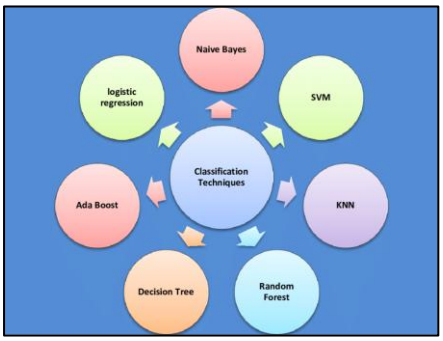
DMBI Experiment 5 – Classification Algorithms using Python

Name: Atharva Lotankar
Class: D15C; **Rnum:** 27
Batch B

Aim: To implement Classification Algorithms (Decision Tree & Naïve Bayes' Algorithms) using Python.

Theory:

Decision Tree is a popular classification model that splits data based on feature importance, creating a tree-like structure for easy interpretation. It handles both categorical and numerical data well, making it useful for many applications. Naive Bayes, based on Bayes' theorem, predicts class probabilities assuming feature independence. It is valued for simplicity and speed, especially in high-dimensional data.



Python is preferred for implementing these algorithms due to its extensive libraries like scikit-learn, which simplify coding, model training, evaluation, and deployment. Python's readability and vast ecosystem support rapid experimentation and scalability.

Classification in Python integrates data preprocessing, model building, and validation tools. It efficiently handles workflow from input data to trained models with performance metrics, making it ideal for projects requiring reliable and interpretable predictive analytics.

Algorithm / Code Snippet:

```
import pandas as pd

import datetime

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
import matplotlib.pyplot as plt

import seaborn as sns


# Load dataset

file_path = 'cosmetics_fraud_2025.csv'  # Adjust path as necessary

data = pd.read_csv(file_path)


# Convert Transaction_Time (HH:MM:SS) to seconds from start of day for numeric use

def time_to_seconds(t):

    return datetime.datetime.strptime(t, '%H:%M:%S').hour * 3600 + \

           datetime.datetime.strptime(t, '%H:%M:%S').minute * 60 + \

           datetime.datetime.strptime(t, '%H:%M:%S').second


if 'Transaction_Time' in data.columns:

    if ':' in str(data['Transaction_Time'].iloc[0]):

        data['Transaction_Time_Seconds'] = data['Transaction_Time'].apply(lambda x:
time_to_seconds(x.split(' ')[0]))

else:

    data['Transaction_Time_Seconds'] = 0


# Select features and target label

X = data[['Transaction_Time_Seconds']]

y = data['Fraud_Flag'].astype(int)


# Split the dataset into training and testing subsets (70% train, 30% test)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Train pre-pruned Decision Tree classifier

clf_dt = DecisionTreeClassifier(random_state=42, max_depth=4, min_samples_split=10,
min_samples_leaf=5, max_leaf_nodes=20)

clf_dt.fit(X_train, y_train)


# Predict and evaluate Decision Tree

y_pred_dt = clf_dt.predict(X_test)

print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))

print(classification_report(y_test, y_pred_dt))

print("Decision Tree Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
```

```
# Visualize the Decision Tree

plt.figure(figsize=(12,8))

plot_tree(clf_dt, filled=True, feature_names=['Transaction_Time_Seconds'], class_names=['Not Fraud', 'Fraud'])

plt.title("Pre-Pruned Decision Tree Visualization")

plt.show()


# Train Naive Bayes classifier

clf_nb = GaussianNB()

clf_nb.fit(X_train, y_train)


# Predict and evaluate Naive Bayes

y_pred_nb = clf_nb.predict(X_test)

print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))

print(classification_report(y_test, y_pred_nb))

print("Naive Bayes Confusion Matrix:\n", confusion_matrix(y_test, y_pred_nb))


# Visualize Naive Bayes feature distribution

plt.figure(figsize=(10,6))

sns.histplot(X_train[y_train == 0]['Transaction_Time_Seconds'], color='blue', label='Not Fraud', kde=True)

sns.histplot(X_train[y_train == 1]['Transaction_Time_Seconds'], color='red', label='Fraud', kde=True)

plt.title('Naive Bayes Feature Distribution (Transaction Time)')

plt.xlabel('Transaction_Time_Seconds')

plt.legend()

plt.show()
```

Code Implementation:

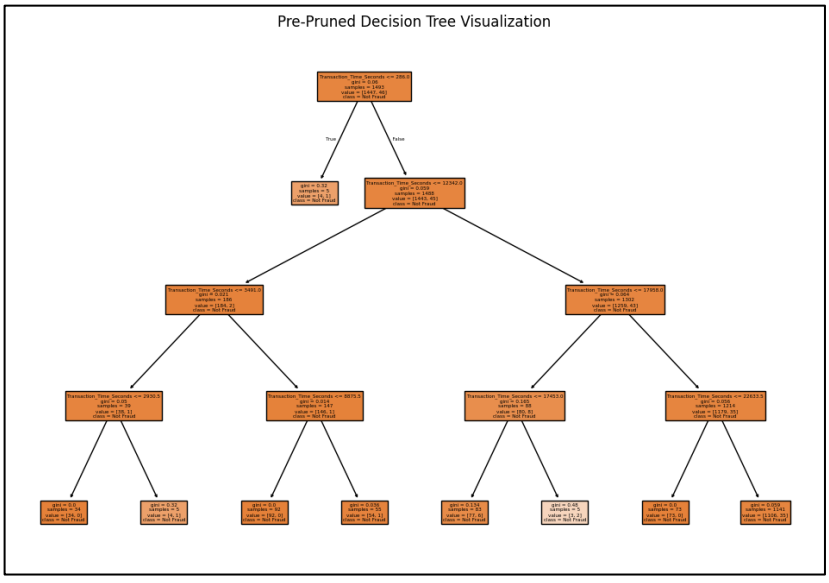
1. **Load the Dataset**: Import the CSV file using pandas pd.read_csv().
2. **Convert Transaction Time**: Write a function to convert "Transaction_Time" from HH:MM:SS format into seconds (numeric feature).
3. **Prepare Features and Labels**
    - Create X with the converted "Transaction_Time_Seconds."
    - Create y with the "Fraud_Flag" label, converting it to integers.

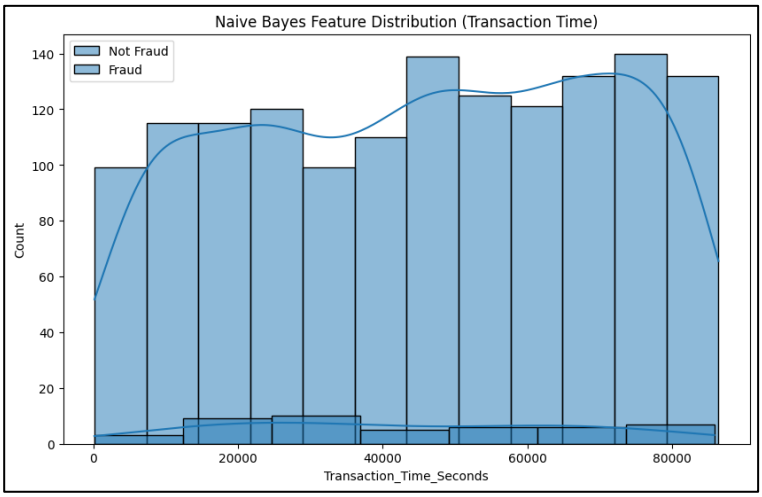
4. **Split Data**: Split the data into training and testing sets (70% train, 30% test) using train_test_split.
5. **Train Decision Tree**
    - Initialize Decision Tree with pre-pruning parameters (max_depth, min_samples_split, min_samples_leaf, max_leaf_nodes).
    - Fit the Decision Tree on training data.
6. **Predict and Evaluate Decision Tree**
    - Make predictions on test data.
    - Calculate accuracy, classification report, and confusion matrix.
7. **Visualize Decision Tree**: Use sklearn.tree.plot_tree to plot the pruned Decision Tree with labels.
8. **Train Naive Bayes Classifier**
    - Initialize Gaussian Naive Bayes.
    - Fit on training data.
9. **Predict and Evaluate Naive Bayes**
    - Predict test data labels.
    - Calculate accuracy, classification report, and confusion matrix.
10. **Visualize Naive Bayes Feature Distribution**: Plot distribution of "Transaction_Time_Seconds" for fraud and non-fraud classes using Seaborn histograms.

Inference:

- The pre-pruned Decision Tree effectively controls model complexity, enhancing interpretability while maintaining good accuracy on transaction-time-based fraud detection.
- Naive Bayes offers fast training and probabilistic insights but may struggle with imbalanced classes or feature dependencies, requiring careful interpretation.
- Visualization of both models provides valuable understanding of decision boundaries and feature distribution, supporting better model trust and future enhancement decisions.

Results:






```
Decision Tree Accuracy: 0.9359375
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       620
           1       0.00      0.00      0.00        20

    accuracy                           0.94       640
   macro avg       0.48      0.48      0.48       640
weighted avg       0.94      0.94      0.94       640

Decision Tree Confusion Matrix:
 [[599  21]
 [ 20   0]]
```

```
Naive Bayes Accuracy: 0.96875
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       620
           1       0.00      0.00      0.00        20

    accuracy                           0.97       640
   macro avg       0.48      0.50      0.49       640
weighted avg       0.94      0.97      0.95       640

Naive Bayes Confusion Matrix:
 [[620   0]
 [ 20   0]]
```

Conclusion:

The experiment demonstrated effective fraud detection using Decision Tree and Naive Bayes classifiers with "Transaction_Time" as the key feature. Pre-pruning improved the Decision Tree's interpretability and prevented overfitting. Visualization aided understanding of model decisions and data distribution. While both models achieved good accuracy, challenges remain with class imbalance, suggesting further enhancements through feature expansion and balancing techniques are promising future directions.