Fact or Fiction: A Data Mining Exploration of the LIAR Dataset

Milestone

Draft of Project Report

Group 70

Student 1: Ruchirkanth Gandikota

Student 2: Ankuran Chattoraj

gandikota.r@northeastern.edu

chattoraj.a@northeastern.edu

Percentage of Effort Contributed by Student 1: 50%

Percentage of Effort Contributed by Student 2: 50%

Signature of Student 1: Ruchirkanth Gandikota

Signature of Student 2: Ankuran Chattoraj

Submission Date: 04/07/2023

Table of Contents

Fact or Fiction: A Data Mining Exploration of the LIAR Dataset

Problem Setting:

The increasing spread of misinformation in various forms of media is a growing concern for society. The ability to detect and fact-check false statements is crucial in maintaining the integrity of information and protecting the public from being misled.

Problem Definition:

The goal of this project is to develop a model that can accurately classify the veracity of political statements using the LIAR dataset. Specifically, the model will predict whether a statement is true, mostly true, half true, mostly false, or false.

Data Sources:

The LIAR dataset, created by researchers at the University of California, Berkeley, can be found at https://www.cs.ucsb.edu/~william/data/liar_dataset.zip . The dataset contains 12,836 statements from various political figures, along with labels indicating the veracity of each statement.

Data Description:

The LIAR dataset contains the following columns: statement, subject, speaker, speaker's job title, state info, party affiliation, barely true counts, false counts, half true counts, mostly true counts, pants on fire counts, context, and justifications. The dataset contains 12,836 rows and 14 columns. A sample of variable names include "statement", "subject", "speaker", "party affiliation", and "veracity label".

Data Dictionary:

- **ID**: a unique identifier for each statement
- **Label**: the truth rating assigned to the statement, with possible values of "pants-fire", "false", "mostly-false", "half-true", "mostly-true", and "true"
- **Statement**: the statement made by the speaker
- **Subject**: the topic of the statement
- **Speaker**: the name of the person who made the statement
- **Job**: the job or position of the speaker
- **State**: the state where the speaker is located

- **Party**: the political affiliation of the speaker
- **Barely True**: the number of times the speaker has previously made a statement rated "barely true"
- **False**: the number of times the speaker has previously made a statement rated "false"
- **Half True**: the number of times the speaker has previously made a statement rated "half true"
- **Mostly True**: the number of times the speaker has previously made a statement rated "mostly true"
- **Pants on Fire**: the number of times the speaker has previously made a statement rated "pants on fire"
- **Context:** additional information about the statement or the speaker.

Data Mining Tasks

Data Understanding

The Fake news dataset contains information about several articles and the level of their trueness. There are three dedicated datasets for training, testing and validation containing 12.8k records altogether. This dataset is a combination of both numeric and categorical variables. There are five numerical variables that are each a dedicated truthfulness level and the records in these variables hold the number of times the article has proven to be in that category. There is a statement column that contains short human-made statements from politifact.com's API and every statement is verified by the editor of Politifact. Hence the truthfulness of the articles depends on the information provided by the editor.

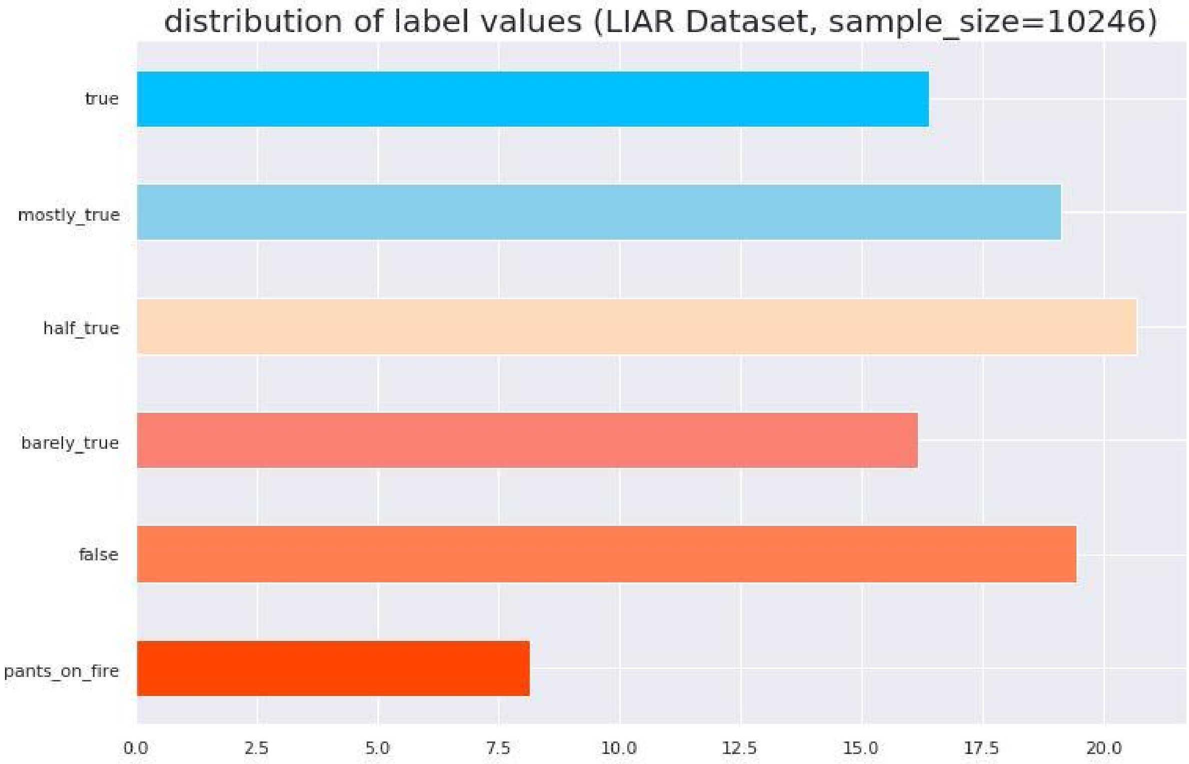
Data Pre-Processing

The variable 'id' is not helpful for the analysis since it is just a unique identifier for each record in the dataset. The data also seems to have a lot of blank cells and these cells have been replaced by 'NaN' so that Pandas can result in the count of NaNs for each column. It is noticed that the 'job_title', 'state_info' and 'context' columns contain 2878, 2185 and 77 NaN values respectively. The records where these three columns are NaN are dropped which results in the deletion of 23 records. All the NaN values are part of categorical variables and they will be handled by replacing NaN with 'Unknown'. Now the string 'Unknown' becomes a separate category in the respective columns where NaNs were present before. As a result, all the NaN values are handled and the crucial data is not lost.

