Data Mining

1st Abdullah Naveed
Computer Science

2nd Karan Kumar
Computer Science

3rd Ahmer Jamil
Computer Science

4th Esha Fatima
Computer Science

***Abstract*—This phase of the project deals with relevant mining techniqunues applied over our transformed dataset so as to gain relevant trends and information from it. Majorly, the paper focuses on key techniques such as logistic regression and Neural Networks.**

***Index Terms*—Neural Networks, regression, classification**

I. Introduction

The dataset transformed during the exploratory data analysis phases presents an encapsulation of features corresponding to tweets and the relevant tweet label - fake/real.

Using the transformed dataset at hand, this paper focuses on relevant mining techniques applied and the corresponding evaluation to gauge if there can be some model to classify a given tweet as real or fake.

II. Transformed dataset characteristics

The dataset consists of 201,531 entries of tweets that each possess 12 attributes as follows:

- user-id
- tweet-id
- text
- followers
- following
- location
- verified
- statuses-count
- profile-background-tile
- profile-use-background-image
- favorite-count
- label

III. Classification Techniques

The paper briefs over several different classification techniques applied over to the dataset. The goal is to possible devise different classification models and evaluate their accuracy to decide which one is most appropriate for tweet label classification.

A. Naive Baye's classification

We trained a Sci-kit implementation of Naive Baye's classifier for developing a model suited to classify tweets as real or fake.

Preprocessing

For each of the tweet entries in the dataset, the tweet text was preprocessed to convert the text to lowercase, remove hyperlinks, usernames, digits, next line symbols, and punctuation marks. Finally, all stop words are removed to ensure that generic verbal words and pronouns do not cause overfitting. A detailed list of stop-words that are removed is given below:

'i', 'iḿ', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'youŕe', 'youv́e', 'youĺl', 'youd́', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'sheś', 'her', 'hers', 'herself', 'it', 'itś', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'thatĺl', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'doń', 'should', 'shouldv́e', 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', 'areń', 'couldn', 'couldń', 'didn', 'didń', 'doesn', 'doesń', 'hadn', 'hadń', 'hasn', 'hasń', 'haven', 'haveń', 'isn', 'isń', 'ma', 'mightn', 'mightń', 'mustn', 'mustń', 'needn', 'needń', 'shan', 'shań', 'shouldn', 'shouldń', 'wasn', 'wasń', 'weren', 'wereń', 'won', 'woń', 'wouldn', 'wouldń'

Train-Test Split

The entire data was randomly divided into training data and test data using a 70-30 train-test ratio. Corresponding to each of the training and testing data, the tweet- label column is isolated such that there is a feature vector (X-test and X-train) and a label vector (Y-test and Y-train) for both the training and test data.

Accordingly the number of samples in each of the training and test data are as follows:

- Training data - 141056
- Test data - 60453

Count Vectorizer

- A list, L, of all unique words in the training dataset's tweet text is extracted.
- A count vectorizer is used to indicate the count of each

Identify applicable funding agency here. If none, delete this.

of each word in L for each tweet in the training data.
- Consequently, the training dataset is transformed such that each row corresponds to a single tweet and the columns list the counts of each of each unique words in L.
- Similarly, the test data set is mapped and transformed onto each of the unique words in L

In our example, the number of unique words in the tweets text for training data after preprocessing are 90580. Hence, the dimensions of the transformed test and training matrices are as follows:

- Transformed training data - 141056 x 90580
- Transformed test data - 60453 x 90580

To automate this process we use Sci-kit's count vectorizer.

Training and Evaluation

The transformed feature vector for training data is fed into the Sci-kit's Multinomial Classifier to produce a classification model.

For evaluation, we predict the labels for the transformed feature matrix for the test data.

Results

The trained classifier reported an overall accuracy of 92.6% on the test data. Additionally, the model reports an F1- score of 0.926 as

—	Actual Positives	Actual Negatives
Predicted Positives	27125	2952
Predicted Negatives	1509	28867

B. Logistic Regression

Data Extraction

The location column is transformed via a label encoder so that each of the unique values in the location column is tagged by a unique numerical ID. This is done via a Sci-kit library's Label Encoder.

We extract relevant columns from the dataset so as to ensure that only relevant attributes are used in the training process of our classifier. Consequently, a heatmap that depicts the correlation between different numerical attributes is created.

The relevant attributes meaningful to our classifier are:

- user-id
- tweet-id
- followers
- following
- location
- verified
- statuses-count

Additionally, it is worth noting that there are very few unique user-ids and tweet-ids, which implies that our model can make use of this in our training process.

Data Normalization

The values in the numerical attributes are scaled down to values between 0 and 1 for the purposes of uniformity between various attributes.

Train-Test Split

The entire data was randomly divided into the training data and test data using a 80-20 train-test ratio. Corresponding to each of the training and testing data, the tweet- label column is isolated such that there is a feature vector (X-test and X-train) and a label vector (Y-test and Y-train) for both the training and test data.

