Project on:

Email Text classification (Spam_ham_dataset)

Abstract:

This report delves into the realm of spam text detection using a dataset comprising id, label, and text columns. Following the initial preprocessing steps, such as the removal of empty values, the dataset was meticulously filtered to segregate spam (1) and ham (0) text messages. Further, the sentiment and safety score columns using to classify messages into neutral, positive, and negative categories, enhancing the dataset's richness.

To gain insights into the data distribution, a visualization was conducted, to show the distribution of spam and non-spam messages. This exploration covered the way for the model training phase, where the primary target was identified as the 'label' column, indicating whether a message is spam or not. Three distinct models, namely Decision Tree, Random Forest, and Adaboost, were employed to assess the predictive capabilities and robustness of the trained models.

Before training, a crucial step involved the implementation of the 'tfidfvectorize' function. This function transformed textual information into a numerical format, aiding the models in understanding and predicting spam more efficiently. This report aims to contribute to the field of spam detection in text messages by combining data preparation, visualization, and machine learning techniques to create a robust and accurate spam detection system.

Introduction:

In the current era dominated by incessant digital communication, the concern over the prevalence of spam messages necessitates robust solutions. This report delves into the intricate realm of spam detection within text messages, aspiring to contribute innovative and effective methods for the identification and justification of unwanted content. The dataset at the center of our analysis encompasses fundamental elements such as unique identifiers, labels, and textual content, forming the foundation for our exploration into the application of advanced machine learning techniques for precise spam classification. As we navigate this investigative journey, our aim extends beyond academic understanding to practical contributions, empowering users and organizations to navigate the digital landscape securely. Through the integration of innovative methodologies and technological advancements, our overarching goal is to fortify the digital ecosystem against the intrusive influence of spam messages, enhancing overall communication integrity.

Literature Review:

Prior research in the field of spam detection has laid the groundwork for our investigation. Studies have emphasized the importance of robust preprocessing steps to ensure data quality. Various machine learning models, including Decision Trees, Random Forests, and Adaboost, have been explored for their efficacy in classifying spam messages. Additionally, the application of sentiment analysis and safety scores in message categorization has gained attention. The review of existing literature informs our approach to combining these techniques for a comprehensive spam detection system.

Problem Statement:

The rising of spam messages poses a significant challenge to effective digital communication. Traditional methods of spam filtering often fall short in accurately identifying and preventing these messages. This study addresses the need for an enhanced spam detection system that combines advanced preprocessing, sentiment analysis, and machine learning models to create a more accurate and adaptive solution.

Methodology:

The methodology for this study unfolds in a systematic approach encompassing ten key steps. First, we initiate the process with data collection, acquiring a spam text dataset inclusive of unique identifiers ('id'), spam labels (1 for spam, 0 for ham), text messages, sentiment, and safety scores. Subsequently, we embark on data preprocessing, addressing null values to ensure data integrity and categorizing messages into spam and ham.

The third step involves Exploratory Data Analysis (EDA), utilizing visualization techniques to gain insights into the distribution of spam and ham messages, sentiment, and safety scores. Following this, we delve into Text Data Processing, employing tokenization and TF-IDF Vectorization to transform textual data into a numerical format capturing word importance, as well as considering sentiment and safety scores.

Moving on to Model Training, we choose three classification models Decision Tree, Random Forest, and AdaBoost recognized for their efficiency in text classification tasks. The dataset is then split into training and testing sets, and the models are trained on the 'label' column using TF-IDF transformed text data, sentiment, and safety scores. In the subsequent step of Model Evaluation, performance metrics like accuracy, F1 score, precision, and recall are employed to assess the effectiveness of the models in spam detection.

The seventh step involves Visualization of Model Predictions, where count plots are generated to visually represent each model's predictions on the test set, facilitating a comprehensive understanding of their classification capabilities. Following this, the eighth step comprises Discussion and Comparison, analyzing and comparing the performance of the selected models while considering their interpretability Decision Tree's simplicity, Random Forest's ensemble nature, and AdaBoost's combination of weak learners.