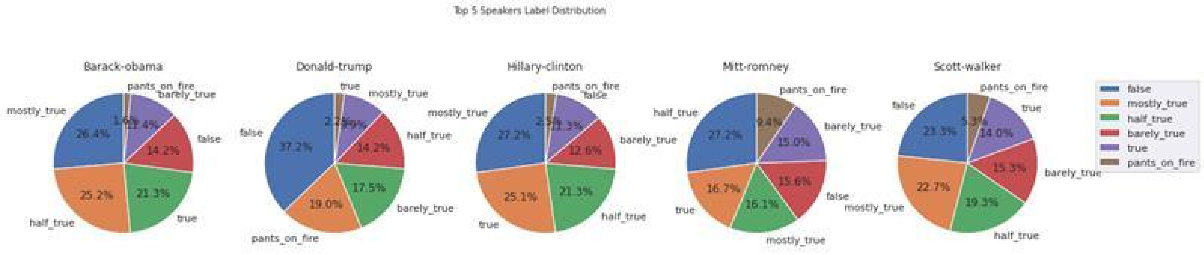
Data Exploration

Many visualization techniques have been used to explore the behaviour of categorical and numerical variables.

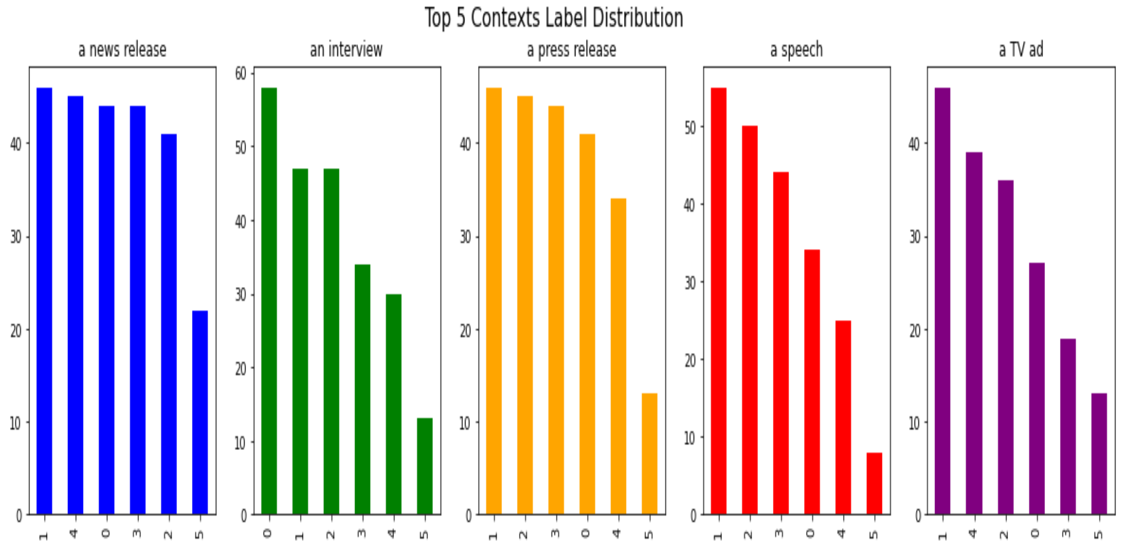
1) In the following visualization, we are exploring the distribution of the labels across the entire training dataset and presenting them using a horizontal bar chart.



2) We extract the top 5 speakers with the highest count of statements and create a pie chart for each of the top 5 speakers that shows the distribution of the labels for their statements. The resulting chart provides insight into the distribution of labels across the top speakers in the dataset.

