not clearly biased towards one classification over the other. Still, we just barely met our goal of achieving 75% for these metrics. We plotted the top 50 features of our BERT embeddings and saw that some features were clearly more important than others. As a result, we believe that reducing our feature space to focus more on these better features might potentially improve some of our metric scores. Additionally, the visualization of one of our random forest trees shows that it is very complicated. This is not necessarily a bad thing, but reducing our feature space could lead to a simpler model that still performs well.

Now looking at the SVM classifier, we see that for our quantitative metrics we achieved a accuracy, precision, recall, and f1-score of around 0.79 for all metrics. This is better than what we were able to achieve with the Random Forest Classifier. From the visualizations of the PCA scatterplot, we can see that many of our classifications mostly match the ground truth classifications which is great. However, we can see from the plot that the SVM labeled a lot more datapoints as ‘1’ than the ground truth had. Note that this plot with only 3 dimensions still does not present a clear separation between the two classes. This is simply due to the complexity of our data, as 3 features are not enough to differntiate the classes visually. Also, from the confusion matrix, we see that we have a good proportion of false positives and false negatives again, therefore there is improvement to be had. The main improvement of SVM over Random Forest is that SVM is better at avoiding False Negatives than Random Forest is. For this reason, SVM is our preferred method for this problem.

Next Steps

Our next steps include improving both our existing features and random forest model while also exploring new models. Firstly, our features consits of BERT embeddings which have 768 components. We found through our Random Forest analysis that some of these features are not very impactful. We plan to reduce our feature space to simplify the model and hopefully lead to higher scores. We also want to explore new models to improve our accuracy. For example, we might want to look into using ensembles(VotingClassifier) to combine the two as well as potentially other models/classifiers. Another thing that we could do is to train on substantailly more data in hopes of improving our matrices.

References:

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Proceedings of the 2019 Conference of the North, vol. 1, 2019, doi: https://doi.org/10.18653/v1/n19-1423.

[2] F. Tabsharani, “What is support vector machine (SVM)? - Definition from WhatIs.com,” WhatIs.com, Aug. 2023. https://www.techtarget.com/whatis/definition/support-vector-machine-SVM

[3] M. Rathi, A. Malik, D. Varshney, R. Sharma, and S. Mendiratta, “Sentiment Analysis of Tweets Using Machine Learning Approach,” 2018 Eleventh International Conference on Contemporary Computing (IC3), Aug. 2018, doi: https://doi.org/10.1109/ic3.2018.8530517.

Member	Proposal Contributions	Midterm Report Contributions	Final Report and Video Contributions
Arvind	Introduction, background, problem definition	developed the visualizations and results analysis	Conducted the next steps and analysis of models based on Rohits visualizations, slide presentation
Sai	ML Methods and research	handled Bert embeddings for preprocessing	Developed the code to execute SVM classifier models with preprocessed data, slide presentation
Rohit	Results, discussion, and setup of pages	handled data upload and cleaning up the text (modifiying labels, removing mentions etc.)	Created visualizations for analysis of SVM and PCA comparison, slide presentation
Navin	References, slides, youtube video	developed the code to execute the Random Forest model based on Bert Vector Embeddings	Implemented unsupervised learning model isolation forest to preprocess data, slide presentation

Gantt Chart

GANTT CHART

PROJECT TITLE	Twitter Sentiment Analysis

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	M
Project Proposal					
Introduction & Background	All	9/27/21	10/4/21	7d	
Problem Definition	All	9/27/21	10/4/21	7d	
Methods	Arvind	9/27/21	10/4/21	7d	
Potential Results & Discussion	Sai	9/27/21	10/4/21	7d	
Video Recording	Rohit	10/4/21	10/7/21	3d	
GitHub Page	Navin	10/4/21	10/7/21	3d	
Model 1					
Data Sourcing and Cleaning	Rohit	10/7/21	10/15/21	8d	
Model Selection	Navin	10/15/21	10/18/21	3d	
Data Pre-Processing	Sai	10/18/21	10/25/21	7d	
Model Coding	Navin	10/25/21	11/8/21	13d	
Results Evaluation and Analysis	Arvind	11/8/21	11/16/21	8d	
Midterm Report	Sai, Arvind, Rohit, Navin	11/8/21	11/16/21	8d	
Model 2					
Data Sourcing and Cleaning	Arvind	10/18/21	10/22/21	4d	
Model Selection	Sai	10/22/21	10/25/21	3d	
Data Pre-Processing	Rohit	10/25/21	10/29/21	4d	
Model Coding	Navin	10/25/21	11/19/21	24d	
Results Evaluation and Analysis	Sai, Arvind, Rohit, Navin	11/19/21	11/24/21	5d	
Model 3					
Data Sourcing and Cleaning	Sai	10/18/21	10/22/21	4d	
Model Selection	Rohit	10/22/21	10/25/21	3d	
Data Pre-Processing	Navin	10/25/21	10/29/21	4d	
Model Coding	Arvind	10/25/21	11/19/21	24d	
Results Evaluation and Analysis	Sai, Arvind, Rohit, Navin	11/19/21	11/24/21	5d	
Evaluation					
Model Comparison	Sai, Rohit	11/29/21	12/7/21	8d	
Presentation	Sai, Arvind, Rohit, Navin	11/29/21	12/6/21	7d	
Recording	Sai, Arvind, Rohit, Navin	12/6/21	12/7/21	1d	
Final Report	Sai, Arvind, Rohit, Navin	11/29/21	12/7/21	8d	