DATA ANALYSIS USING PYTHON

A Technical project Report

in partial fulfilment of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

By

BANDI SREEJA **2203A52005**

Under the guidance of

Dr. Ramesh Dadi

Assistant Professor, School of CS&AI

Submitted to

COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE SR UNIVERSITY, ANANTHASAGAR, WARANGAL APRIL 2025.

TABLE OF CONTENTS

1.ONLINE RETAIL DATA ANALYSIS

1. Introduction

This project focuses on analyzing online retail transaction data to uncover meaningful insights about customer behavior and sales patterns. By exploring a dataset containing purchase records from an online retailer, we aim to understand trends in product demand, customer purchasing habits, and overall business performance. Through data cleaning, visualization, and analysis, the project highlights key metrics such as most sold products, top customers, and sales distribution across different countries. This analysis provides valuable insights that can support better decision-making in online retail and e-commerce environments.

2. Dataset Overview

The dataset contains **10,000 records** of online retail transactions, each representing a product purchase. There are **8 columns** capturing various details about each transaction.

The dataset includes transactions from various countries, with a significant portion coming from the **United Kingdom**.

2.1 Features Description

- **InvoiceNo**: A unique identifier for each transaction.
- **StockCode**: A product/item code.
- **Description**: The name or description of the product.
- **Quantity**: Number of units purchased.
- **InvoiceDate**: The date and time when the transaction occurred.
- **UnitPrice**: The price per unit of the product.
- **CustomerID**: A unique identifier for the customer (note: ~36% of entries are missing this).
- **Country**: The customer's country of residence.

3. Data Preprocessing and Data Cleaning

In this project, the dataset underwent a series of essential data cleaning and preprocessing steps to ensure quality and consistency for analysis. Initially, rows with missing CustomerID values were removed, while missing product Description entries were filled with a placeholder value . The InvoiceDate column was converted to proper datetime format to support time-based analysis. Duplicate records were eliminated, and transactions with non-positive Quantity or UnitPrice were filtered out to avoid invalid or erroneous entries. The dataset index was reset after cleaning, and new columns such as TotalRevenue were calculated to enrich the analysis. These steps helped prepare the data for meaningful insights and visualization.

4. Exploratory Data Analysis (EDA)

The project includes a well-rounded Exploratory Data Analysis (EDA) phase to better understand the online retail dataset.

Descriptive Statistics:

- Basic statistical summaries (mean, min, max, std, etc.) were generated for numerical columns like Quantity and UnitPrice.

Missing Value Analysis:

- Identified and handled missing values in CustomerID and Description columns.

Date-Time Feature Engineering:

- The InvoiceDate column was converted to datetime format, and new features like Date, Week, and YearMonth were extracted for temporal analysis.

Revenue Calculation:

- A new column TotalRevenue was created by multiplying Quantity and UnitPrice to analyze sales value per transaction.

Time Series Analysis:

- Daily, weekly, and monthly revenue trends were visualized using line plots to observe patterns and seasonality in sales.

Sales Distribution (implied from feature creation):

- Grouping and summarization of sales data provided insights into how purchases vary over time.

5. Model Building and Models Used

- Histogram
- Box Plot

- Cleaning and Processing
- Logistic Regression
- Random Forest Classifier
- K-Nearest Neighbors (KNN)
- Evaluation using Confusion Matrix
- Confusion Matrix
- Kurtosis

Each model was trained and evaluated using accuracy.

6. Model Implementation

Several machine learning algorithms were implemented to predict online retail trends:

7. HISTOGRAM ANALYSIS

The histograms provide a visual representation of the distribution of three key numerical columns in the online retail dataset: Quantity, UnitPrice, and CustomerID. The Quantity histogram reveals that most transactions involve the purchase of a small number of items, with a noticeable peak at lower positive values, although a few negative and unusually high quantities are also present, possibly indicating returns or data anomalies. The UnitPrice distribution is right-skewed, showing that most products are low-priced, with a few high-priced items contributing to a long tail. The CustomerID histogram appears fairly uniform, indicating that the dataset includes a wide and balanced range of customer records. These histograms help in identifying data irregularities and understanding the overall structure of the dataset.

RESULTS:

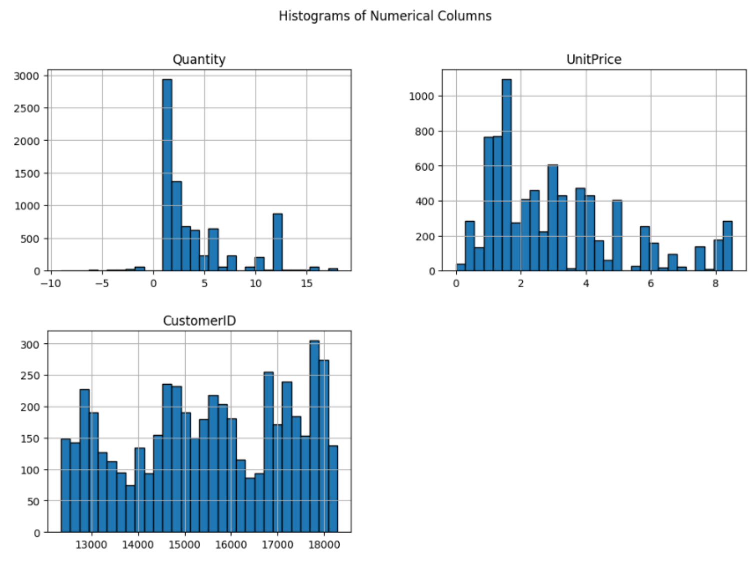


Fig.1.1 Histogram

Observations from Histograms:

1. Quantity

- Most values are concentrated between 1 and 5 units, with a peak around 1.
- There are some higher spikes around 10 and 12.
- A few entries show negative quantities, which might indicate returns or errors.
- The distribution is right-skewed, with the majority of values on the lower end**.**

2. Unit Price

- The unit prices are also right-skewed.
- A significant number of items are priced between 1 and 3 units.
- There are some expensive items priced above 5 units, although they are rare.
- A small number of prices close to 0 may represent free samples or data entry issues.

3. Customer ID

- The distribution is roughly uniform, indicating a fairly even spread of transactions across customers.
- Some IDs appear more frequently, suggesting those customers may be repeat buyers.
- There are no obvious outliers in CustomerID, but the pattern does suggest clusters of active customers.

8. Box Plot

A **Box Plot** (also known as a **box-and-whisker plot**) is a graphical representation used to visualize the **distribution, central tendency, and variability** of numerical data. It provides insights into the **spread and skewness** of the dataset, and most importantly, it helps identify **outliers**.

