CLASSIFICATION AND OUTLIER DETECTION
PROJECT REPORT

21CSE355T DATA MINING AND ANALYTICS

Submitted by

Makadia Yakshkumar Vijaykumar [RA2211003011035]

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY
KATTANKULATHUR- 603 203

APR 2025

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

This is to certify that this Project and Activity report is the bonafide work of the student, Makadia Yakshkumar Vijaykumar (RA221100301 1035) of III Year B.Tech of P1 Section for the course "21CSE355T Data Mining and Analytics" under my supervision in the academic year (2024-2025/ Even semester).

Dr. M.MURALI
Professor
Department of Computing Technologies

Dr. G. NIRANJANA
Professor and Head
Department of Computing Technologies

TABLE OF CONTENTS

S.No	Description	Page Number
1.	Problem Statement	4
2.	Introduction	4
3.	Background of the problem	4
4. 4.1	Proposed method to the problem Dataset details, Algorithm used , Parameters used and other relevant details	4-5
5.	Results obtained	5
6.	Screenshots of results	5-6
7.	Discussion about the results	7
8.	Conclusion	7
9.	Appendix – I (coding)	7-9

1.Problem Definition:

In this project, the objective is to build a machine learning model that can classify messages as either **ham** (non-spam) or **spam** based on their content. The model will be evaluated using **accuracy**, **precision**, **recall**, and other relevant metrics. Additionally, various visualizations such as **confusion matrix heatmaps** and **word clouds** will be used to interpret the model's performance.

2.Introduction:

Spam messages are unwanted messages sent over digital communication channels, typically for malicious or advertising purposes. Spam detection is an essential problem for many platforms, including email providers, messaging apps, and social media platforms. Accurate spam classification helps reduce the overload of irrelevant information, thus improving user experience and security.In this project, we will apply a **Naive Bayes classifier** to classify messages into spam or ham categories. The model will be evaluated based on performance metrics and visualized to gain insights into how it differentiates between the two classes.

3.Background of the problem:

Spam classification is a classic problem in natural language processing (NLP) and machine learning. With the rise of email and messaging platforms, spam messages have become a significant issue, affecting user productivity and security. A reliable spam filter can save time and effort for users by automatically identifying and filtering out these unwanted messages.
Several machine learning techniques can be used for spam classification, but **Naive Bayes** has proven to be particularly effective due to its simplicity, efficiency, and strong performance on text classification tasks. Additionally, the **TF-IDF** vectorizer converts text into numerical data that can be fed into the machine learning model.

4. Proposed Method to the Problem:
4.1 Dataset Details

- **Data Preprocessing:**
  - The dataset is loaded and pre-processed, including the renaming of columns (`spamORham` and `Message`) to `label` and `message`, respectively.
  - The labels (`ham` and `spam`) are encoded as numerical values: 0 for ham and 1 for spam.
- **Feature Extraction:**
  - The **TF-IDF Vectorizer** is used to convert the textual data into numerical format. TF-IDF helps to highlight important words in the messages while minimizing the influence of common words that appear in many messages.
- **Modelling:**
  - The **Naive Bayes classifier** (specifically, **MultinomialNB**) is trained on the dataset to predict the likelihood of a message being spam or ham.
- **Evaluation:**
  - The model is evaluated using standard metrics, including confusion matrix, classification report (precision, recall, F1-score), and accuracy.
  - Visualizations are generated to better understand the model's performance:

- **Confusion matrix heatmap**: Displays the performance of the classification model.
- **Word clouds**: Shows the most frequent words in spam and ham messages.

