



CSE422
Artificial Intelligence

Lab Project Report

Lab Section : 7
Group : 8
Semester : Spring 2025

Group members :

2*****	Md. Shariar Islam Shuvo
2*****	Shakib Shadman Shoumik
2*****	Md Nurullah

Submitted Date: 6-05-2025
Submitted to - Farhan Faruk, Md. Mahadi Hasan

Table of Contents	Page
Introduction	3
Dataset Description	3
Source	3
Dataset Overview	3
Correlation Analysis	5
Imbalanced Dataset	5
Dataset Pre-processing	6
Faults Identified and Solutions Applied	6
Feature Scaling	6
Dataset Splitting	7
Model Training & Testing	7
Models Implemented	7
Model selection/Comparison analysis	8
Conclusion	11

1. Introduction:

This project focuses on predicting delivery speed in a food delivery system using machine learning models. By analyzing various factors such as the distance between the restaurant and the customer, weather conditions, traffic levels, and the courier's experience, the goal is to accurately predict the delivery speed. In this project, we will leverage essential data science techniques, including data preprocessing, feature engineering, and model evaluation, to build an efficient model that can help optimize food delivery operations. Through the application of these methods, the model will provide valuable insights into the factors affecting delivery time and ultimately improve service quality in the food delivery industry.

2. Dataset Description:

Dataset Overview:

Source: [Food_Delivery_Times_Classification](#)

Number of features: The dataset contains 9 columns in total. Of these, 8 are predictor variables (features) and 1 is the target variable (Delivery_Speed).

This is a classification problem because the goal is to predict a categorical target variable (e.g., delivery speed with classes such as "Fast", "Average", "Slow") based on multiple features (like distance, weather, traffic level, etc.). In classification problems, the model predicts discrete classes or categories rather than continuous values, which aligns with the objective of categorizing the data into predefined classes.

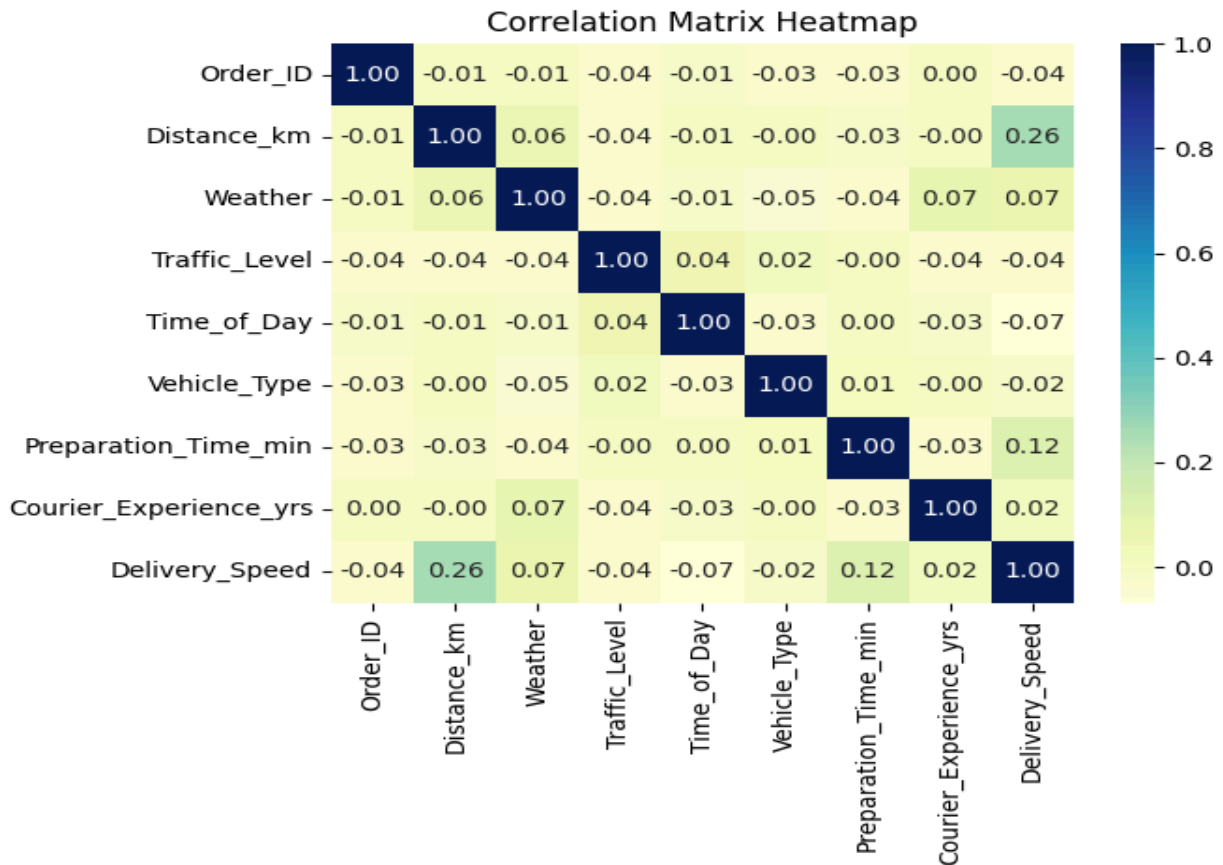
Type of problem: This is a classification problem. The target variable, Delivery_Speed, takes on a small set of discrete categories (e.g. "Slow", "Average", "Fast"), so we're predicting a class label rather than a continuous value.