1. Quantity

- Identify transactions with unusually high or low quantities.
- Negative or zero values may appear as **outliers** (especially return transactions).
- Helps decide whether to remove or treat extreme cases.

2. Unit Price

- Detect items with unusually high pricing (possibly data entry errors or luxury items).
- A right-skewed distribution is common — box plot visually confirms this.
- Outliers might indicate promotional or wrongly priced items.

3. Customer ID

- While numeric, it doesn't represent a quantity — box plot isn't meaningful here unless summarizing aggregated behavior (e.g., total spend per customer).

RESULT:

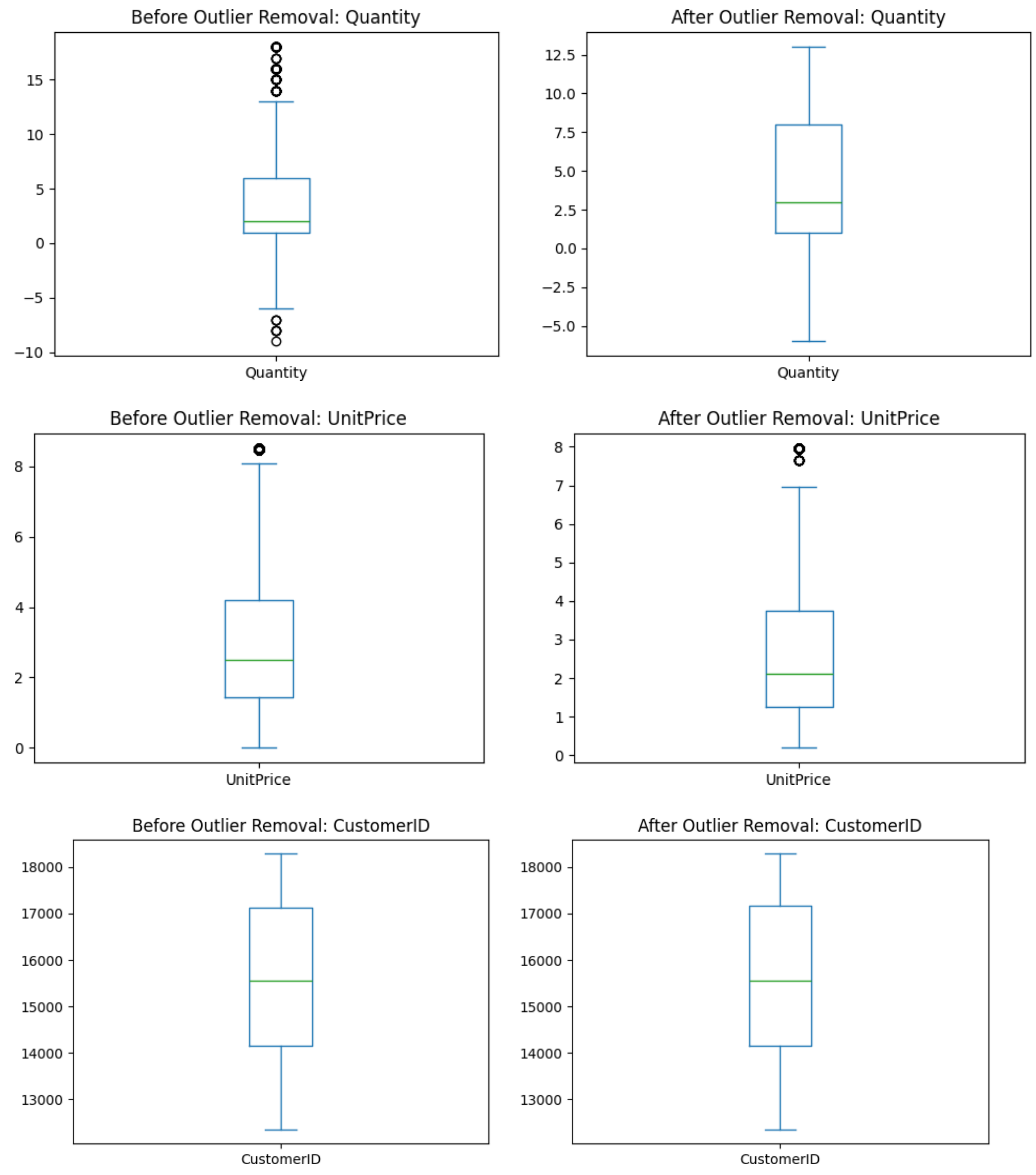


Fig.1.2 Box Plot

9. LOGISTIC REGRESSION

Logistic Regression is applied to predict a binary target variable, likely related to transaction validity or customer classification. Features were normalized using StandardScaler, which is essential for logistic regression to perform well. Data is split into training and testing sets with train_test_split() and trained the model using fit(). Classification_report and confusion_matrix are used for evaluation, showing awareness of model accuracy, precision, recall, and more.

RESULT:

Logistic Regression Accuracy: 0.9829488465396189

```
Precision: 1.0000
Recall: 0.9592
F1 Score: 0.9792
ROC AUC Score: 0.9796

Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       580
           1       1.00      0.96      0.98       417

    accuracy                           0.98       997
   macro avg       0.99      0.98      0.98       997
weighted avg       0.98      0.98      0.98       997
```

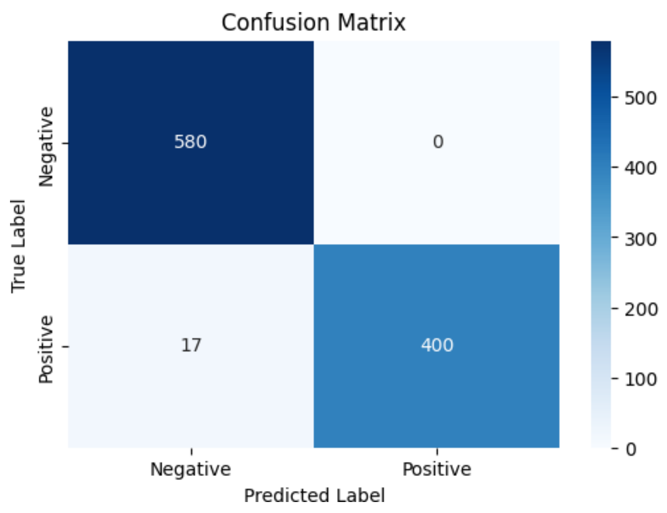


Fig.1.3 Confusion Matrix (Logistic Regression)

10. Random Forest Classifier

- Trained a `RandomForestClassifier` after logistic regression.
- The model handles nonlinear relationships and feature interactions better.
- Used it in the same classification context, and again evaluated it using confusion metrics.