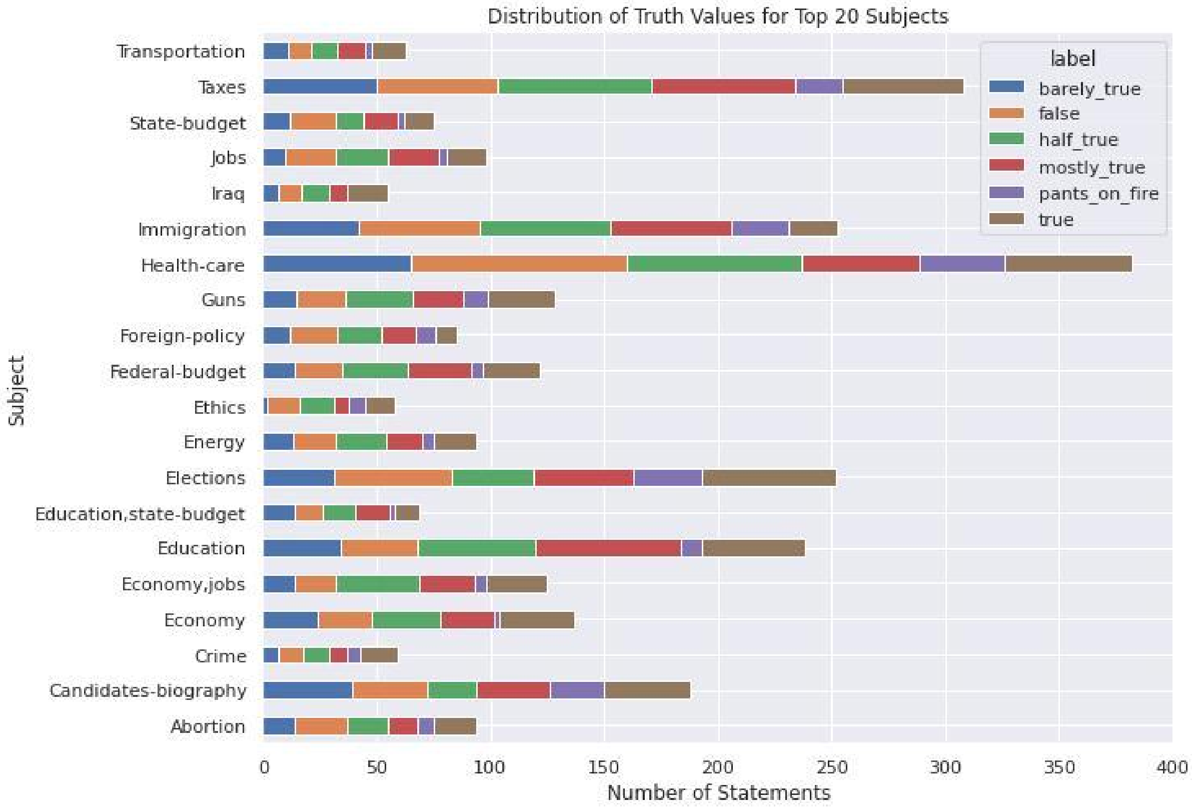


3) We are finding the top 5 Contexts by the label distribution of the dataset. We are grouping the data based on the 'context' column and counting the number of occurrences for each label.

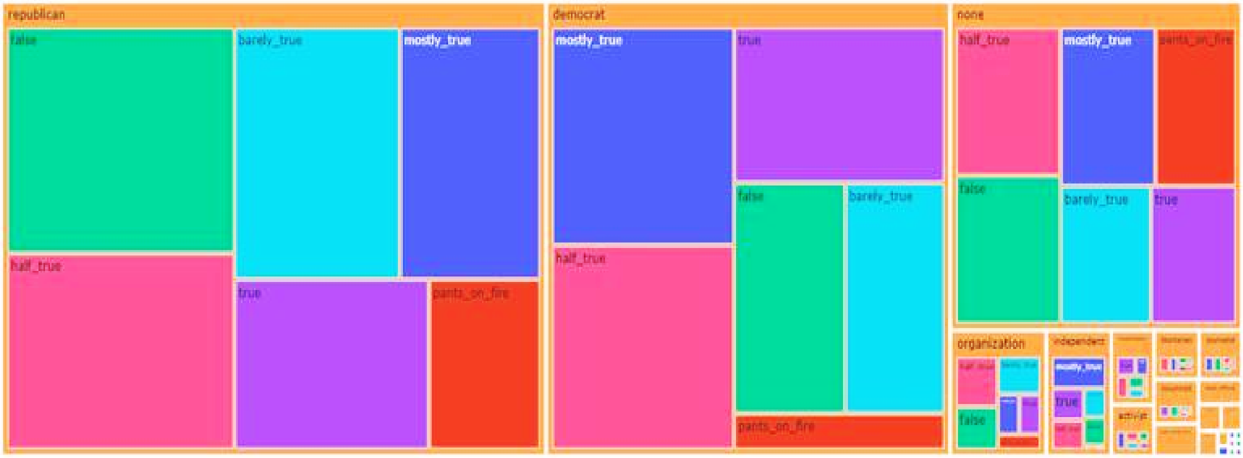


Overall, this helps us visualize the label distribution for the top 5 contexts in the dataset.

4) We first get the top 20 most common subjects from the liar dataset, then we filtered the dataset to only include these subjects. After that, we grouped the data by subject and true value and counted the number of statements in each group. Finally, we created a horizontally stacked bar chart to visualize the count of truth values for the top 20 subjects. The chart clearly shows how many true, false, and other statements there are for each subject.

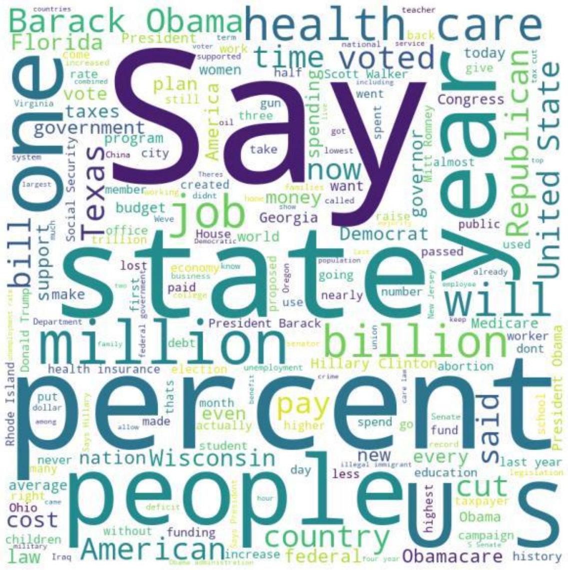


5) We have represented the distribution of statement labels across political affiliations using a treemap visualization. The treemap shows the distribution of statement labels across different political affiliations. The size of the boxes represents the number of statements, and the colour represents the label. The darker the colour, the higher the percentage of false statements.