Number of data points: There are 1,000 delivery records (rows) in the file.

Feature breakdown:

Feature	Type	Description
Order_ID	Identifier	Unique ID for each order (not typically used as a predictive feature)

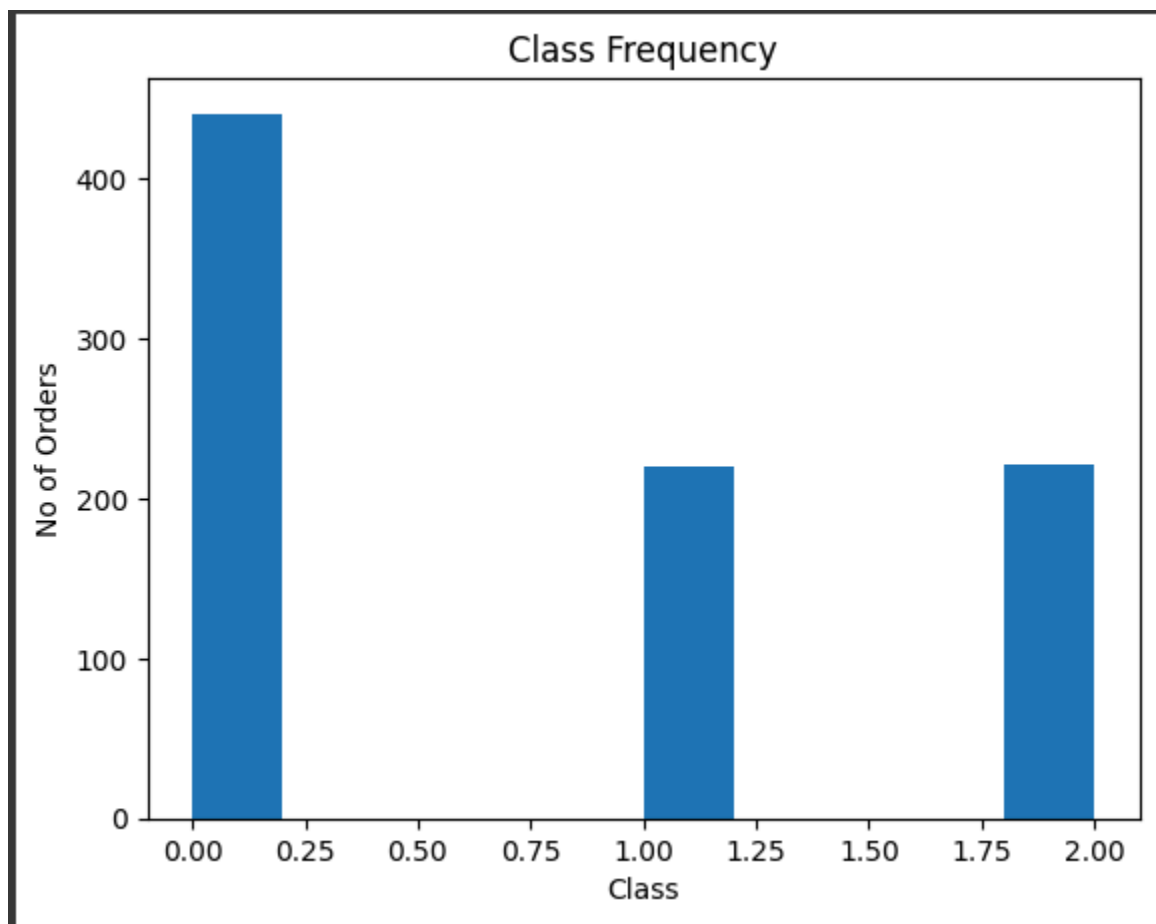
Distance_km	Quantitative	Distance between restaurant and customer (in km)
Preparation_Time_min	Quantitative	Time taken to prepare the order (in minutes)
Courier_Experience_yrs	Quantitative	Delivery Person's Experience
Weather	Categorical	e.g. "Clear", "Rainy", "Foggy", "Windy"
Traffic_Level	Categorical	e.g. "Low", "Medium", "High"
Time_of_Day	Categorical	e.g. "Morning", "Afternoon", "Evening", "Night"
Vehicle_Type	Categorical	e.g. "Bike", "Scooter", "Car"
Target: Delivery_Speed	Categorical	e.g. "Slow", "Average", "Fast"