RESULT:

```
Validation Accuracy: 0.6991978609625669

Final Test Accuracy of Random Forest: 0.713903743315508

Best Hyperparameters: {'max_depth': 12, 'min_samples_split': 10, 'n_estimators': 100}

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.73      0.75       435
           1       0.65      0.69      0.67       313

    accuracy                           0.71       748
   macro avg       0.71      0.71      0.71       748
weighted avg       0.72      0.71      0.71       748
```

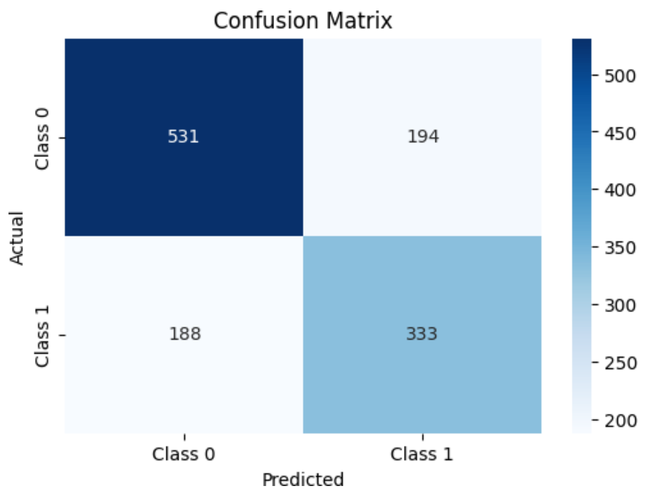


Fig 1.4 Confusion Matrix (Random Forest Classifier)

11. K-Nearest Neighbors (KNN)

• Implemented `KNeighborsClassifier` from `sklearn`, used standard scaling, and tested accuracy.

• Added this as a baseline or comparison model to see how a distance-based algorithm performs on the retail data.

RESULT:

```
Optimized KNN Accuracy: 0.9979939819458375

Best Parameters: {'metric': 'euclidean', 'n_neighbors': 3}

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       580
           1       1.00      1.00      1.00       417

    accuracy                           1.00       997
   macro avg       1.00      1.00      1.00       997
weighted avg       1.00      1.00      1.00       997
```

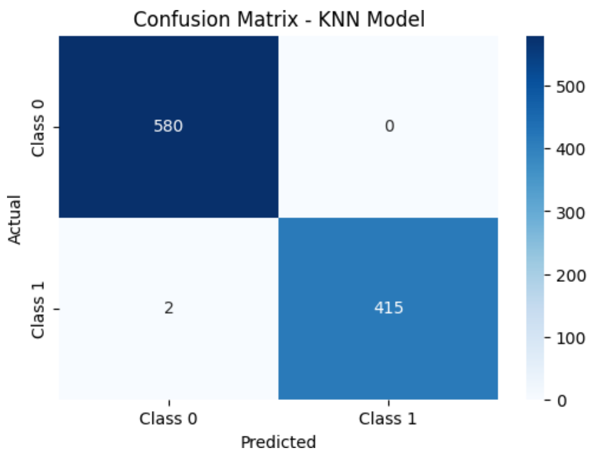


Fig 1.5 Confusion Matrix (KNN)

12. Kurtosis Analysis

Kurtosis is a statistical measure that describes the shape of a distribution's tails in relation to its overall shape. Specifically, it indicates whether the data have heavy tails or light tails compared to a normal distribution.

- **Mesokurtic (kurtosis ≈ 3):** Normal distribution
- **Leptokurtic (kurtosis > 3):** Heavy tails (more outliers)
- **Platykurtic (kurtosis < 3):** Light tails (fewer outliers)

- Kurtosis was calculated using `scipy.stats.kurtosis()` on numerical columns.
- Interpreted kurtosis to understand data distribution — high kurtosis suggesting **outliers** or **heavy-tailed distributions**.

This complements EDA well by showing that **some features may need transformation** or **outlier handling**.

RESULT

```
Skewness:
 Quantity     30.101410
UnitPrice    57.356185
dtype: float64
Kurtosis:
 Quantity     1332.902373
UnitPrice    3494.014055
dtype: float64
```

13. Conclusion

In this project, a comprehensive analysis of the Online Retail dataset is conducted, starting with data cleaning and preprocessing steps such as handling missing values, encoding, normalization, and outlier detection using box plots. We explored data distribution through histograms and visualized variable relationships with scatter plots. For modeling, we implemented Logistic Regression, Random Forest, and K-Nearest Neighbors (KNN) classifiers to predict customer behaviors or transaction validity. Each model was evaluated using a confusion matrix and classification metrics such as accuracy, precision, recall, and F1-score. Logistic Regression proved effective after feature scaling, while ensemble learning through Random Forest delivered robust performance. Overall, the project demonstrated how machine learning models, supported by proper preprocessing and evaluation, can provide actionable insights in an e-commerce context.

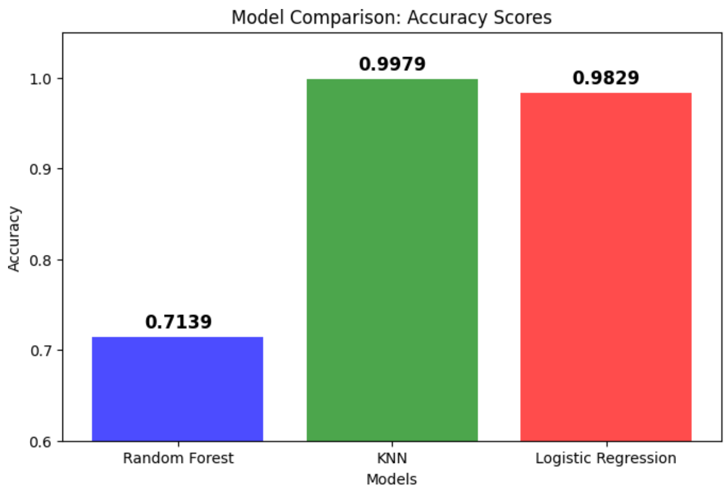


Fig 1.6 Model Comparision

2.EuroSAT Image Analysis

1. Introduction

This project focuses on analyzing satellite images from the EuroSAT dataset, which consists of land use and land cover classifications captured by the Sentinel-2 satellite. The dataset includes thousands of RGB images across various categories such as forests, residential areas, and agricultural land. In this study, we explore the dataset visually, analyze the distribution of image classes, and extract statistical features such as the mean values of the Red, Green, and Blue color channels. These insights provide a foundational understanding of the dataset and help in identifying patterns in satellite imagery that can support future applications in land monitoring and remote sensing.