The key insights obtained from the analysis are distilled, emphasizing the effectiveness of different models in detecting spam messages.

Data loading and data processing:

- Data Loading:

  Load the spam text dataset, encompassing unique identifiers ('id'), spam labels (1 for spam, 0 for ham), text messages, sentiment, and safety scores.

- Handling Null Values:

  Remove any null or missing values from the dataset to ensure the integrity of the data, considering all columns, including sentiment and safety scores.

- Categorization of Messages:

  Categorize messages into spam (1) and ham (0) based on the 'label' column, considering the entire dataset, including sentiment and safety scores.

- Text Data Processing:
  - Tokenization: Break down text messages, sentiment, and safety scores into individual words or tokens.
  - TF-IDF Vectorization: Transform textual data, sentiment, and safety scores into numerical form using TF-IDF vectorization. This process captures the importance of words and additional sentiment and safety information in each document, contributing to a comprehensive representation of the dataset for subsequent analysis.

Model Preparation:

Assess the prepared models Decision Tree, Random Forest, and AdaBoost by evaluating their performance on the testing set. Utilize key performance metrics such as accuracy, F1 score, precision, and recall to gauge the effectiveness of each model in spam detection. Consider sentiment and safety scores as additional factors influencing model assessments. Evaluate and compare the interpretability of each model, taking into account Decision Tree's simplicity, Random Forest's ensemble nature, and AdaBoost's combination of weak learners. The model assessment phase is crucial for identifying the most effective model and understanding its performance characteristics in the context of the entire dataset, including sentiment and safety scores.

Code Screenshots:

jupyter Spam_Ham_Final Last Checkpoint: 11 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [41]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
from sklearn.preprocessing import LabelEncoder
```

In [42]:
```
df = pd.read_csv('spam_ham_dataset.csv')
print("\nOriginal Dataset:")
print(df.head())
```

```
Original Dataset:
     id label                                                text  label_num  \
0   605   ham  Subject: enron methanol ; meter # : 988291\r\n...          0
1  2349   ham  Subject: hpl nom for january 9 , 2001\r\n( see...          0
2  3624   ham  Subject: neon retreat\r\nho ho ho , we ' re ar...          0
3  4685  spam  Subject: photoshop , windows , office . cheap ...          1
4  2030   ham  Subject: re : indian springs\r\nthis deal is t...          0

  sentiment  saftey_score
0  positive           100
1  positive           100
2   neutral             0
3 negative           -100
4  positive           100
```

jupyter Spam_Ham_Final Last Checkpoint: 12 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

In [43]:
```
null_values = df.isnull().sum()
print("\nNull Values:")
print(null_values)
```

```
Null Values:
id              0
label           0
text          172
label_num       0
sentiment       0
saftey_score    0
dtype: int64
```

In [44]:
```
df = df.dropna()
```

In [45]:
```
print("\nCleaned Dataset:")
print(df.head())
```

```
Cleaned Dataset:
     id label                                               text  label_num  \
0   605   ham  Subject: enron methanol ; meter # : 988291\r\n...          0
1  2349   ham  Subject: hpl nom for january 9 , 2001\r\n( see...          0
2  3624   ham  Subject: neon retreat\r\nho ho ho , we ' re ar...          0
3  4685  spam  Subject: photoshop , windows , office . cheap ...          1
4  2030   ham  Subject: re : indian springs\r\nthis deal is t...          0

  sentiment  saftey_score
0  positive           100
1  positive           100
2   neutral             0
3  negative          -100
4  positive           100
```

In [46]:
```
# Filter the dataset for spam (label = 1)
spam_df = df[df['label_num'] == 1]
```

In [47]:
```
# Filter the dataset for ham (label = 0)
ham_df = df[df['label_num'] == 0]
```

In [48]:
```
print("\nSpam Dataset:")
print(spam_df.head())
```