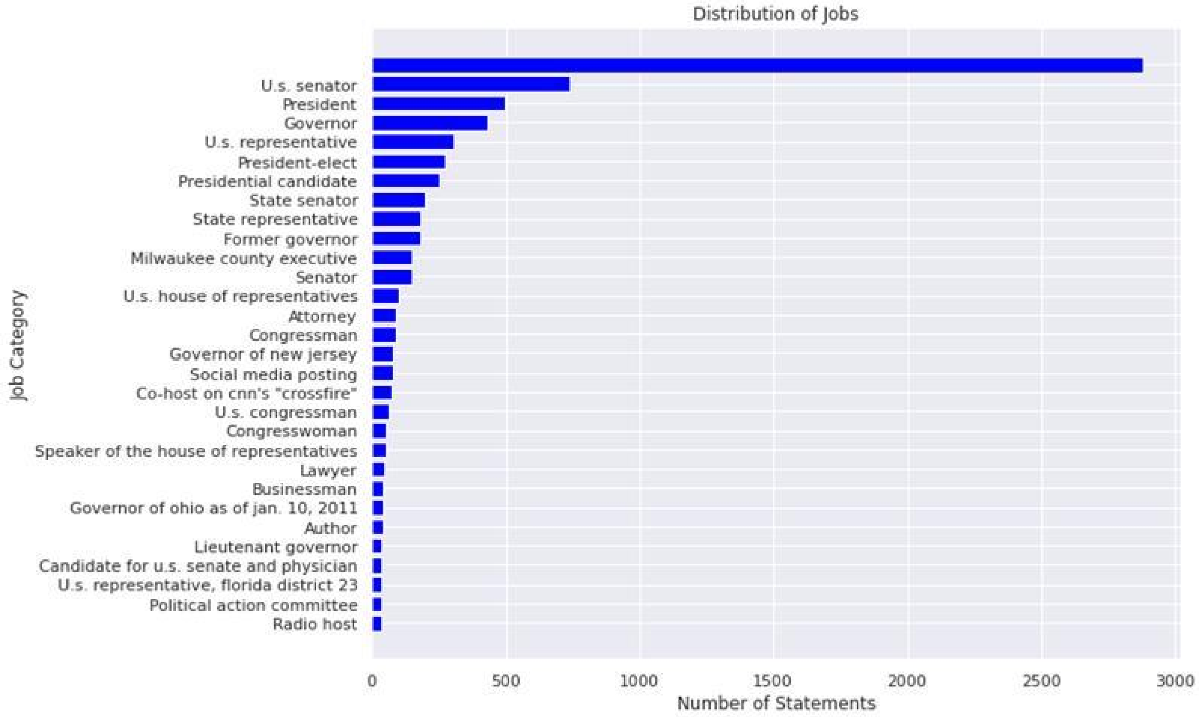


The treemap shows that the largest number of statements come from the republican party and the majority of them are rated as false. The democratic party also has a large number of statements, but a lower percentage of them are rated as false.

6) We have created a word cloud from the Liar dataset, which is a visual representation of the most commonly occurring words in the dataset. The word cloud helps to identify trends and patterns in the language used in the statements and to gain insights into the most commonly used words in the dataset.

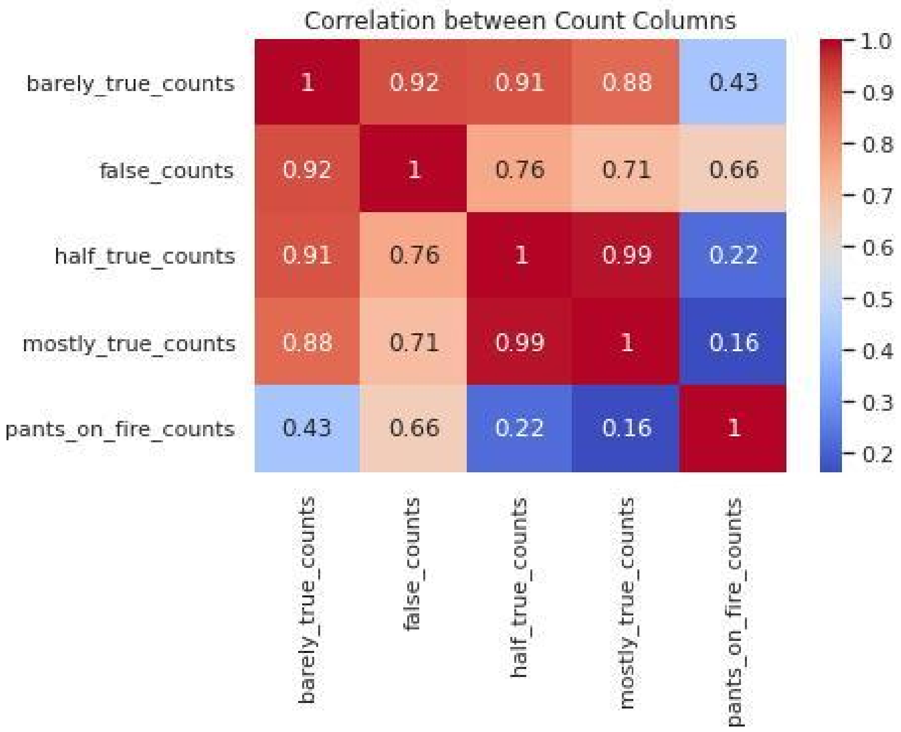


7) We have visualized the distribution of job categories in the Liar dataset and identified the job categories with the largest number of statements. The resulting horizontal bar chart shows the 30 job categories with the largest number of statements and their corresponding statement counts, which can provide insights into the prevalence of certain job categories in the dataset and their role in the generation of statements in the dataset.



8) We are exploring the relationship between the different count columns in the Liar dataset. The count columns represent the number of times a given speaker has been labelled as "barely true", "false", "half true", "mostly true", and "pants on fire".

   We use the seaborn library to create a heatmap of the correlation matrix. The heatmap is a graphical representation of the correlation matrix where each cell is coloured according to the value of the correlation coefficient. The heatmap is annotated with the correlation coefficients for better visualization. The colour map used is 'cool warm' which ranges from blue (negative correlation) to red (positive correlation). The title of the plot is set to 'Correlation between Count Columns'.



Overall, this code snippet helps us to understand the relationship between the different count columns in the Liar dataset and identify any strong positive or negative correlations between them.