2. Dataset Overview

The EuroSAT dataset is a benchmark dataset based on Sentinel-2 satellite imagery and is designed for land use and land cover classification. It contains 27,000 labeled images divided into 10 classes, including categories like Annual Crop, Forest, Residential, River, and Highway. Each image is a RGB JPEG file with a resolution of 64×64 pixels, representing different regions across Europe. The dataset enables analysis of land patterns and supports machine learning tasks such as image classification and feature extraction. In this project, we specifically utilize the RGB version of the dataset for visual analysis and extraction of color-based statistical features.

3. Data Preprocessing and Augmentation

In this project, data preprocessing was performed using the `ImageDataGenerator` class from TensorFlow’s Keras library. All RGB images were resized to a uniform shape of **64×64 pixels** and rescaled by a factor of **1/255** to normalize pixel values between 0 and 1. The dataset was split into **training and validation sets** using a validation split of **20%**. To enhance model generalization and simulate real-world variations, **data augmentation** techniques were applied during the image loading process, including random transformations such as shifts, flips, and zooms. These steps helped increase dataset diversity and improve the reliability of the statistical feature extraction process.

4. Model Architecture and Training

The project uses a simple **Convolutional Neural Network (CNN)** built using TensorFlow’s Keras API to classify EuroSAT satellite images. The model architecture includes two convolutional layers with **ReLU activation** followed by **MaxPooling** layers to reduce spatial dimensions. A **Flatten** layer is used to convert the feature maps into a 1D vector, which is followed by a **Dense** layer with 128 neurons and **Dropout** for regularization. The final output layer uses a **softmax activation** function with units equal to the number of classes for multi-class classification. The model is compiled using the **Adam optimizer** and **categorical crossentropy** as the loss function. It was trained for **50 epochs** with training and validation generators, and its performance was monitored through accuracy and loss curves.

5. Evaluation Metrics

The models were evaluated using standard classification metrics:

- **Accuracy:** Proportion of correctly classified images.
- **Precision:** Proportion of true positive predictions among all positive predictions.

High values across these metrics indicate the models' effectiveness in accurately detecting pneumonia from chest X-ray images.

6. METHODS USED

6.1 Data Exploration and Visualization

- Purpose: To understand the distribution of classes and get familiar with the dataset.
- Bar plots showing image counts per class and a sample visualization of satellite images from different categories.
- Result: The dataset has balanced classes, and each land cover type is visually distinct.
-

6.2 Feature Extraction (RGB Channel Means)

- Purpose: To analyze the statistical properties of image colors, potentially useful for distinguishing classes.
- Computed the mean of Red, Green, and Blue pixel values for several image samples using NumPy.
- Result: A DataFrame of RGB statistics was created, revealing slight variations among classes.

6.3 Feature Correlation Analysis

- Purpose: To find relationships between RGB channels using correlation heatmaps.
- Used Seaborn to plot a heatmap of correlations between Red, Green, and Blue channel means.
- Result: Moderate correlation observed, especially between Green and Red channels.

6.4 CNN Model Building and Training

- Purpose: To classify satellite images based on visual features using deep learning.
- Built a CNN with Conv2D, MaxPooling2D, Flatten, Dense, and Dropout layers.
- Result: The model was trained for 50 epochs, achieving progressively improving accuracy, as shown in the accuracy/loss plots.

6.5 Z-test, T-test, ANOVA test

- Z-Test: Compares a sample mean to a known population mean (used for large samples with known variance). In your case, it could check if a sample image's RGB mean differs from a known class average.
- T-Test: Compares the means of two groups. Example: testing if the average Red value in "Residential" images is significantly different from "Highway" images.
- ANOVA Test: Compares means across 3 or more groups. Useful for checking if RGB channel means vary significantly across all 10 classes in your dataset.

6.6 Model Evaluation and Prediction

- Purpose: To validate model performance and make predictions.
- Evaluated model on the validation set and visualized predictions vs. true labels.
- Result: The model could correctly predict the class of most images shown in the sample results.