Correlation Analysis: Using a heatmap to perform the correlation test between the features, we find important relationships between variables, indicating potential feature importance for model prediction.

Imbalanced Dataset:

The dataset consists of two target classes: "Yes" (1) and "No" (0), representing the presence or absence of heart disease. A bar chart illustrating the class distribution reveals a significant imbalance, with a much larger number of instances labeled as not having heart disease. This disparity can negatively influence machine learning models by causing them to favor the majority class, potentially leading to biased predictions.



3. Dataset Pre-processing

Faults Identified and Solutions Applied

Missing Values: Rows containing missing values were removed to simplify the preprocessing, as the dataset is large enough that this decision has minimal impact on data quality. While replacing missing values with the mean or median is a common practice, it was not necessary here due to the dataset's sufficient size.

Categorical Values: Categorical features were encoded using LabelEncoder, which converts each category into a numerical value. This allows the categorical data to be used effectively by machine learning algorithms, while preserving the essential information.

```
Dropping rows with missing values

[9] 1 data = data.dropna()

[10] 1 data.shape

(883, 9)

Check all the null values after dropping null values

1 data.isnull().sum()

0
Order_ID      0
Distance_km    0
Weather        0
Traffic_Level  0
Time_of_Day    0
Vehicle_Type   0
Preparation_Time_min  0
Courier_Experience_yrs  0
Delivery_Speed  0

dtype: int64
```

Feature Scaling

Feature scaling is an essential step in the pre-processing to let numeric features contribute equally during the learning process of a model. We tested multiple scaling techniques, namely, Min-Max Scaling and Robust Scaling.

While both Min-Max Scaling and Robust Scaling improved the model's performance with respect to unscaled data, StandardScaler showed the best results in overall model accuracy and stability. Hence, StandardScaler is selected as the final scaling method for the dataset.

4. Dataset splitting:

- Random/Stratified: The splitting process uses a fixed random_state to ensure reproducibility.
- Train set (70%)
- Test set (30%)

5. Model training & testing:

Models Implemented:

Model	Accuracy%	Key Highlights
K-Nearest Neighbors (KNN)	68.3	Sensitive to scaling, best for well-defined clusters.
Decision Tree Classifier	69.8	Captures complex patterns, interpretable but prone to overfitting.
Logistic Regression	80.7	Strong baseline for linear relationships.
Neural Network	67.9	Can capture complex non-linear patterns, powerful but requires tuning.