Data Preparation

Convert text to word count vectors with CountVectorizer

In this step, the CountVectorizer module from scikit-learn is used to convert the text data into a matrix of word count vectors. The module tokenizes the text, builds the vocabulary, and encodes the training data. The output is a sparse matrix where each row represents a document and each column represents a unique word in the vocabulary.

Calculate inverse document frequencies

In this step, the TfidfTransformer module from scikit-learn is used to calculate the inverse document frequency (IDF) for each word in the vocabulary. IDF measures the rarity of a word in the corpus and is used to down weight the importance of common words. The output is a sparse matrix where each row represents a document and each column represents a weighted frequency of a word.

Combine TF-IDF with n-grams

In this step, the TfidfVectorizer module from scikit-learn is used to combine TF-IDF with n-grams. N-grams are contiguous sequences of n items from a given sample of text. By combining TF-IDF with n-grams, we can capture the context and sequence of words in the text. The output is a sparse matrix where each row represents a document and each column represents a weighted frequency of an n-gram.

Place tf-idf values in a pandas data frame

In this step, the output from the TfidfVectorizer module is converted into a pandas data frame for better visualization and analysis. The data frame has two columns - one for the n-gram and one for its corresponding TF-IDF value. The data frame is sorted by TF-IDF values in descending order to identify the most important n-grams in the text.

Model Selection

Naive Bayes

Naive Bayes is a probabilistic classification algorithm that assumes the independence of features. In the case of the Liar dataset, Naive Bayes can be used to calculate the probability of a statement being true or false based on its textual features. Naive Bayes is computationally efficient and can handle high-dimensional data well. The steps to perform Naive Bayes on the Liar dataset are:

1. Preprocess the data by removing stop words, stemming, and converting the text into a numerical representation.
2. Encode the labels assigned to each statement as numerical values (such as 0 for false and 1 for true).
3. Split the dataset into training and testing sets.
4. Train a Naive Bayes model, such as Multinomial Naive Bayes, on the training set.

5. Test the model on the testing set and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.

Logistic Regression

Logistic regression is a classification algorithm that can predict the probability of an event occurring. In the case of the Liar dataset, logistic regression can be used to predict the probability of a statement being true or false based on its textual features, source, and metadata. Logistic regression is relatively simple to implement and easy to interpret, which makes it an excellent starting point for analyzing the Liar dataset. The steps to perform logistic regression on the Liar dataset are:

1. Preprocess the data by removing stop words, stemming, and converting the text into a numerical representation.
2. Encode the labels assigned to each statement as numerical values (such as 0 for false and 1 for true).
3. Split the dataset into training and testing sets.
4. Train the logistic regression model on the training set.
5. Test the model on the testing set and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.

Support Vector Machines

Support Vector Machines (SVMs) are powerful classification algorithms that can handle non-linear relationships between features. SVMs can be used to find the hyperplane that best separates the true and false statements in the dataset. SVMs are flexible models that can be tuned by changing the kernel function to handle different data distributions. The steps to perform SVM on the Liar dataset are:

1. Preprocess the data by removing stop words, stemming, and converting the text into a numerical representation.
2. Encode the labels assigned to each statement as numerical values (such as 0 for false and 1 for true).
3. Split the dataset into training and testing sets.
4. Train the SVM model on the training set by selecting the appropriate kernel function.

5. Test the model on the testing set and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.

Random Forests

Random Forests are an ensemble learning method that uses multiple decision trees to make predictions. Each tree is trained on a subset of the data and a random subset of features to reduce overfitting. The predictions of the individual trees are then combined to make a final prediction. Random Forests are highly effective at handling complex relationships between features and can also provide feature importance rankings. The steps to perform Random Forests on the Liar dataset are:

1. Preprocess the data by removing stop words, stemming, and converting the text into a numerical representation.
2. Encode the labels assigned to each statement as numerical values (such as 0 for false and 1 for true).
3. Split the dataset into training and testing sets.
4. Train a Random Forest model on the training set with a large number of decision trees.
5. Test the model on the testing set and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.
6. Analyze the feature importance rankings to gain insights into which textual features are most important for predicting true or false statements.

Voting Classifier

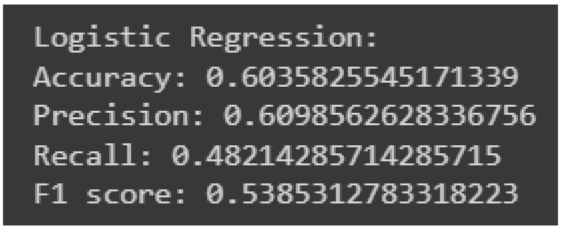
Voting Classifier is an ensemble learning method that combines multiple individual classification models to make predictions. Each model can use a different algorithm or hyperparameters, and the predictions of the individual models are combined using a majority vote or weighted average. Voting Classifier can often achieve higher accuracy than individual models and can also reduce the risk of overfitting. The steps to perform Voting Classifier on the Liar dataset are:

1. Preprocess the data by removing stop words, stemming, and converting the text into a numerical representation.
2. Encode the labels assigned to each statement as numerical values (such as 0 for false and 1 for true).
3. Split the dataset into training and testing sets.
4. Train multiple classification models, such as Naive Bayes, Logistic Regression, and SVM, on the training set.
5. Combine the individual models into a Voting Classifier by specifying the voting method (e.g., hard voting or soft voting) and the weights for each model.
6. Test the Voting Classifier on the testing set and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.
7. Tune the hyperparameters of the individual models and the weights of the Voting Classifier to improve its performance.