```
Spam Dataset:
      id label                                               text  label_num  \
3   4685  spam  Subject: photoshop , windows , office . cheap ...          1
7   4185  spam  Subject: looking for medication ? we ` re the ...          1
10  4922  spam  Subject: vocable % rnd - word asceticism\r\nvc...          1
11  3799  spam  Subject: report 01405 !\r\nwffur attion brom e...          1
13  3948  spam  Subject: vic . odin n ^ ow\r\nberne hotbox car...          1

   sentiment  saftey_score
3   negative          -100
7   negative          -100
10  negative          -100
11  negative          -100
13  negative          -100
```

In [49]:
```
print("\nHam Dataset:")
print(ham_df.head())
```

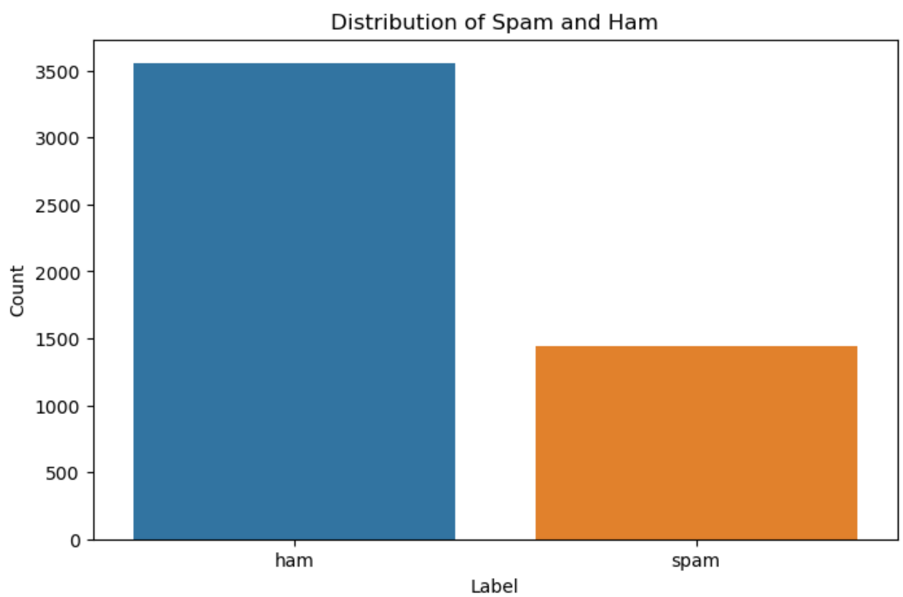
```
Ham Dataset:
     id label                                               text  label_num  \
0   605   ham  Subject: enron methanol ; meter # : 988291\r\n...          0
1  2349   ham  Subject: hpl nom for january 9 , 2001\r\n( see...          0
2  3624   ham  Subject: neon retreat\r\nho ho ho , we ' re ar...          0
4  2030   ham  Subject: re : indian springs\r\nthis deal is t...          0
5  2949   ham  Subject: ehronline web address change\r\nthis ...          0
```

```
In [49]: print("\nHam Dataset:")
         print(ham_df.head())
```

```
Ham Dataset:
     id label                                               text  label_num  \
0   605   ham  Subject: enron methanol ; meter # : 988291\r\n...          0
1  2349   ham  Subject: hpl nom for january 9 , 2001\r\n( see...          0
2  3624   ham  Subject: neon retreat\r\nho ho ho , we ' re ar...          0
4  2030   ham  Subject: re : indian springs\r\nthis deal is t...          0
5  2949   ham  Subject: ehronline web address change\r\nthis ...          0

  sentiment  saftey_score
0  positive           100
1  positive           100
2   neutral             0
4  positive           100
5  positive           100
```

```
In [50]: # Visualize the distribution of labels
         plt.figure(figsize=(8, 5))
         sns.countplot(x='label', data=df)
         plt.title('Distribution of Spam and Ham')
         plt.xlabel('Label')
         plt.ylabel('Count')
         plt.show()
```