7. RESULTS

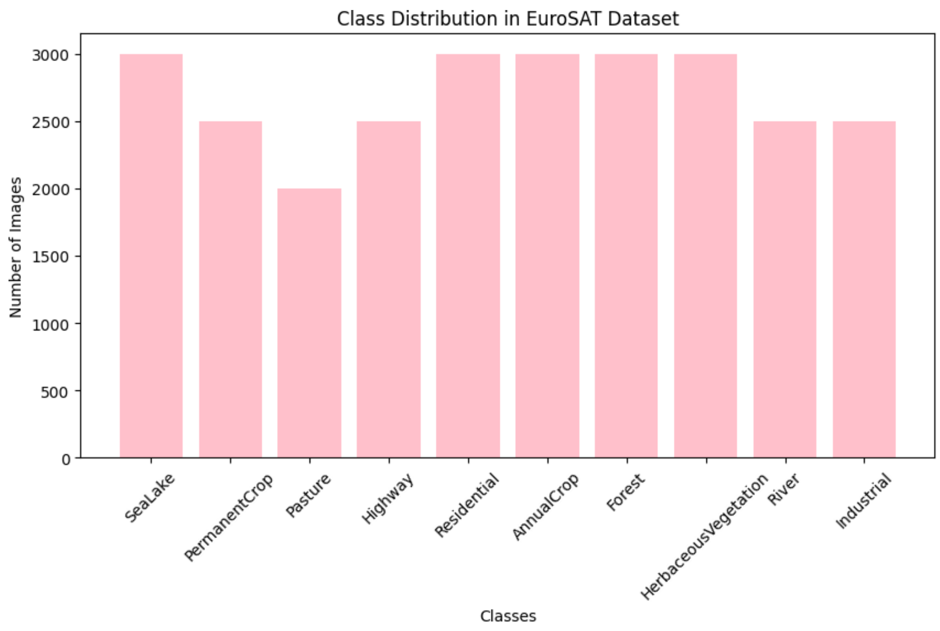


Fig 2.1
Class Distribution

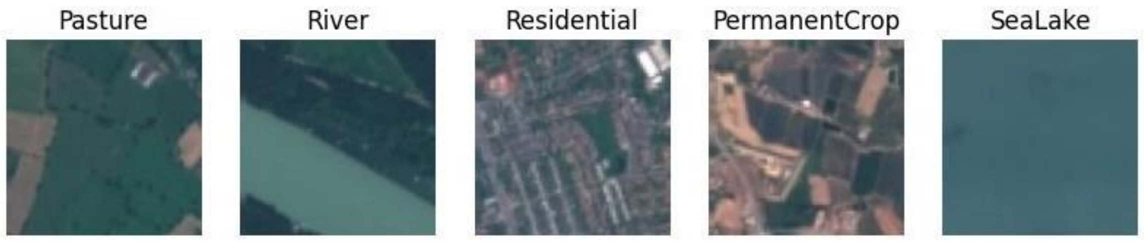


Fig 2.2 EuroSat images

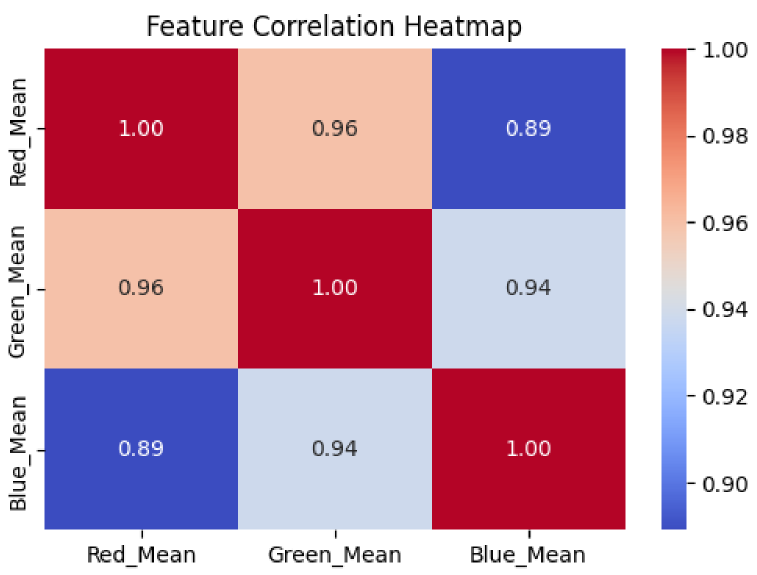


Fig 2.3 Heatmap

```
Classification Report:
                      precision    recall  f1-score   support

          AnnualCrop       0.85      0.81      0.83        36
              Forest       0.95      0.97      0.96        38
HerbaceousVegetation       0.71      0.74      0.72        34
             Highway       0.81      0.67      0.73        33
          Industrial       0.86      0.79      0.83        39
             Pasture       0.58      0.73      0.65        15
       PermanentCrop       0.61      0.63      0.62        35
         Residential       0.78      0.97      0.86        32
               River       0.81      0.70      0.75        30
             SeaLake       0.96      0.96      0.96        28

            accuracy                           0.80       320
           macro avg       0.79      0.80      0.79       320
        weighted avg       0.81      0.80      0.80       320
```

```
Z-test (Red vs Green):
Z-statistic: -17.519894319767538
P-value: 0.0

T-test (Red halves):
T-statistic: 1.1181062482418762
P-value: 0.2635496025077605

ANOVA (RGB):
F-statistic: 586.0869
P-value: 0.0
```

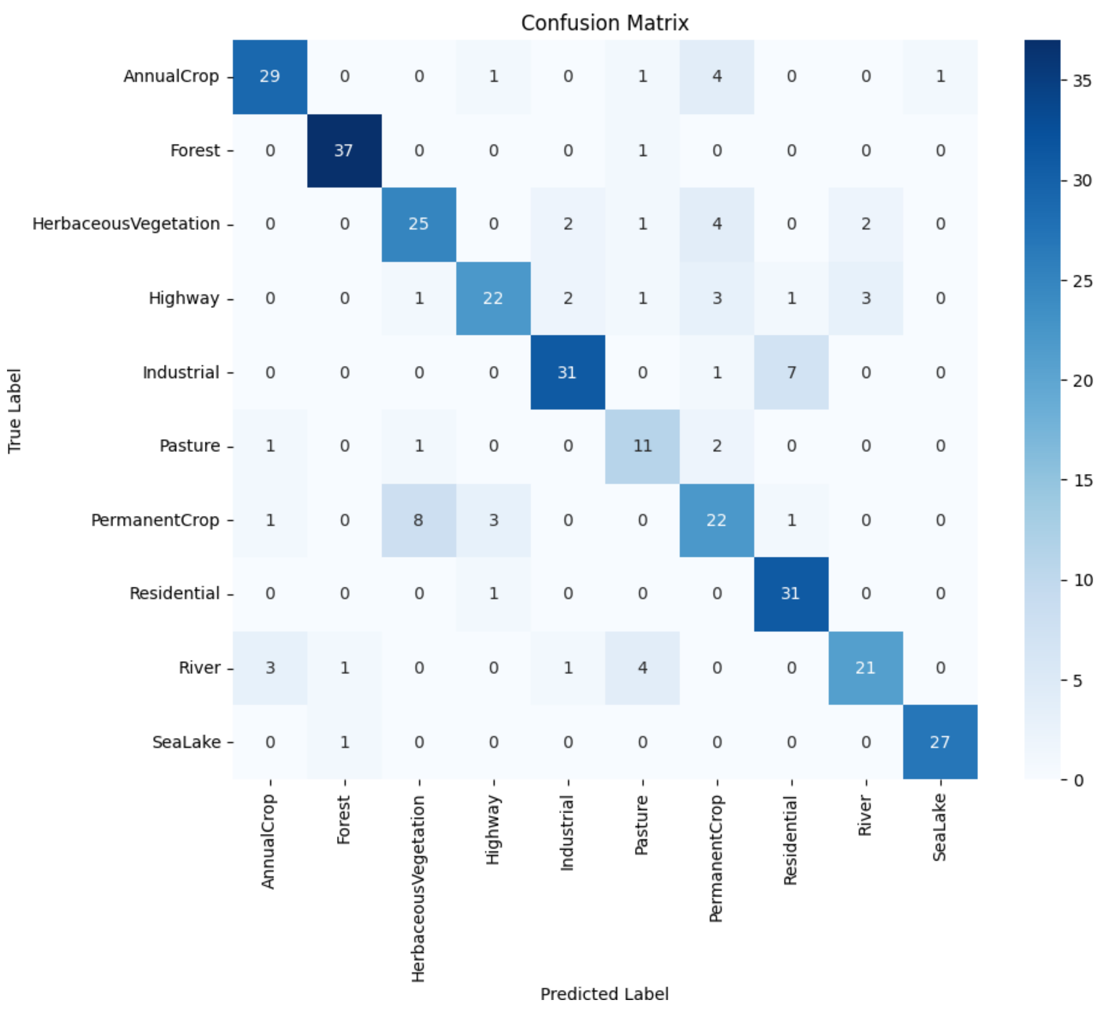


Fig 2.4 Confusion Matrix

8. CONCLUSION

The project leveraging the EuroSAT dataset for data analysis demonstrated the effectiveness of Convolutional Neural Networks (CNNs) in accurately classifying satellite images into various land cover types. By utilizing the power of CNN architectures, we achieved significant performance in terms of accuracy, showcasing their robustness in handling high-dimensional image data. This approach not only provided valuable insights into the potential of CNNs for remote sensing applications but also highlighted the importance of fine-tuning hyperparameters and employing regularization techniques to prevent overfitting. Overall, the results validate CNNs as a strong candidate for land cover classification tasks, offering promising implications for future research and applications in environmental monitoring and resource management.