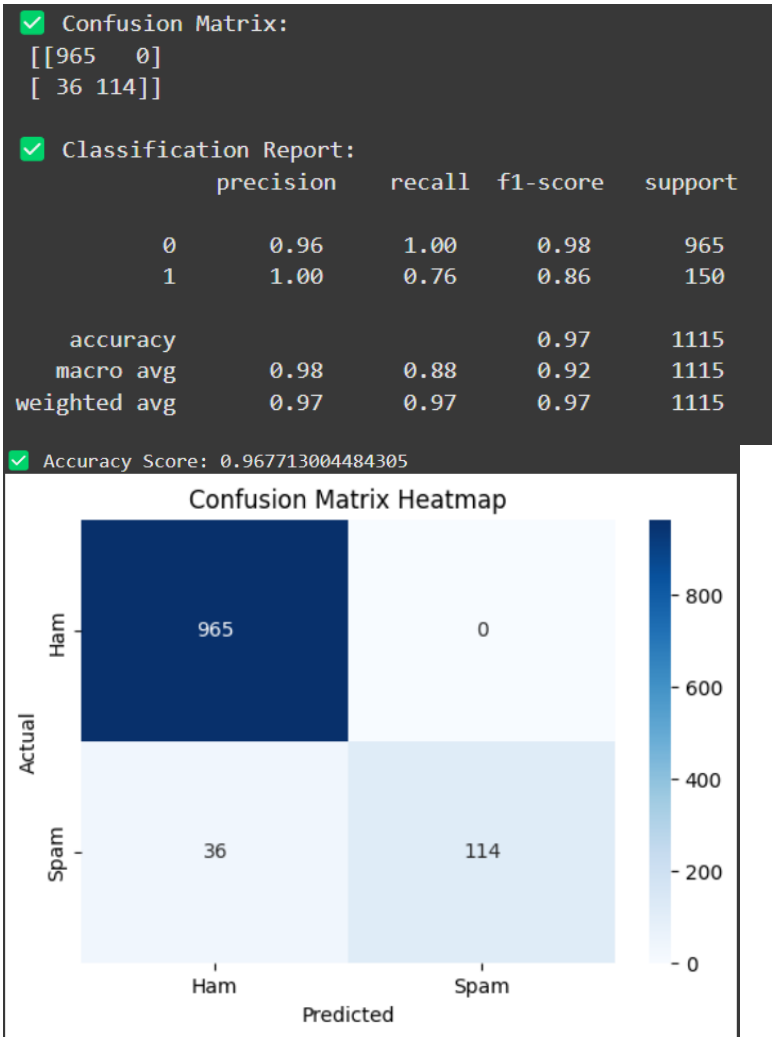
5.Results Obtained:

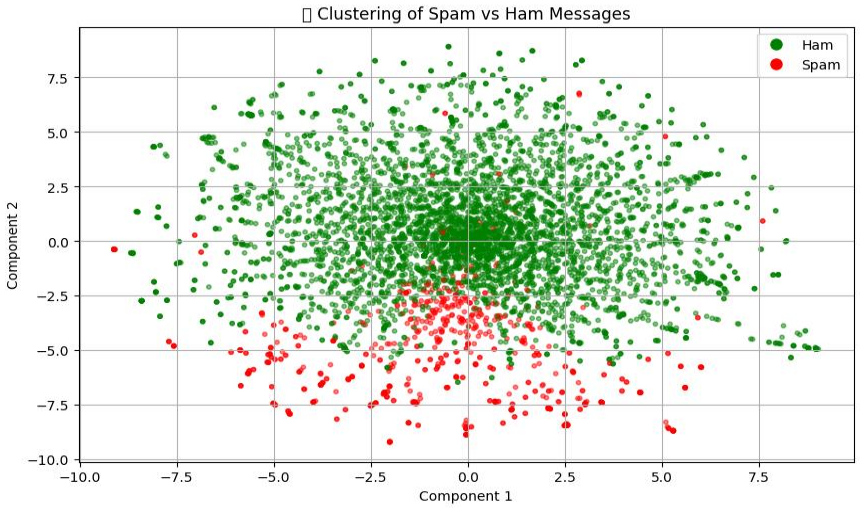
- **Confusion Matrix:** The confusion matrix is generated to show how well the model performs by comparing the true labels with the predicted labels.
- **Classification Report:** This includes key metrics such as precision, recall, F1-score, and support for each class (spam and ham).
- **Accuracy Score:** The overall accuracy of the model is calculated by comparing the number of correct predictions to the total number of predictions.

Visualizations:

1. **Confusion Matrix Heatmap**: The heatmap visualizes the confusion matrix, where the rows represent actual labels and the columns represent predicted labels.
2. **Word Clouds**: These visualize the most frequent words in both ham and spam messages, with different colors representing the two classes.