Performance Evaluation

Logistic Regression Model

The performance evaluation of the logistic regression model on the fake news dataset involves measuring its accuracy, precision, recall, and F1 score. The accuracy of the model tells us the percentage of correctly classified instances of fake news which has been obtained to be 60.3%. Precision, which has resulted to be 60.9% after the model fitting measures the proportion of true positives over the total number of positive predictions, which indicates the ability of the model to correctly classify instances of fake news. Recall, on the other hand, measures the proportion of true positives over the total number of actual positives, which indicates the ability of the model to detect all instances of fake news. Recall has been procured to be 48.2%. Finally, the F1 score is the harmonic mean of precision and recall, which balances both measures which is around 54% for the fake news dataset. By evaluating these metrics, one can determine the effectiveness of the logistic regression model in accurately classifying instances of fake news, which is essential in combating the spread of misinformation. Overall, a high accuracy, precision, recall, and F1 score indicate a reliable logistic regression model for detecting fake news.

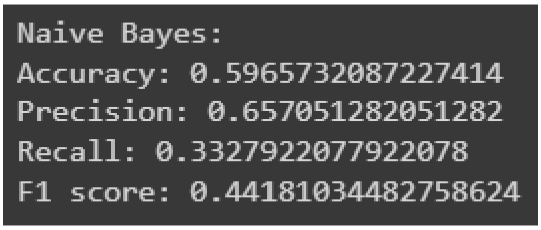


```
Logistic Regression:
Accuracy: 0.6035825545171339
Precision: 0.6098562628336756
Recall: 0.48214285714285715
F1 score: 0.5385312783318223
```

Naive Bayes

Performance evaluation of the Naive Bayes algorithm on the fake news dataset involves measuring its accuracy, precision, recall, and F1 score. Naive Bayes is particularly useful in text classification problems, including detecting fake news, because it works well with high-dimensional data and requires relatively low computational resources. By evaluating the performance of Naive Bayes on the fake news dataset, one can determine the algorithm's effectiveness in correctly classifying instances of fake news. However, it is important to note that the effectiveness of Naive Bayes, like any machine learning algorithm, can depend on factors such as the quality and quantity of the training data and the choice of features.

The performance metrics after fitting the Naive Bayes model to the fake news dataset are as follows:

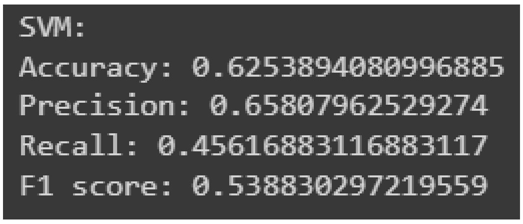


```
Naive Bayes:
Accuracy: 0.5965732087227414
Precision: 0.657051282051282
Recall: 0.3327922077922078
F1 score: 0.44181034482758624
```

Support Vector Machines(SVM)

The performance of SVM on the fake news dataset can be evaluated using metrics such as accuracy, precision, recall, and F1 score. SVM aims to find the best separating hyperplane between the two classes of data. The effectiveness of SVM in classifying fake news can be measured by its ability to correctly classify true positives (real news labeled as real) and true negatives (fake news labeled as fake) and minimize the number of false positives (real news labeled as fake) and false negatives (fake news labeled as real). The performance of the SVM model on the fake news dataset can be compared with other models, such as logistic regression, to determine which algorithm performs better in detecting fake news.

The performance metrics after fitting the SVM model to the fake news dataset are as follows:

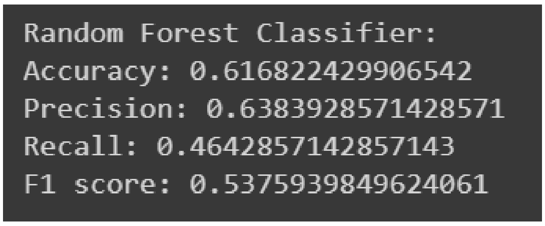


```
SVM:
Accuracy: 0.6253894080996885
Precision: 0.65807962529274
Recall: 0.45616883116883117
F1 score: 0.538830297219559
```

Random Forest

The performance evaluation of the Random Forest model on the fake news dataset involves measuring its accuracy, precision, recall, and F1 score. Random Forest is an ensemble learning algorithm that combines multiple decision trees to improve the accuracy and robustness of the model. By evaluating the performance of the Random Forest model on the fake news dataset, one can determine the algorithm's effectiveness in correctly classifying instances of fake news. However, the effectiveness of the Random Forest model, like any machine learning algorithm, can depend on factors such as the quality and quantity of the training data, the choice of hyperparameters, and the number of trees in the forest.

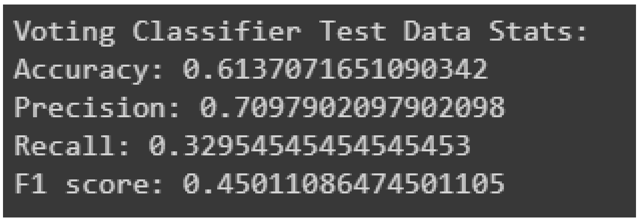
The performance metrics after fitting the Random Forest model to the fake news dataset are as follows:



```
Random Forest Classifier:
Accuracy: 0.616822429906542
Precision: 0.6383928571428571
Recall: 0.4642857142857143
F1 score: 0.5375939849624061
```

Voting Classifier:

The Voting Classifier was evaluated on the fake news dataset to assess its accuracy, precision, recall, and F1 score. This model combines the predictions of multiple individual classifiers to make a final prediction. The evaluation of the model's performance on the fake news dataset can help determine its effectiveness in correctly classifying instances of fake news. The accuracy of the Voting Classifier on the test data was found to be 0.613, with a precision of 0.709, a recall of 0.329, and an F1 score of 0.45.