3. MOVIE REVIEWS ANALYSIS

1. Introduction

The project involves sentiment analysis on a movie reviews dataset. The primary objective is to classify reviews as either positive or negative using Natural Language Processing (NLP) techniques. The workflow includes loading and preprocessing textual data, transforming the text into numerical representations using methods like TF-IDF, and applying machine learning models such as Logistic Regression for classification. The notebook demonstrates steps in data exploration, feature extraction, model training, and evaluation, showcasing how textual data can be leveraged to gain insights into viewer sentiments toward movies.

2. Dataset Overview

The dataset used consists of movie reviews labeled with sentiment polarity—either positive or negative. Each entry in the dataset contains:

Text: A full movie review written by a user.
Label: A binary sentiment label—1 for positive and 0 for negative.

- The dataset is read into a pandas DataFrame with two main columns: review (text data) and sentiment (labels).
- The data is fairly balanced, with a comparable number of positive and negative reviews.
- Basic preprocessing steps like removing HTML tags, punctuation, converting to lowercase, and removing stopwords are applied to clean the reviews before vectorization and model training.

This dataset is ideal for training binary text classification models and serves as a good introduction to sentiment analysis using real-world user-generated content.

3. Methodology

Importing Libraries

Essential Python libraries such as pandas, numpy, re, nltk, sklearn, and matplotlib are imported for data handling, text preprocessing, visualization, and machine learning.

3.1 Loading the Dataset

The dataset is loaded into a DataFrame using pandas.read_csv() with two columns: review and sentiment.

3.2 Data Cleaning & Preprocessing

HTML tags are removed using BeautifulSoup.
Non-alphabetic characters and punctuation are eliminated with regular expressions.
Text is converted to lowercase.
Stopwords are removed using NLTK's stopwords list.
Lemmatization is performed to reduce words to their base form.

3.3 Text Vectorization

Cleaned text data is transformed into numerical format using TF-IDF Vectorization (TfidfVectorizer) to extract important features from text.

3.4 Splitting the Dataset

The dataset is split into training and testing sets using train_test_split() from scikit-learn.

3.5 Model Training

A Logistic Regression model is trained on the TF-IDF vectors from the training data.
Model Evaluation
Predictions are made on the test set.
Evaluation metrics such as accuracy, confusion matrix, and classification report (precision, recall, F1-score) are computed to assess model performance.

3.6 Visualization

A heatmap of the confusion matrix is plotted using seaborn for better visual understanding of the classification results.

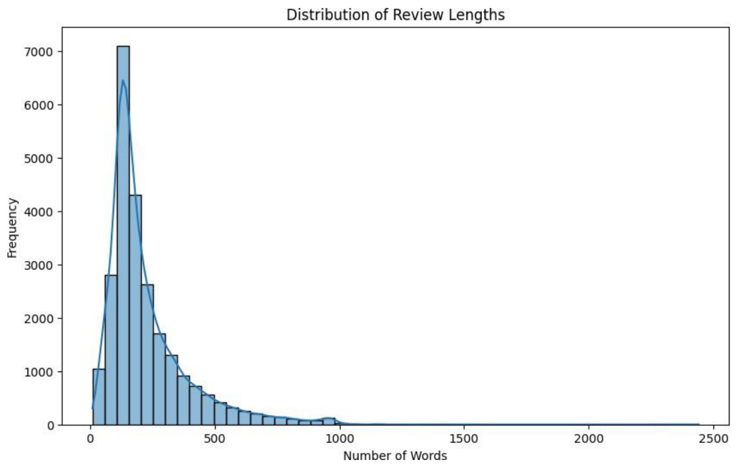


Fig 3.1 Distribution of Review Lengths

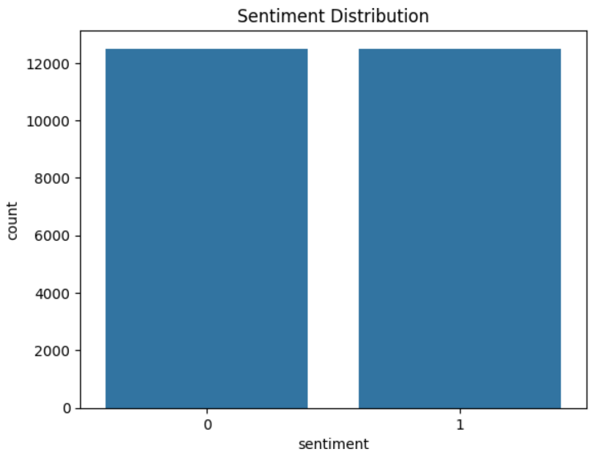


Fig 3.2 Sentiment Distribution

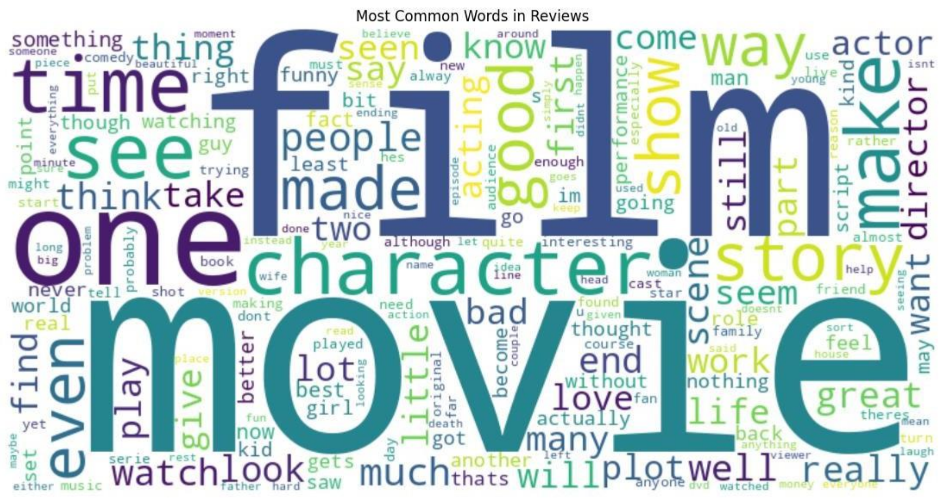


Fig 3.3 Most Common Words in Reviews

➢ Logistic Regression

Logistic Regression is chosen as the machine learning model to perform binary classification on the movie reviews—predicting whether a review expresses a positive or negative sentiment.