Training and Evaluation

The training data was trained using the Logistic Regression Classifier of the Sci-kit library.

Results

The trained classifier reported an overall accuracy of 76.5% on the test data. Additionally, the model reports an F1- score of 0.765 as

—	Actual Positives	Actual Negatives
Predicted Positives	15078	4966
Predicted Negatives	4479	15779

C. Neural Net Classifiers

Classification types

In this section, we discuss and implement classificaction using deep-learning based models..

Primarily, we train classifiers of 3 types:

- Multilayer Perceptron
- Support Vector Machine
- Linear Perceptron

A linear perceptron is a two-layer network that has only one input and output layer. However, a multi-layer perceptron has at least one hidden layer.

Results

The trained classifiers reported an overall accuracy and F1 score as given below:

- Multilayer Perceptron-
  - Accuracy - 81.3%
  - F1-score - 0.813
- Linear Perceptron
  - Accuracy - 49.7%
  - F1-score - 0.335
- Support Vector Machine
  - Accuracy - 76.9%
  - F1- score - 0.769

Confusion Matrix for Multilayer Perceptron

—	Actual Positives	Actual Negatives
Predicted Positives	15832	4212
Predicted Negatives	3308	16950

Confusion Matrix for Linear Perceptron

—	Actual Positives	Actual Negatives
Predicted Positives	19982	62
Predicted Negatives	20193	65

Confusion Matrix for Support Vector Machine

—	Actual Positives	Actual Negatives
Predicted Positives	16004	4040
Predicted Negatives	5252	15006

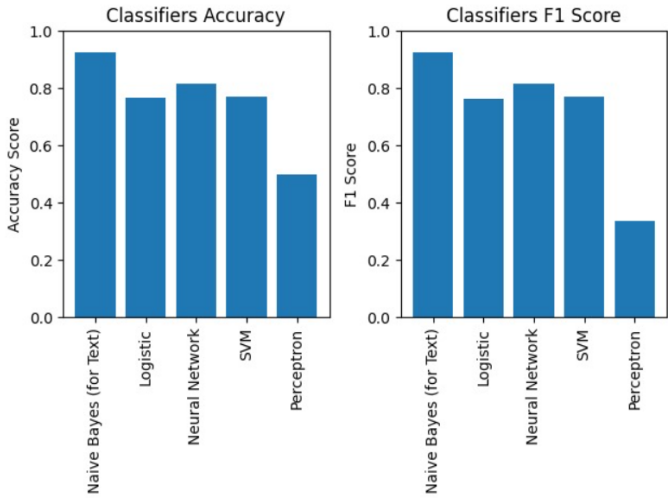

Fig. 1. Results

D. Feature Selection for Multilayer Perceptron

From the above observed results, it is evident that a multilayer perceptron obtains the highest accuracy and F1 score. This derives us to possible exploratory room for improved accuracy for the multilayer perceptron.

Hence, we employ feature selection to choose out the best possible combination of features that would prevent the model from overfitting to training instances

. The three major techniques that we employ for feature selection as follows:

1) Filter method
2) Wrapper selection and Decision tree classifier
3) Wrapper selection and Gradient Boosting Regressor

Additionally, the tweet text column is obviously unique and is not considered in our analysis. Incorporating tweet text through one hot encoding resulted in an excessive number of dimensions (201509 rows × 118160), making it impractical to compute and causing memory problems.

Filter Method

In this approach, we set a variance threshold 0.1. Any attribute that observes a variance value lower than this threshold is stagnant for our classifier and does not add any new information. Hence, such attributes are discarded.

Given this approach, the attributes that remain after filtering are as follows:

- user-id
- tweet-id
- followers
- following
- location
- statuses-count
- profile-background-tile
- profile-use-background-image
- favorite-count

Wrapper selection and Decision tree classifier

For this approach we use Recursive Feature Selection (RFE) along with a Decision Tree Classifier as the estimator for (RFE). Additionally, the number of features that we wish to obtain are 7.

The selected features by our model as given below:

- user-id
- tweet-id
- followers
- following
- location
- statuses-count
- favorite-count

Wrapper selection and Gradient Boosting Regressor Similar to the previous approach an RFE model is used to extract out 7 features. However, the estimator used is Gradient Boosting Regressor. The features selected by our model are as listed below:

- user-id
- tweet-id
- followers
- statuses-count
- profile-background-tile
- profile-use-background-image
- favorite-count

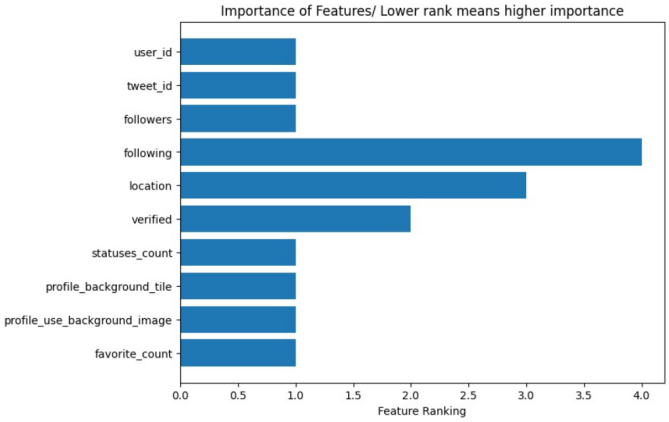

Fig. 2. Relative importance of features

Training and Evaluation

We train and evaluate the Multilayer Perceptron on 4 possible feature sets.