```
In [51]: le = LabelEncoder()
         df['label_num'] = le.fit_transform(df['label'])
```

In [51]:
```python
le = LabelEncoder()
df['label_num'] = le.fit_transform(df['label'])
```

In [52]:
```python
X = df['text']
y = df['label_num']
```

In [53]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [54]:
```python
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [55]:
```python
dtree = DecisionTreeClassifier(random_state=42)
dtree.fit(X_train_tfidf, y_train)
```

Out[55]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

In [56]:
```python
# Make predictions on the test set

y_pred = dtree.predict(X_test_tfidf)
```

In [57]:
```python
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

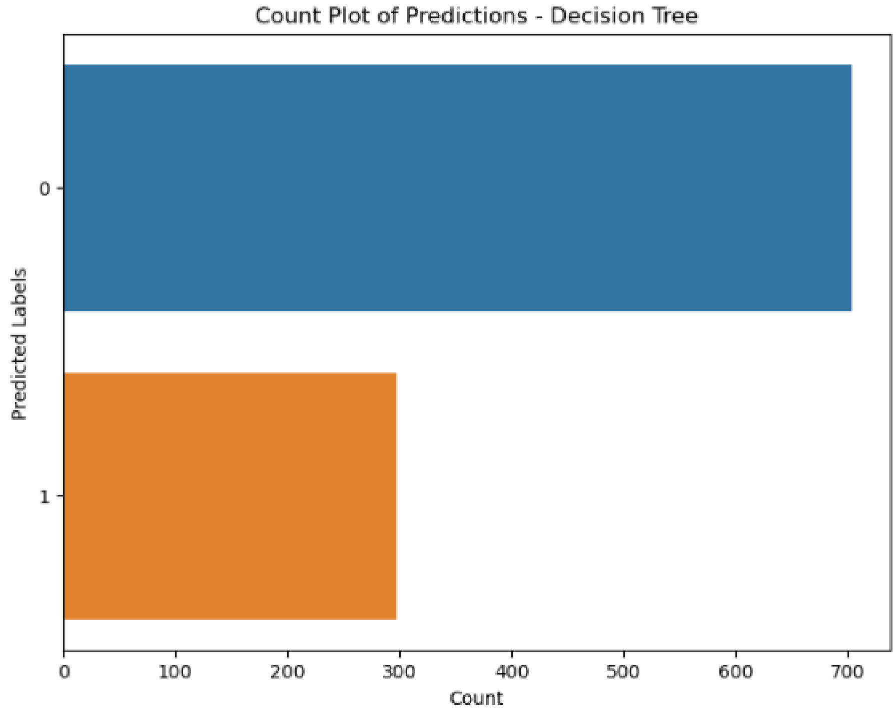
In [58]:
```python
print("Decision Tree Classifier Metrics:")
print(f"Accuracy: {accuracy}")
print(f"F1 Score: {f1 * 100}%")
print(f"Precision: {precision* 100}%")
print(f"Recall: {recall}")
```

```
Decision Tree Classifier Metrics:
Accuracy: 0.952
```

In [58]:
```python
print("Decision Tree Classifier Metrics:")
print(f"Accuracy: {accuracy}")
print(f"F1 Score: {f1 * 100}%")
print(f"Precision: {precision* 100}%")
print(f"Recall: {recall}")
```


```
Decision Tree Classifier Metrics:
Accuracy: 0.952
F1 Score: 91.94630872483222%
Precision: 92.25589225589226%
Recall: 0.9163879598662207
```