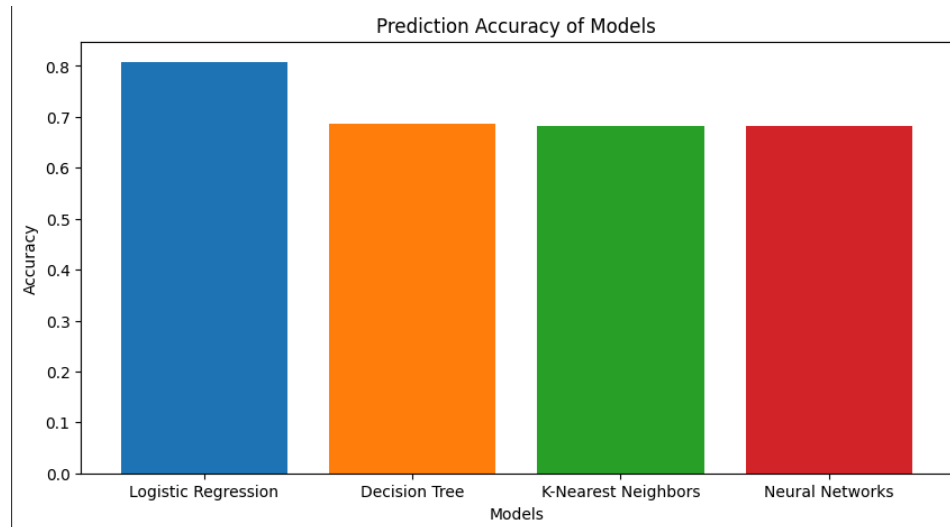
```

Logistic Regression
Accuracy: 0.807547808113286
Confusion Matrix: (1128 7 141)
[[ 14  55  41]
 [  3  1 485]]
Classification Report: {'f1': {'precision': 0.7628947368421053, 'recall': 0.875, 'f1-score': 0.81581573815673, 'support': 136.0}, '1': {'precision': 0.8734046573342465, 'f1-score': 0.888235294117647, 'support': 73.0}, '2': {'precision': 0.8, 'recall': 0.7142857142857143, 'f1-score': 0.7547106543814433}}
Recall: 0.807547808113286
y_pred: [0 1 2 0 0 2 0 0 0 2 0 1 2 0 0 1 2 0 0 2 0 0 0 0 0 0 2 0 0
2 0 1 0 1 1 1 2 1 0 0 0 1 1 0 0 0 0 1 2 2 2 1 0 0 0 0 0 1 1 2 0 0
2 0 0 0 0 0 0 0 1 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 2 1 2 0 0 0 2 1 1 0 1 0 0 0 0 1 0 2 0 0 0 1 0 0 0 0 0 0 0 1 2 2
1 0 0 0 2 0 0 0 0 2 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 2 0 0 1 0 0 0 0 2 0 0 0 0 2 2 1 0 0 0 0 0 0 0 2 1 0 0 0 2
0 0 1 0 1 2]
Decision Tree
Accuracy: 0.698112875471606
Confusion Matrix: (196 18 22)
[[ 17  34  21]
 [24  3 363]]
Classification Report: {'f1': {'precision': 0.7218845112781954, 'recall': 0.7858823529411765, 'f1-score': 0.7534646481487, 'support': 136.0}, '1': {'precision': 0.7397260273972602, 'f1-score': 0.7397260273972602, 'support': 73.0}, '2': {'precision': 0.583226338838568, 'recall': 0.625, 'f1-score': 0.603120479745669}}
Recall: 0.698112875471606
y_pred: [1 1 0 0 0 0 0 0 0 0 2 2 2 0 1 1 2 0 0 1 2 0 0 1 0 2 2 1 0 0 1 0
0 0 1 1 1 2 0 0 0 0 1 1 1 1 0 0 0 2 0 0 0 0 0 1 0 0 0 0 1 0 2 0 0
2 0 1 2 0 1 1 1 2 1 0 0 0 0 1 0 2 1 0 0 2 1 0 0 0 1 0 0 1 1 2 0 0
2 0 0 0 0 0 0 0 1 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 2 1 1 1 2 1 0 0 0 0 2 2 0 0 1 1 2 0 0 0 1 0 0 0 1 2 2
