Name: Alok Yadav Div: D15C Roll No.: 59

Experiment No.6

Aim: Perform Classification modelling
a. Choose a classifier for classification problem.
b. Evaluate the performance of the classifier.
Perform Classification using the below 4 classifiers on the same dataset which you have used for for experiment no 5:
K-Nearest Neighbors (KNN)
Naive Bayes
Support Vector Machines (SVMs)
Decision Tree

Theory:

1. **Decision Tree:** Decision Tree is a supervised learning algorithm that recursively splits the data into subsets based on feature values, creating a tree-like structure. Each internal node represents a decision based on a feature, and each leaf node corresponds to a class label (for classification) or a value (for regression). The goal is to partition the data in a way that minimizes uncertainty or entropy at each decision point.

2. **Naive Bayes: Naive Bayes** is a probabilistic classifier based on Bayes' Theorem, which assumes that the features are conditionally independent given the class label. It calculates the probability of each class and assigns the class with the highest probability to each instance. Naive Bayes is simple, efficient, and works well with large datasets, especially when the independence assumption holds, but it may perform poorly if features are highly correlated.

Dataset Overview: Yellow Taxi: Yellow Medallion Taxicabs: These are the famous NYC yellow taxis that provide transportation exclusively through street hails. The number of taxicabs is limited by a finite number of medallions issued by the TLC. You access this mode of transportation by standing in the street and hailing an available taxi with your hand. The pickups are not pre-arranged.

Implementation:

1. **Preprocessing and Split data**

```
if df['payment_type'].dtype == 'object':
    df['payment_type'] = pd.to_numeric(df['payment_type'], errors='coerce')

df.dropna(subset=[
    'passenger_count', 'trip_distance',
    'pickup_longitude', 'pickup_latitude',
    'dropoff_longitude', 'dropoff_latitude',
    'payment_type', 'fare_amount', 'extra', 'mta_tax',
    'tolls_amount', 'improvement_surcharge', 'tip_flag',
    'pickup_hour', 'trip_duration', 'pickup_dayofweek',
    'store_and_fwd_flag'
], inplace=True)
df.reset_index(drop=True, inplace=True)
df['store_and_fwd_flag'] = df['store_and_fwd_flag'].map({'N': 0, 'Y': 1})
features = [
    'passenger_count', 'trip_distance',
    'pickup_longitude', 'pickup_latitude',
    'dropoff_longitude', 'dropoff_latitude',
    'payment_type', 'fare_amount', 'extra', 'mta_tax',
    'tolls_amount', 'improvement_surcharge', 'tip_flag',
    'pickup_hour', 'trip_duration', 'pickup_dayofweek'
]
scaler = StandardScaler()
df[features] = scaler.fit_transform(df[features])
X = df[features]
y = df['store_and_fwd_flag']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

2. Decision Tree Classification:

```
dtc = DecisionTreeClassifier(random_state=42)
dtc.fit(X_train, y_train)
y_pred_dt = dtc.predict(X_test)

print("Decision Tree Classifier Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
print("Classification Report:\n", classification_report(y_test, y_pred_dt))
```

```
Decision Tree Classifier Results:
Accuracy: 0.9879466666666666
Confusion Matrix:
 [[296337   2053]
 [  1563     47]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99    298390
           1       0.02      0.03      0.03      1610

    accuracy                           0.99    300000
   macro avg       0.51      0.51      0.51    300000
weighted avg       0.99      0.99      0.99    300000
```

Accuracy: 98.79% overall accuracy.
Confusion Matrix: The model heavily favors class 0 (majority class), with only 47 correct predictions for class 1 (minority class).

- **True Positives (TP)**: 47
- **False Positives (FP)**: 2053
- **True Negatives (TN)**: 296337
- **False Negatives (FN)**: 1563

Performance for Class 1:

- **Precision**: 0.02
- **Recall**: 0.03
- **F1-score**: 0.03

Performance for Class 0:

- **Precision**: 0.99
- **Recall**: 0.99
- **F1-score**: 0.99

3. **Naive Bayes Classification:**

```
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)

print("\nNaïve Bayes Classifier Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_nb))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_nb))
print("Classification Report:\n", classification_report(y_test, y_pred_nb))
```

```
Naïve Bayes Classifier Results:
Accuracy: 0.9688
Confusion Matrix:
 [[290535   7855]
 [  1505    105]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98    298390
           1       0.01      0.07      0.02      1610

    accuracy                           0.97    300000
   macro avg       0.50      0.52      0.50    300000
weighted avg       0.99      0.97      0.98    300000
```

Accuracy: 96.88% overall accuracy.
Confusion Matrix: The model predominantly predicts class 0 (majority class), with only 105 correct predictions for class 1 (minority class).

- **True Positives (TP)** = 105
- **False Positives (FP)** = 7855
- **True Negatives (TN)** = 290535
- **False Negatives (FN)** = 1505

Performance for Class 1:

- **Precision**: 0.01
- **Recall**: 0.07
- **F1-score**: 0.02

Performance for Class 0:

- **Precision**: 0.99
- **Recall**: 0.97
- **F1-score**: 0.98

The model has high accuracy due to the dominance of class 0 but struggles with classifying class 1.

Conclusion:

In this experiment, we applied Decision Tree and Naive Bayes classification on the NYC taxi dataset to predict the store_and_fwd_flag. Both the Decision Tree and Naive Bayes classifiers performed well in terms of overall accuracy (98.79% and 96.88%, respectively), but their performance was heavily influenced by the class imbalance, as they favored the majority class (class 0). Both models struggled with correctly classifying the minority class (class 1), showing very low precision, recall, and F1-scores for class 1