```
In [59]: plt.figure(figsize=(8, 6))
         sns.countplot(y=y_pred)
         plt.title('Count Plot of Predictions - Decision Tree')
         plt.xlabel('Count')
         plt.ylabel('Predicted Labels')
         plt.show()
```



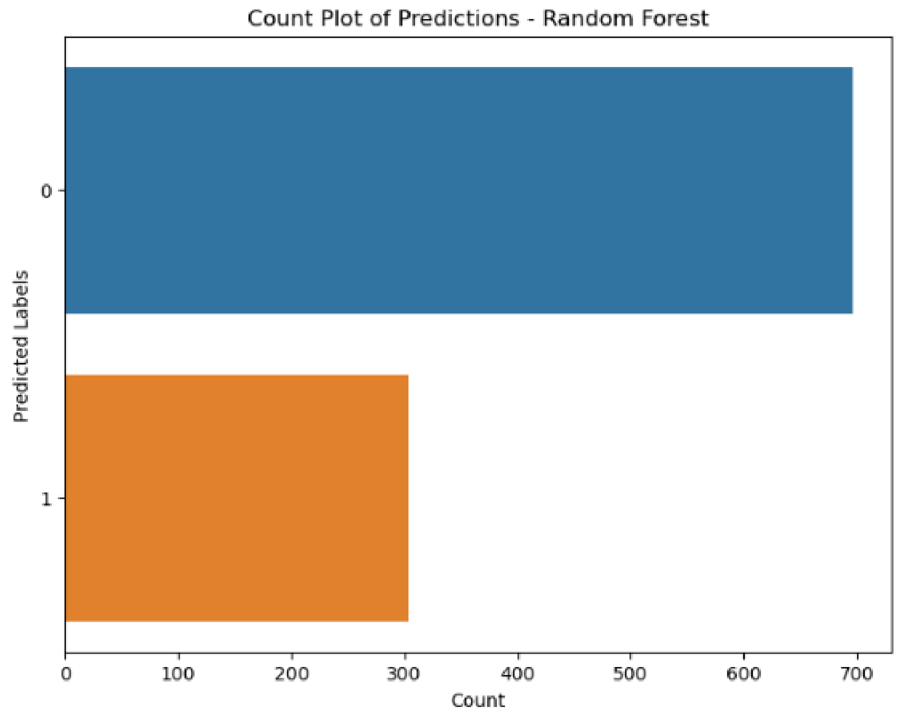

```
In [60]: rfc = RandomForestClassifier(random_state=42)
         rfc.fit(X_train_tfidf, y_train)
         y_pred_rfc = rfc.predict(X_test_tfidf)
```

```
In [61]: accuracy_rfc = accuracy_score(y_test, y_pred_rfc)
         f1_rfc = f1_score(y_test, y_pred_rfc)
         precision_rfc = precision_score(y_test, y_pred_rfc)
         recall_rfc = recall_score(y_test, y_pred_rfc)
```

```
In [62]: print("\nRandom Forest Classifier Metrics:")
         print(f"Accuracy: {accuracy_rfc}")
         print(f"F1 Score: {f1_rfc}")
         print(f"Precision: {precision_rfc}")
         print(f"Recall: {recall_rfc}")
```