1 0 0 0 2 0 0 0 1 0 0 2 2 1 1 0 0 1 0 0 0 0 2 2 2 2 2 0 0 0 2 0 0
0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 2 2 2 1 0 0 0 0 0
2 0 0 0 0 2]
K-Nearest Neighbors
Accuracy: 0.683586762643283
Confusion Matrix: (1122 18 141)
[[ 24  47]
 [33  1 221]]
Classification Report: {'f1': {'precision': 0.654978762339381, 'recall': 0.8235294117647059, 'f1-score': 0.7296416938118749, 'support': 136.0}, '1': {'precision': 0.8183448275862069, 'recall': 0.643835164383562, 'f1-score': 0.7125572519883969, 'support': 73.0}, '2': {'precision': 0.6111111111111112, 'recall': 0.39285714285714285}}
Recall: 0.683586762643283
y_pred: [0 1 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 2 0 1 0 0 1 0
0 0 1 1 2 0 0 0 0 0 1 1 2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0
2 0 1 0 2 0 1 1 2 1 2 0 0 0 1 0 2 1 0 0 1 0 0 0 1 0 0 0 2 0 0 0
2 0 0 0 0 0 0 0 1 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 0 1 0 2 0 0 0 2 1 1 1 2 1 0 0 0 0 2 2 0 0 1 1 2 0 0 0 0 0 0 0 0
0 1 0 0 2 0 1 0 0 0 0 0 2 1 1 0 0 0 1 2 0 1 0 0 2 0 0 0 0 0 0 0 0
0 1 1 0 0 0 2 0 0 1 0 0 0 0 0 0 0 2 2 1 0 1 0 0 0 0 0 0 1 0 0 0 2 2
0 0 0 0 2 2]
Neural Networks
Accuracy: 0.675945208188829
Confusion Matrix: (1136 11 91)
[[ 35  46 241]]
Classification Report: {'f1': {'precision': 0.648839779805525, 'recall': 0.8529411764705882, 'f1-score': 0.7318611807381783, 'support': 136.0}, '1': {'precision': 0.7843137254981081, 'recall': 0.6111111111111112, 'f1-score': 0.69285714285714285}, '2': {'precision': 0.7272727272727273, 'recall': 0.42857142857142855}}
Recall: 0.675945208188829

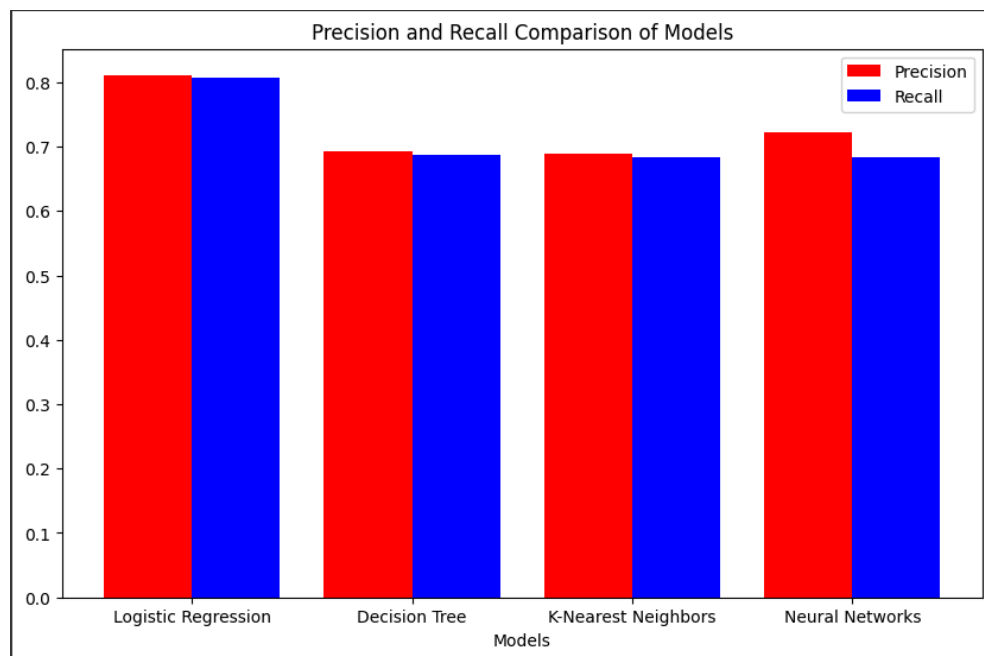
```