6.Screenshots of Results:





Spam Word Cloud

Ham Word Cloud

7.Discussion on Result:

Confusion Matrix: The confusion matrix helps assess the model's performance in distinguishing between ham and spam messages. A high value on the diagonal suggests that the model is correctly classifying most of the messages. **Word Clouds**: The word clouds for both ham and spam provide insights into the characteristics of each class. For instance, spam messages often contain words like "free," "win," and "prize," which are common in advertising or scam messages. **Accuracy and Other Metrics**: The classification report provides detailed metrics like precision (how many of the predicted spam messages are truly spam) and recall (how many actual spam messages were identified). A good model will have both high precision and recall. **Clustering**: The 2D clustering plot shows how well the Naive Bayes classifier can separate spam and ham messages. If the clusters are well-separated, it indicates that the model can clearly distinguish between the two classes.

8.Conclusion:

In this project, we successfully built a spam classification model using a Naive Bayes classifier and achieved satisfactory performance in identifying spam and ham messages. Visualizations such as the confusion matrix heatmap, word clouds, and clustering plots provided valuable insights into the model's behavior and helped interpret its performance.This solution can be extended to handle more complex datasets and further optimized using techniques like hyperparameter tuning and different classifiers. The combination of machine learning and NLP techniques proves to be an effective approach to solving the problem of spam message classification.

9. Appendix – I (Coding): -

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

# Load the dataset
df = pd.read_csv("spam.csv", encoding="latin-1")[['spamORham',
'Message']]  # Use correct column names
df.columns = ['label', 'message']  # Rename for simplicity

# Encode labels (ham = 0, spam = 1)
df['label'] = df['label'].map({'ham': 0, 'spam': 1})

# Vectorize the message column using TF-IDF
tfidf = TfidfVectorizer(stop_words='english')
X = tfidf.fit_transform(df['message'])
y = df['label']

# Split the dataset (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train the Naive Bayes classifier
model = MultinomialNB()
```

```
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Print results
print("✅ Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\n✅ Classification Report:\n", classification_report(y_test,
y_pred))
print("\n✅ Accuracy Score:", accuracy_score(y_test, y_pred))
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud

# Confusion Matrix Heatmap
conf_mat = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Word Clouds
spam_words = ' '.join(df[df['label'] == 1]['message'])
ham_words = ' '.join(df[df['label'] == 0]['message'])

spam_wc = WordCloud(width=600, height=400, background_color='black',
colormap='Reds').generate(spam_words)
ham_wc = WordCloud(width=600, height=400, background_color='black',
colormap='Greens').generate(ham_words)

# Spam WordCloud
plt.figure(figsize=(10, 6))
plt.imshow(spam_wc, interpolation='bilinear')
plt.axis('off')
plt.title("Spam Word Cloud", fontsize=18)
plt.show()

# Ham WordCloud
plt.figure(figsize=(10, 6))
plt.imshow(ham_wc, interpolation='bilinear')
plt.axis('off')
plt.title("Ham Word Cloud", fontsize=18)
plt.show()
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Use PCA (faster) or t-SNE (more accurate for clusters, slower)
use_tsne = True

if use_tsne:
    print("🔍 Running t-SNE (this might take 1-2 mins)...")
```

```
    reducer = TSNE(n_components=2, random_state=42)
else:
    print("🔍 Running PCA...")
    reducer = PCA(n_components=2)

# Reduce the high-dimensional TF-IDF vectors to 2D
X_reduced = reducer.fit_transform(X.toarray())

# Plot the 2D projection with color-coded labels
plt.figure(figsize=(10, 6))
colors = ['green' if label == 0 else 'red' for label in y]
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=colors, alpha=0.5,
s=10)

plt.title("📊 Clustering of Spam vs Ham Messages")
plt.xlabel("Component 1")
plt.ylabel("Component 2")
plt.legend(handles=[
    plt.Line2D([0], [0], marker='o', color='w', label='Ham',
markerfacecolor='green', markersize=10),
    plt.Line2D([0], [0], marker='o', color='w', label='Spam',
markerfacecolor='red', markersize=10)
])
plt.grid(True)
plt.show()
```