1) All columns except text - (retained for comparison) - **Set-1**
2) Feature selection using variance filter method with threshold 0.1 - **Set-2**
3) Feature selection using wrapper method on decision tree classifier - **Set-3**
4) Feature selection using wrapper method and gradient boosted regression - **Set-4**

Confusion Matrix for Set - 1

—	Actual Positives	Actual Negatives
Predicted Positives	15832	4212
Predicted Negatives	3308	16950

Confusion Matrix for Set - 2

—	Actual Positives	Actual Negatives
Predicted Positives	18078	1966
Predicted Negatives	5933	14325

Confusion Matrix for Set - 3

—	Actual Positives	Actual Negatives
Predicted Positives	15651	4393
Predicted Negatives	2711	17547

Confusion Matrix for Set - 4

—	Actual Positives	Actual Negatives
Predicted Positives	16276	3768
Predicted Negatives	3633	16625

Accuracy and F1 Scores

—	Accuracy	F1 score
Set - 1	81.3%	0.813
Set - 2	80.4%	0.804
Set - 3	82.3%	0.823
Set - 4	81.6%	0.816

From our evaluation and the above provided results, it is evident that feature selection using wrapper method on decision tree classifier performs best on the given test data instances.

IV. Frequent Pattern Mining

Frequent pattern mining is a data mining technique that aims to identify recurring patterns in large datasets. The process involves detecting sets of items that frequently appear together in a dataset, such as commonly purchased items in a supermarket. In this project, we employ frequent pattern mining on our dataset to explore the relationship between repetitions and labels.

Apriori

Due to limitations in processing power, we are unable to run the Apriori algorithm on the entire dataset. Instead, we break down the data into smaller chunks and analyze repeated patterns within each chunk. Before running Apriori, we preprocess the data by converting all values in columns to strings and appending the column names to the values. This is necessary to differentiate between identical values in different columns since Apriori tries to establish relationships across columns.

We run the Apriori algorithm twice. First, we analyze the Boolean attributes, tweet-id, and favourite count, resulting in a total of six attributes: tweet-id, verified, profile-background-tile, profile-use-background-image, favourite-count, and label. By focusing on these columns, we are able to run Apriori without encountering memory issues. We identify a list of repeated patterns, which we further divide to determine the frequent patterns for labels 1 and 0, respectively.

One possible approach is to execute Apriori with various combinations and consolidate the findings to identify the most frequently occurring patterns. Nonetheless, considering time constraints and the priority of specific attributes highlighted by feature selection, we proceed with the previously selected features and run Apriori. To enhance the likelihood of discovering frequent patterns, we eliminate the user-Id and statuses-count columns from our dataframe, which contained a greater number of unique values and fewer repetitions. We then apply Apriori to the remaining attributes and generate a list of the most prevalent patterns evident in the data.

FP-Growth

FP-growth builds a compact data structure called a frequent pattern tree (FP-tree) to represent the transactional database, which allows it to efficiently mine frequent item sets without generating candidate item sets explicitly. Compared to Apriori, FP-growth is generally faster and uses less memory.

Therefore, we apply the FP growth algorithm to our complete dataset, where we specify the column names along with their respective values for clarity, as we have done before. Running the algorithm generates a list of all frequent patterns. However, we discard any frequent patterns that do not involve the label since our aim is to identify the connection between frequent patterns and labels.

By following the same procedure and setting the support threshold to 0.9, we identify the most frequently occurring patterns. Then, we generate separate outputs for the frequent patterns associated with labels 1 and 0, respectively, which enables us to gain insights into the prevalent patterns in each label. Moreover, this analysis reveals that some patterns are more likely to occur in one label than the other, which could be leveraged to enhance classification tasks.

V. Conclusion

Using the analysis we can conclude that Neural Networks based classifier performs significantly better than other classification techniques. Furthermore, there exists potential room for overfitting which needs to be avoided via feature selection. Additionally, the multinomial classifier (Naive Baye's) reports attainably the highest accuracy as it relies on the bag of words implementation focusing solely on the crux of tweet text. There is little attention paid on extraneous attributes and the probabilistic modelling is conditioned over tweet text. Hence, it can be safely labelled as the most appropriate classification technique for the given data semantics which include a heavy reliance on tweet text.

We are able to extract out maximally overlapping frequent patterns in our dataset using various features selected. This indicates that there are significant trends or patterns within attributes that occur more often than the other. Given the high amount of possible variance in the data, any patterns coming being observed more frequently than the rest would indicate a potential significance.