6. Model selection/Comparison analysis:

Bar chart showcasing prediction accuracy of all models:



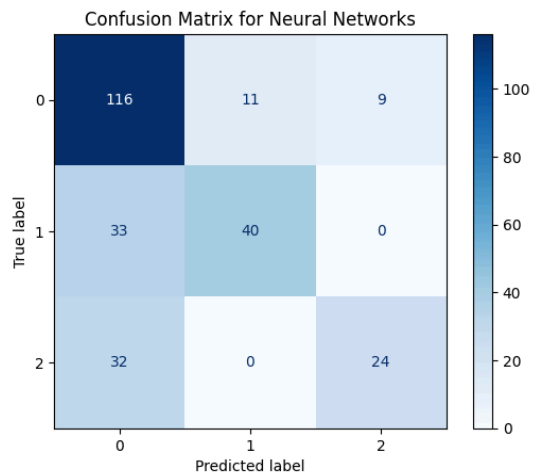
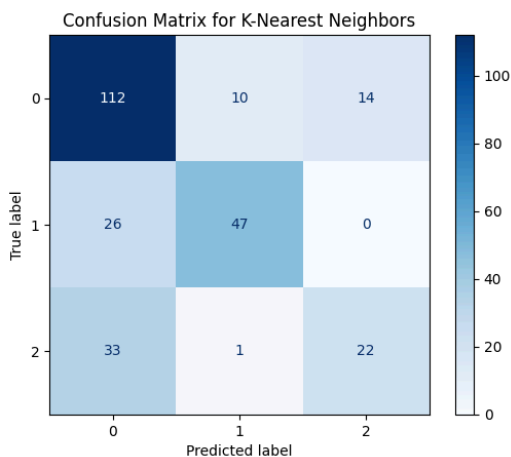
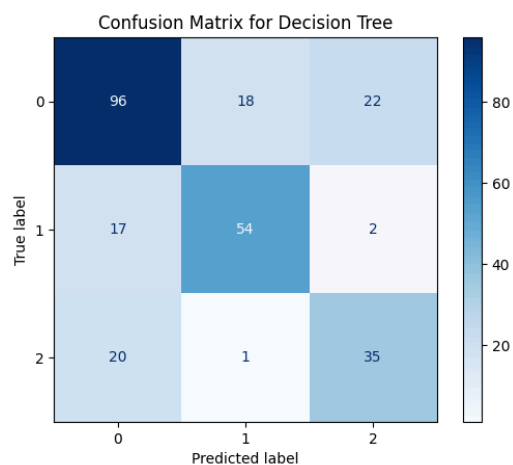
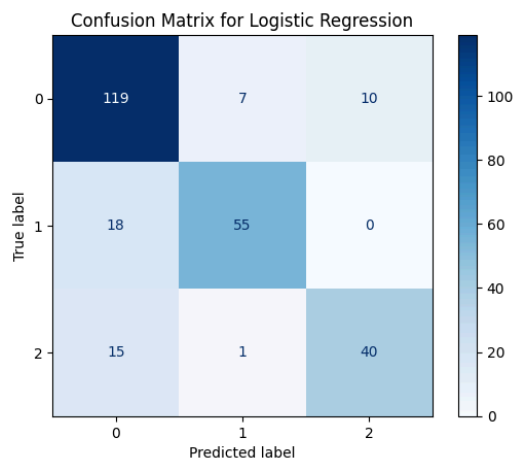
Precision, recall comparison of each model:



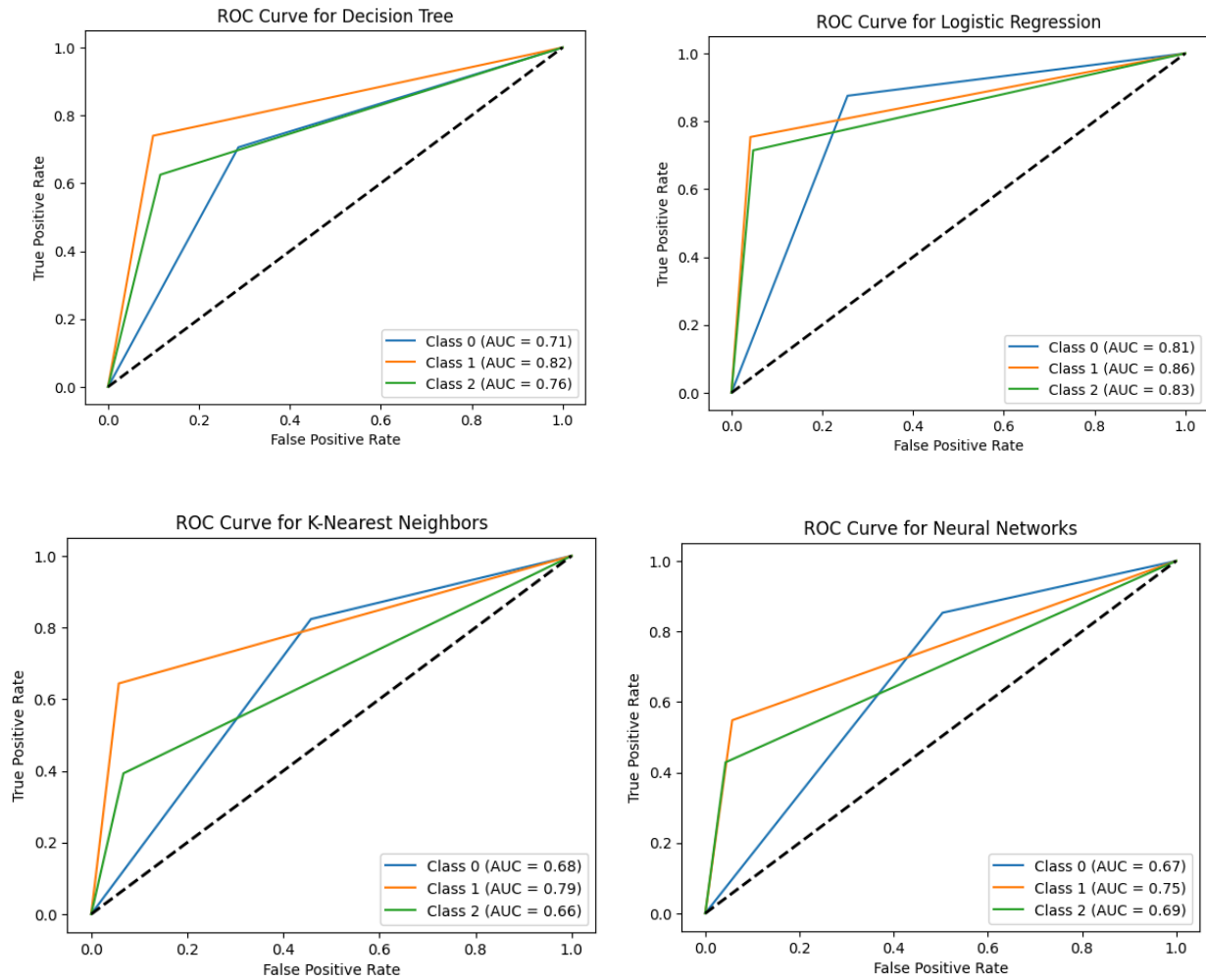
In terms of precision and recall, Logistic Regression had the best balance, with precision of 0.811 and recall of 0.808. It showed high recall for Class 0 (0.875) and Class 1 (0.753). Decision Tree had precision

of 0.700 and recall of 0.698, with strong performance for Class 1 (precision 0.740, recall 0.740) but struggled with Class 2 (precision 0.593, recall 0.625). K-Nearest Neighbors (KNN) had precision of 0.689 and recall of 0.683, performing well for Class 1 (precision 0.810, recall 0.644) but underperforming for Class 2 (precision 0.611, recall 0.393). Neural Networks had precision of 0.699 and recall of 0.679, excelling in recall for Class 0 (0.853) but struggling with Class 1 (precision 0.784, recall 0.548) and Class 2 (precision 0.727, recall 0.429).