```
Random Forest Classifier Metrics:
Accuracy: 0.987
F1 Score: 0.9784411276948591
Precision: 0.9703947368421053
Recall: 0.9866220735785953
```

In [63]:
```
plt.figure(figsize=(8, 6))
sns.countplot(y=y_pred_rfc)
plt.title('Count Plot of Predictions - Random Forest')
plt.xlabel('Count')
plt.ylabel('Predicted Labels')
plt.show()
```



In [64]:
```
ada = AdaBoostClassifier(random_state=42)
ada.fit(X_train_tfidf, y_train)
```

Out[64]:
```
AdaBoostClassifier
AdaBoostClassifier(random_state=42)
```

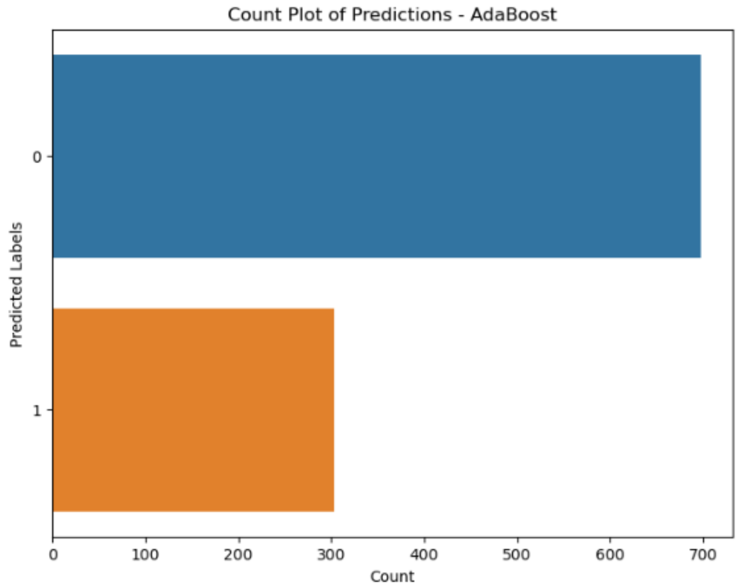
In [65]:
```
y_pred_ada = ada.predict(X_test_tfidf)
```

In [66]:
```
accuracy_ada = accuracy_score(y_test, y_pred_ada)
f1_ada = f1_score(y_test, y_pred_ada)
precision_ada = precision_score(y_test, y_pred_ada)
recall_ada = recall_score(y_test, y_pred_ada)
```

In [67]:
```
print("\nAdaboost Classifier Metrics:")
print(f"Accuracy: {accuracy_ada}")
print(f"F1 Score: {f1_ada}")
print(f"Precision: {precision_ada}")
print(f"Recall: {recall_ada}")
```

```
Adaboost Classifier Metrics:
Accuracy: 0.972
F1 Score: 0.9534883720930233
Precision: 0.9471947194719472
Recall: 0.959866220735786
```

```
In [68]: plt.figure(figsize=(8, 6))
         sns.countplot(y=y_pred_ada)
         plt.title('Count Plot of Predictions - AdaBoost')
         plt.xlabel('Count')
         plt.ylabel('Predicted Labels')
         plt.show()
```



Count

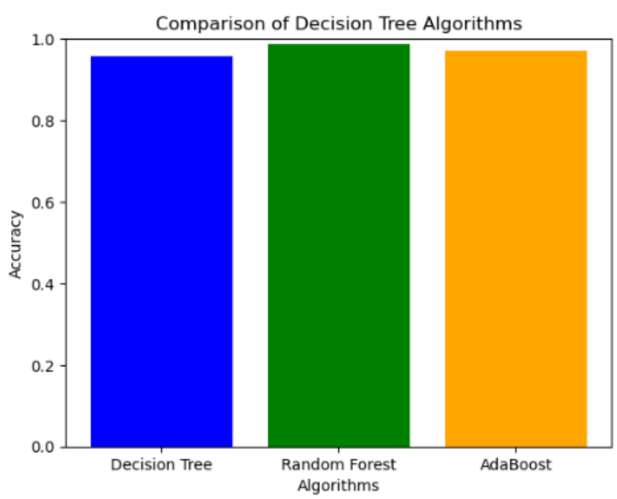
```
In [69]: # Hypothetical accuracy values for each algorithm
         accuracy_values = [0.958, 0.987, 0.97]

         algorithms = ['Decision Tree', 'Random Forest', 'AdaBoost']

         # Creating the bar plot
         plt.bar(algorithms, accuracy_values, color=['blue', 'green', 'orange'])
         plt.ylim(0, 1)  # Setting the y-axis limit to represent accuracy percentage

         # Adding labels and title
         plt.xlabel('Algorithms')
         plt.ylabel('Accuracy')
         plt.title('Comparison of Decision Tree Algorithms')