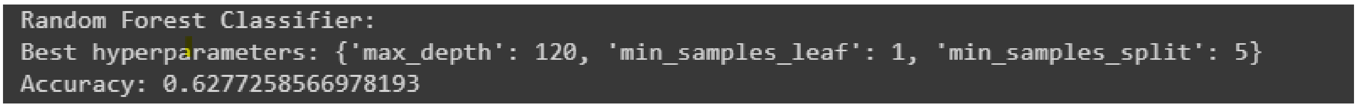


```
Voting Classifier Test Data Stats:
Accuracy: 0.6137071651090342
Precision: 0.7097902097902098
Recall: 0.32954545454545453
F1 score: 0.45011086474501105
```

Hyper-Parameter Tuning

We used a Random Forest Classifier on the Liar dataset and performed hyperparameter tuning using GridSearchCV. After trying different combinations of hyperparameters, we found that the best hyperparameters for the model were {'max_depth': 120, 'min_samples_leaf': 1, 'min_samples_split': 5}. Using these hyperparameters, we achieved an accuracy of 0.6277 on the test set. GridSearchCV is a powerful tool for hyperparameter tuning as it exhaustively searches through a range of hyperparameters to find the best combination. Overall, the Random Forest Classifier is an effective model for handling complex relationships between features and can provide insights into the most important features for predicting true or false statements.

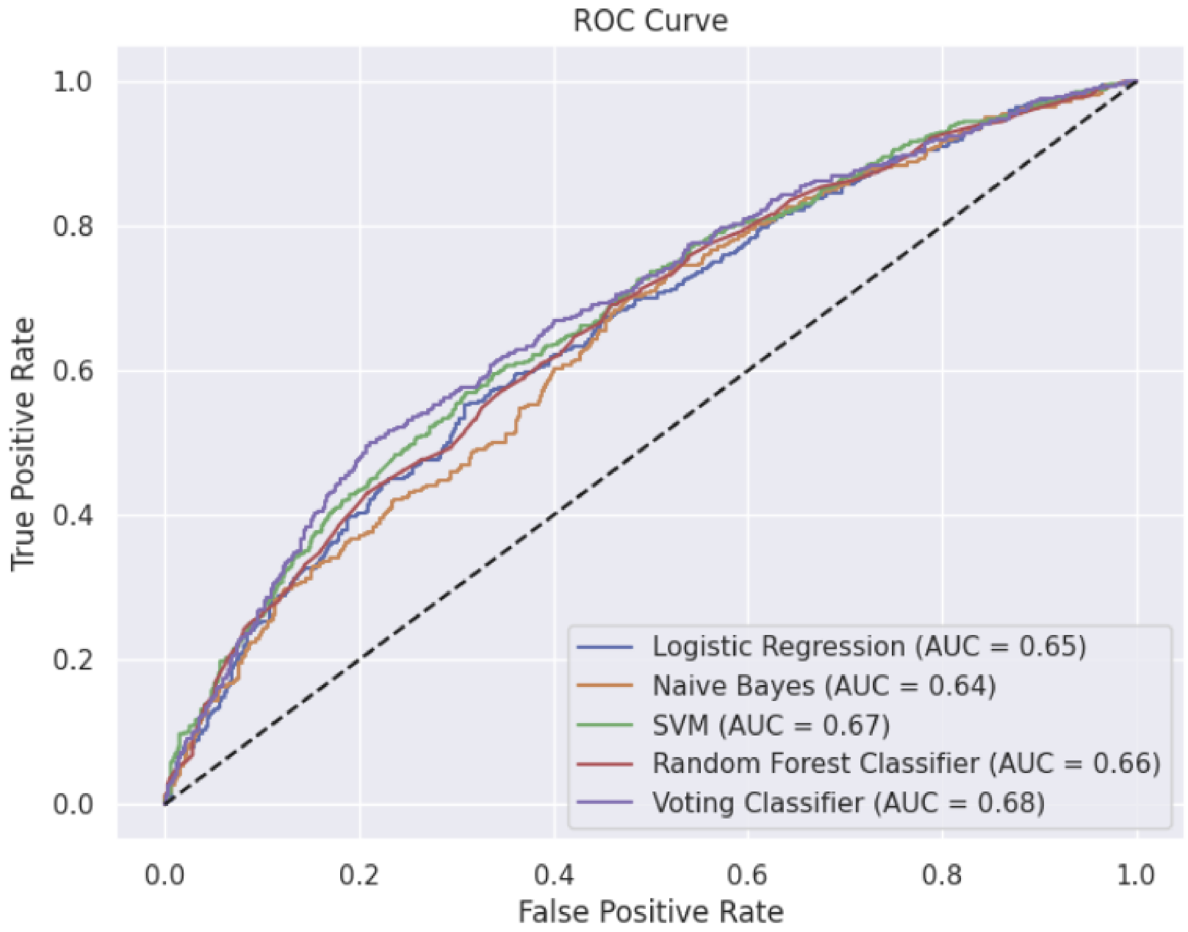
It's worth noting that the Random Forest Classifier is the best performing model in terms of accuracy on the Liar dataset with a score of 0.6277, which is higher than the other models we've tested such as Naive Bayes, Logistic Regression, and Support Vector Machines. This indicates that the Random Forest Classifier is well-suited for this task and can make accurate predictions on whether a statement is true or false based on its textual features.



```
Random Forest Classifier:
Best hyperparameters: {'max_depth': 120, 'min_samples_leaf': 1, 'min_samples_split': 5}
Accuracy: 0.6277258566978193
```