Confusion Matrix:



AUC score and ROC Curve:



```

1 def model_train(X, y, random_state):
2     X_train, X_test, y_train, y_test = train_test_split(
3         X, y, test_size=0.3, random_state=random_state)
4
5     results = {}
6
7     logistic_model = LogisticRegression()
8     logistic_model.fit(X_train, y_train)
9     y_pred_logistic = logistic_model.predict(X_test)
10    results['Logistic Regression'] = accuracy_list(y_test, y_pred_logistic)
11
12    dt_model = DecisionTreeClassifier()
13    dt_model.fit(X_train, y_train)
14    y_pred_dt = dt_model.predict(X_test)
15    results['Decision Tree'] = accuracy_list(y_test, y_pred_dt)
16
17    knn_model = KNeighborsClassifier(n_neighbors=5)
18    knn_model.fit(X_train, y_train)
19    y_pred_knn = knn_model.predict(X_test)
20    results['K-Nearest Neighbors'] = accuracy_list(y_test, y_pred_knn)
21
22    neural_networks_model = Sequential([
23        Flatten(input_shape=(X_train.shape[1],)),
24        Dense(128, activation='relu'),
25        Dense(len(y.unique()), activation='softmax')
26    ])
27
28    neural_networks_model.compile(optimizer=Adam(0.001),
29                                loss=SparseCategoricalCrossentropy(from_logits=False),
30                                metrics=['accuracy'])
31
32    neural_networks_model.fit(
33        X_train, y_train, epochs=10, batch_size=128, verbose=0)
34
35    y_pred_prob = neural_networks_model.predict(X_test)
36    y_pred_neural = np.argmax(y_pred_prob, axis=1)
37    results['Neural Networks'] = accuracy_list(y_test, y_pred_neural)
38
39    return results, y_test

```

Model Comparison

- Metrics Used: Accuracy, Precision, Recall, Confusion Matrix
- Observations: Logistic Regression achieved the highest accuracy at 80.7%, outperforming all other models. The Decision Tree Classifier followed with an accuracy of 69.8%, capturing complex patterns but prone to overfitting. K-Nearest Neighbors (KNN) had an accuracy of 68.3%, sensitive to scaling and best for well-defined clusters. The Neural Network, although capable of capturing complex non-linear patterns, showed the lowest accuracy at 67.9% due to insufficient tuning.

7. Conclusion

From the results, we observe that Logistic Regression performed the best, achieving an accuracy of 80.7%, followed closely by the Decision Tree Classifier at 69.8%. While Neural Networks showed a lower accuracy of 67.9%, they have the potential to model more complex patterns but require tuning to optimize performance. KNN showed the lowest accuracy, indicating its sensitivity to scaling and suitability for well-defined clusters.

The precision and recall comparisons demonstrate that simpler models, such as Logistic Regression, performed competitively with more complex models, offering a balance between interpretability and accuracy. Logistic Regression's strong performance can be attributed to its ability to handle linear relationships in the data and its computational efficiency. On the other hand, Decision Trees offer a good tradeoff between complexity and interpretability, but they are prone to overfitting, especially when trained on small datasets.

The models' performance was influenced by various factors such as the quality of pre-processed data, handling of missing values, feature scaling, and balancing class imbalances. Neural Networks excelled in learning complex relationships but faced challenges such as overfitting. Techniques such as regularization and balancing the dataset helped mitigate some of these issues.

Challenges encountered during the project included dealing with class imbalance, which affected the models' ability to correctly classify certain classes, and overfitting in complex models. These challenges were addressed using data balancing techniques and regularization to improve model generalization.

Key Takeaways:

- Data preprocessing plays a crucial role in enhancing model performance.
- Feature engineering significantly improves the predictive power of models.
- Model selection depends on the dataset's characteristics and problem requirements, with simpler models often providing competitive results for certain datasets.