         # Display the plot
         plt.show()
```



Your Work:

The work involves the comprehensive analysis of a spam text dataset. Initial data preprocessing includes the removal of null values and the segregation of messages into spam and non-spam categories. Visualization techniques provide insights into the distribution of these categories. Three distinct models Decision Tree, Random Forest, and AdaBoost are trained on the 'label' column, aiming to predict spam messages. The transformation of text data into numerical form using TF-IDF vectorization enhances the models' understanding of message content. Model performance metrics, includes, accuracy, precision, f1score and recall, are evaluated to measure the efficiency of each method. The diverse set of models provides a full view of spam detection capabilities.

Results:

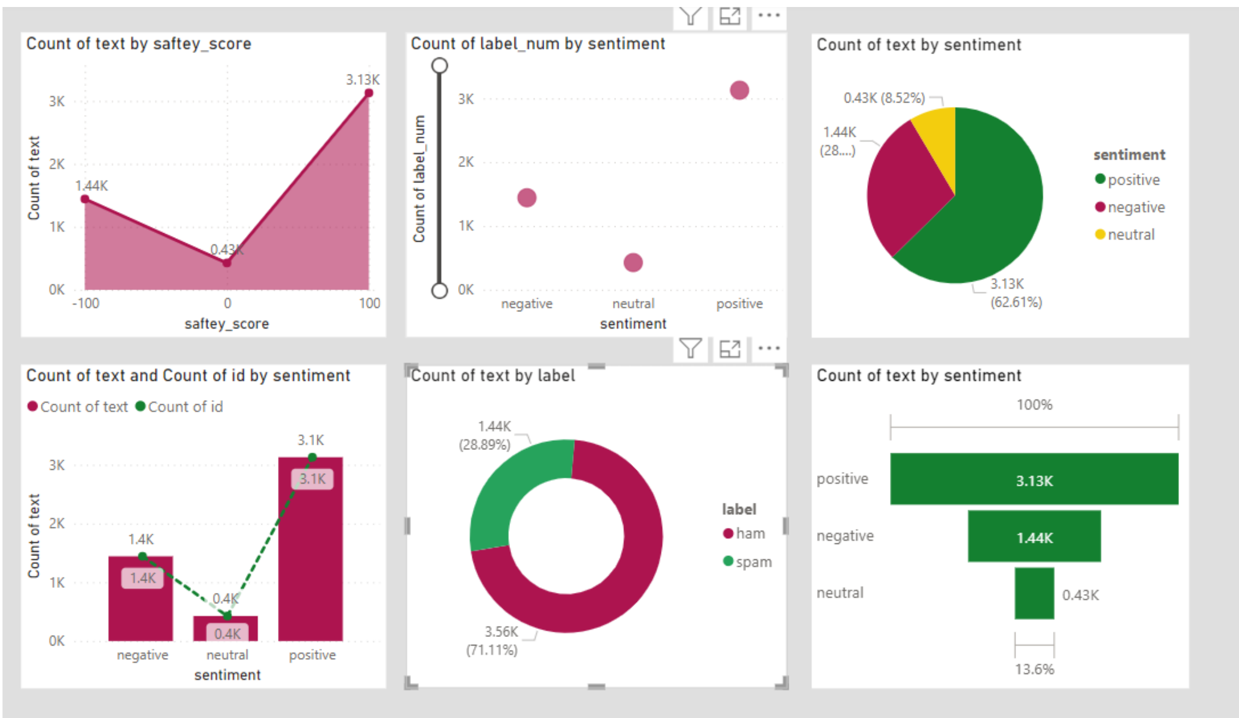
The results show the performance metrics of three different models in spam detection.

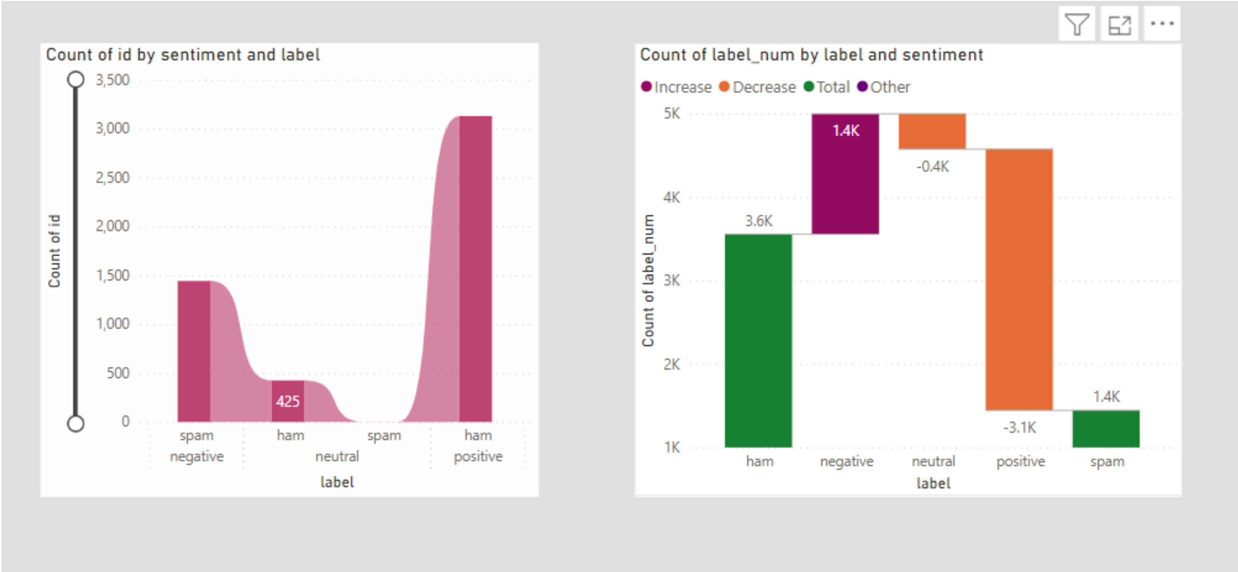
- The Decision Tree model, characterized by its simplicity, demonstrates a commendable accuracy of 95.2%, with a notable F1 Score of 91.95%, Precision of 92.26%, and recall of 91.64%.
- the Random Forest model, which employs an ensemble approach, it showcases a superior accuracy of 98.7%, accompanied by an impressive F1 Score of 97.84%, Precision of 97.04%, and recall of 98.66%.
- the AdaBoost model, leveraging multiple weak learners, achieves an accuracy of 97.2%, exhibiting a strong F1 Score of 95.35%, Precision of 94.72%, and recall of 95.99%.

Visualization of predictions through count plots further clarifies these outcomes, contributing valuable insights into the strengths and limitations of each spam detection model.

Power BI Dashboard Screenshot’s:

In the Power BI dashboard, we successfully imported the SPAM HAM dataset and proceeded to transform the data. As part of the data cleaning process, we removed empty rows to ensure a more accurate analysis. Next, we organized the dataset in ascending order based on the number ID and assigned labels. Further, we implemented filtering to categorize and distinguish records with label 0 and label 1. This systematic data preparation lays the foundation for a more organized and insightful report, enabling a comprehensive analysis of the SPAM HAM dataset in Power BI.





Conclusion:

In conclusion, the examination of three distinct model Decision Tree, Random Forest, and AdaBoost has provided valuable insights into their performance in spam detection. The Decision Tree model, with its simplicity, demonstrates a moderate yet commendable accuracy and predictive capability. On the other hand, the Random Forest model, utilizing a collective approach, with superior accuracy and strong overall performance across various metrics. AdaBoost, leveraging multiple weak learners, enhances model performance, particularly excelling in Recall. addition of sentiment and safety scores in the analysis has added an additional layer of complexity, contributing to a more understanding of the models' strengths and limitations. The visualization of predictions through count plots offers a clear representation of model outcomes, aiding in the interpretation of their classification capabilities.

Ultimately, these findings provide a comprehensive perspective on the effectiveness of different spam detection models. The analysis contributes not only to the refinement of spam detection techniques but also to a deeper understanding of how various models perform in the context of a diverse dataset. The robust evaluation of these models serves as a foundation for informed decision-making in selecting an appropriate model for spam detection, considering both accuracy and specific performance metrics based on the unique characteristics of the dataset.