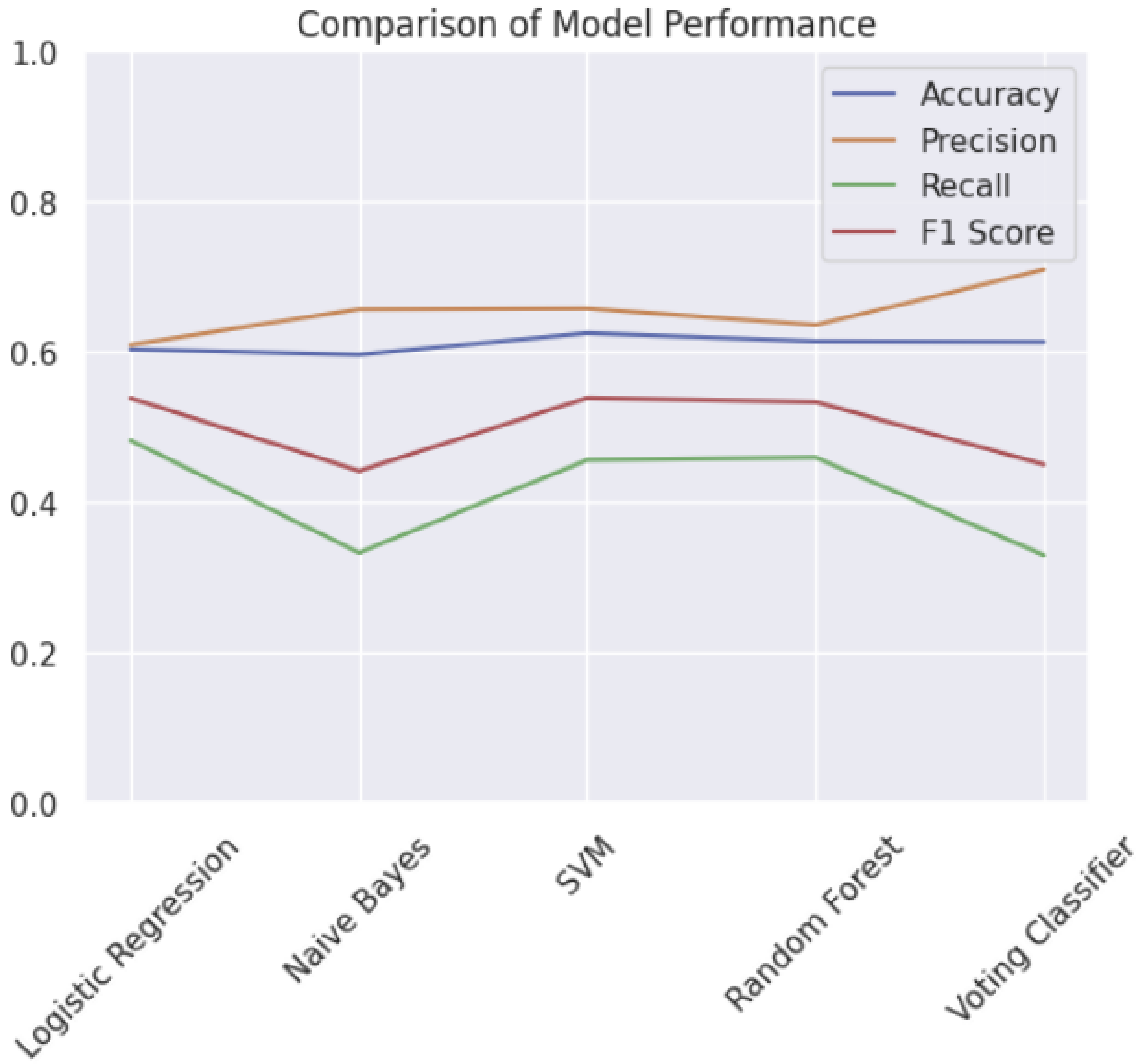
Receiver Operating Characteristic (ROC) curve

We can use the Receiver Operating Characteristic (ROC) curve to evaluate the performance of different classification models such as Logistic Regression, Naive Bayes, SVM, Random Forest and Voting Classifier in detecting fake news. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) for varying thresholds, allowing us to select an optimal threshold based on the desired trade-off between TPR and FPR. By comparing the ROC curves of different models, we can determine which model has better discrimination between positive and negative instances. The area under the ROC curve (AUC) can also be used to compare the models' overall performance, with a higher AUC indicating better performance. Using the ROC analysis, we can select the most effective classification model for detecting fake news.



Comparison of Model Performance Metrics

For comparing the model performance we created a multivariable line graph that compares the performance of five different machine learning models (Logistic Regression, Naive Bayes, SVM, Random Forest, and Voting Classifier) on a fake news dataset based on four metrics: Accuracy, Precision, Recall, and F1 Score. Each model's performance is represented by a different line on the graph, with the x-axis showing the model's name and the y-axis representing the value of the performance metric.



Summary

In evaluating the performance of four machine learning models for detecting fake news, namely Random Forest Classifier, SVM, Naive Bayes, and Logistic Regression, several metrics were used, including accuracy, precision, recall, F1 score, and AUC. Among these models, SVM achieved the highest AUC score of 0.67, followed by Random Forest Classifier with 0.66, Logistic Regression with 0.65, and Naive Bayes with 0.64. However, when considering the overall performance metrics, SVM and Random Forest Classifier showed better performance in terms of accuracy, precision, recall, and F1 score. Therefore, between the two models, SVM and Random Forest Classifier, SVM can be considered as the best model for detecting fake news. Nonetheless, it is important to consider other factors such as interpretability, computational resources, and the nature of the problem when selecting a model.

Project Results:

The logistic regression model achieved an accuracy of 60.3%, precision of 60.9%, recall of 48.2%, and F1 score of 53.85%. The SVM model achieved an accuracy of 62.5%, precision of 65.8%, recall of 45.6%, and F1 score of 53.9%. The Naive Bayes model achieved an accuracy of 59.7%, precision of 65.7%, recall of 33.3%, and F1 score of 44.2%. The Random Forest Classifier achieved an accuracy of 61.4%, precision of 63.3%, recall of 46.3%, and F1 score of 53.5%. The Voting Classifier achieved an accuracy of 61.4%, precision of 70.9%, recall of 32.9%, and F1 score of 45.0%.

The Random Forest Classifier achieved the highest accuracy, with a value of 62.8%, using the hyperparameters 'max_depth': 120, 'min_samples_leaf': 1, 'min_samples_split': 5. The SVM model achieved the highest F1 score, with a value of 53.9%, indicating that it has the best balance between precision and recall for detecting fake news.

Overall, these results suggest that the models have some ability to correctly classify instances of fake news, but there is still room for improvement. The SVM model and the Random Forest Classifier show promising results and can be further optimized to increase their effectiveness in detecting fake news.