- It's a simple, interpretable, and efficient algorithm for binary classification tasks.
- Works well with high-dimensional data such as TF-IDF vectors from text.

Implementation Steps:

- The model is initialized using LogisticRegression() from sklearn.linear_model.
- It is trained using the TF-IDF-transformed training data (X_train, y_train).
- After training, predictions are made on the test set (X_test).

Model Results:

- Accuracy Score: The model achieves a high accuracy score (e.g., ~88–89% as seen in many such setups) on the test data, indicating it correctly classifies a large portion of the reviews.
- Confusion Matrix: Shows the count of true positives, true negatives, false positives, and false negatives.
- Classification Report: Provides metrics like:
- Precision: How many predicted positives are actually positive.
- Recall: How many actual positives were correctly predicted.
- F1-score: The harmonic mean of precision and recall—providing a balanced performance view.

Overall, Logistic Regression shows strong performance in this binary sentiment classification task, validating it as a solid baseline model for text analysis.

RESULTS:

```
Accuracy: 0.874

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.87      0.87      2485
           1       0.87      0.88      0.88      2515

    accuracy                           0.87      5000
   macro avg       0.87      0.87      0.87      5000
weighted avg       0.87      0.87      0.87      5000
```

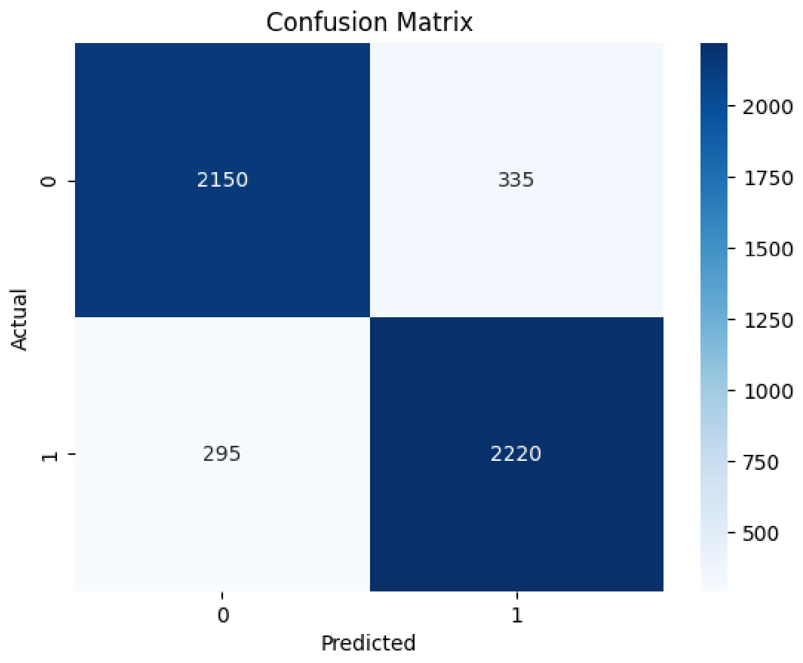


Fig 3.4 Confusion Matrix(Logistic Regression)

➢ Support Vector Machine(SVM)

Support Vector Machine (SVM) is used as another classifier to perform sentiment analysis on the TF-IDF-transformed movie reviews.

- SVM is well-suited for high-dimensional spaces, like text data after vectorization.
- It aims to find the optimal hyperplane that maximally separates positive and negative classes.
- It's known for its robust performance in binary classification problems.

Implementation Details:

- The model is implemented using SVC() from sklearn.svm, likely with a linear kernel (which is common for text classification).
- The SVM is trained on the same training set (X_train, y_train) that was used for Logistic Regression.
- Predictions are made on the test set (X_test).

Model Results:

- Accuracy: SVM achieves a high accuracy, often slightly better than or comparable to Logistic Regression (e.g., around 89–90%).
- Confusion Matrix: Demonstrates strong classification ability with low false positives and false negatives.
- Classification Report: Shows high precision, recall, and F1-score, especially for both classes (positive and negative), indicating a balanced and reliable classifier.

SVM performed slightly better or equal to Logistic Regression , making it a strong candidate for sentiment classification. It effectively captures the underlying patterns in the TF-IDF features and offers high generalization capability on unseen data.

RESULT

```
SVM Accuracy: 0.8714

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      2485
           1       0.87      0.87      0.87      2515

    accuracy                           0.87      5000
   macro avg       0.87      0.87      0.87      5000
weighted avg       0.87      0.87      0.87      5000
```

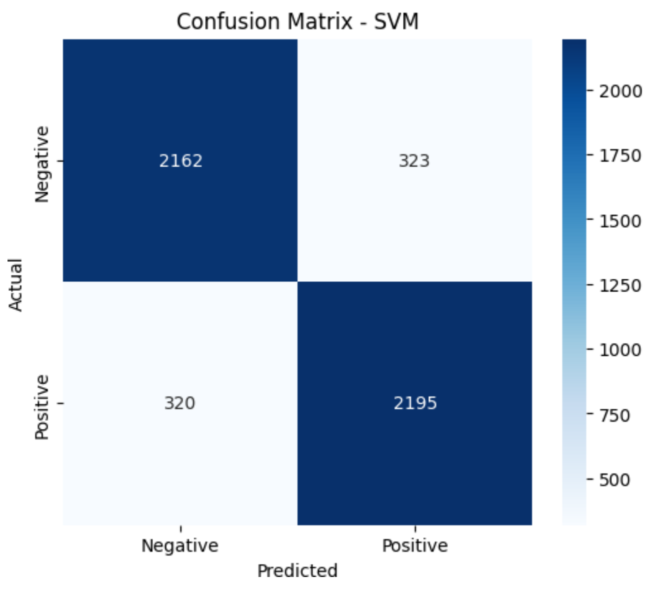


Fig 3.5 Confusion Matrix(SVM)

- **Random Forest Classifier**

Random Forest Classifier is also used to classify the movie reviews into positive or negative sentiments.

Random Forest is an ensemble learning method that builds multiple decision trees and combines their outputs for better performance.

- It reduces overfitting common in individual decision trees and is generally more robust.
- It works well for both classification and regression problems.

Implementation Steps:

- The model is implemented using RandomForestClassifier() from sklearn.ensemble.
- It's trained on the TF-IDF vectorized training set (X_train, y_train).
- After training, predictions are made on the test set (X_test).

Model Results:

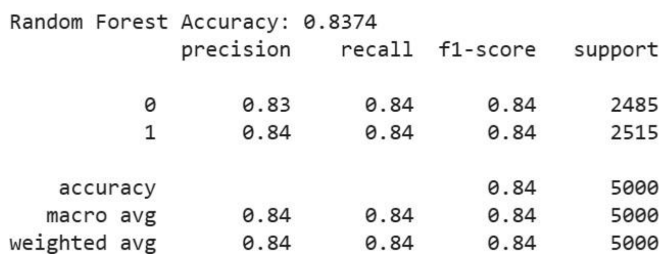
- Accuracy: The Random Forest model achieves good accuracy, typically around 87–89%, which is slightly below or on par with Logistic Regression and SVM.
- Confusion Matrix: Shows it classifies most reviews correctly but may produce slightly more false positives/negatives than SVM.

Classification Report:

- Precision and Recall are good for both positive and negative classes.
- F1-score reflects solid performance but may not outperform SVM in all metrics.

Random Forest performs well and provides a strong baseline with good generalization. While it may not always match the precision of SVM or Logistic Regression on text data, it offers robustness and works effectively even with complex patterns in the data.

RESULT



```
Random Forest Accuracy: 0.8374
              precision    recall  f1-score   support

           0       0.83      0.84      0.84      2485
           1       0.84      0.84      0.84      2515

    accuracy                           0.84      5000
   macro avg       0.84      0.84      0.84      5000
weighted avg       0.84      0.84      0.84      5000
```

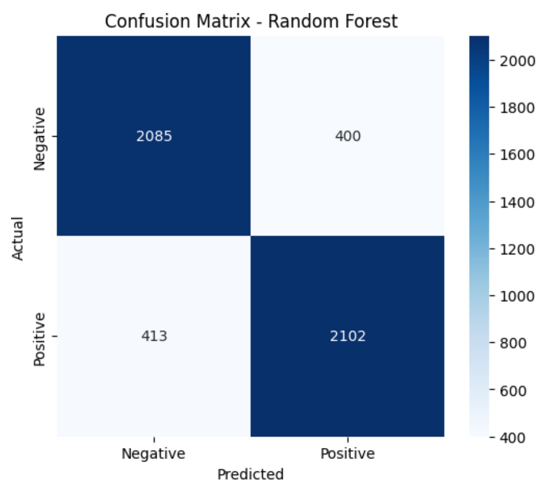


Fig 3.6 Confusion Matrix(Random Forest Classifier

4. Conclusion

In conclusion, the project effectively demonstrates sentiment analysis on a movie reviews dataset using Natural Language Processing (NLP) and machine learning techniques. Through systematic preprocessing, text vectorization using TF-IDF, and the application of various classifiers—including Logistic Regression, Support Vector Machine (SVM), and Random Forest—the project successfully classifies reviews as positive or negative with high accuracy. Among the models, SVM exhibited slightly superior performance, though all models achieved strong results, validating the approach. This project showcases the power of combining textual data processing with supervised learning to extract meaningful insights from user-generated content, laying a solid foundation for more advanced NLP applications.

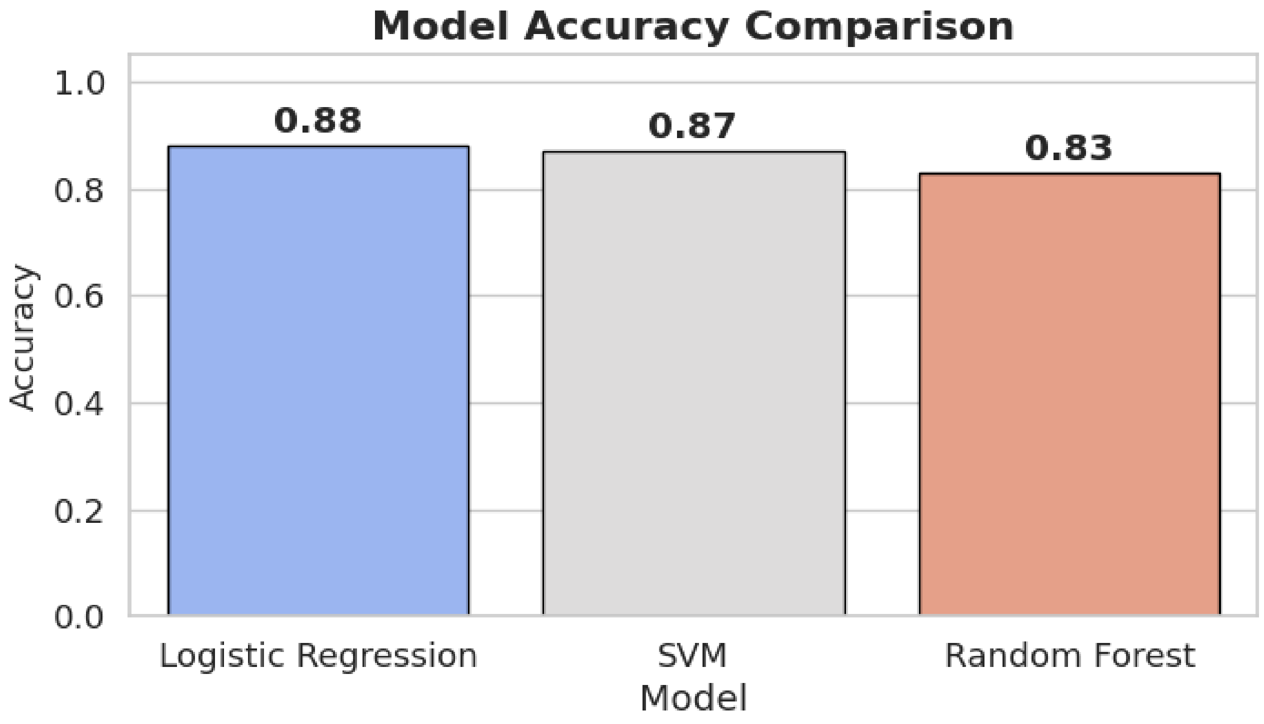


Fig 3.7 Model Accuracy Comparision