Name - Janhvi Saste
PRN - 22311825
Roll No - 282065
Year - SY B

Assignment No - 4

Data Analysis and Visualization of Heart Disease Dataset

Problem Statement :

A cosmetics shop has collected data on customer details and wants to predict whether customers will respond positively to a special offer. The dataset provided is analogous to the one used in the given code, which includes features like GRE, GPA, and rank to predict admission outcomes. In this context, we adapt the dataset to represent customer attributes (e.g., age, purchase history, loyalty score) to predict whether a customer will accept a special offer. The goal is to apply an appropriate machine learning algorithm to classify customer responses and evaluate the model's performance using metrics such as accuracy, precision, recall, and F1 score, along with a confusion matrix.

Objective :

The objective is to:

- Build a machine learning model to predict whether a customer will respond positively (accept) to a special offer based on their attributes.
- Evaluate the model's performance using a confusion matrix and compute the following metrics:
    - Accuracy
    - Precision
    - Recall
    - F1 Score
- Provide insights into the model's effectiveness and its potential application in optimizing marketing strategies for the cosmetics shop.

Software Used :

- Python 3.x
- Google Colab

Libraries and Packages Used :

The following Python libraries were utilized for data preprocessing and analysis:

- Pandas (For loading, manipulating, and analyzing tabular data)
- Scikit-learn :
  - train_test_split: To split the dataset into training and testing sets.
  - RandomForestClassifier: To build the classification model.
  - confusion_matrix: To generate the confusion matrix for model evaluation.
  - accuracy_score, precision_score, recall_score, f1_score: To compute performance metrics.

Theory :

Methodology :

The methodology involves using a Random Forest Classifier, a robust ensemble machine learning algorithm, to predict customer responses (binary classification: accept or reject the offer). The dataset is split into training (80%) and testing (20%) sets. The model is trained on the training data and evaluated on the test data using a confusion matrix and performance metrics.

Main Function :

- The Random Forest Classifier constructs multiple decision trees during training and outputs the class that is the mode of the classes of the individual trees. It uses features like age, purchase history, and loyalty score (analogous to GRE, GPA, and rank in the provided dataset) to predict the response (accept/reject, analogous to admit/not admit).

Advantages :

- Robustness: Handles non-linear relationships and interactions between features effectively.
- Reduced Overfitting: Averaging multiple trees reduces the risk of overfitting compared to a single decision tree.
- Feature Importance: Provides insights into which customer attributes are most influential in predicting responses.

Disadvantages :

- Computational Complexity: Training a large number of trees can be computationally expensive.
- Interpretability: Less interpretable compared to simpler models like logistic regression.
- Hyperparameter Tuning: Requires tuning parameters like the number of trees for optimal performance.

Applications with Example :

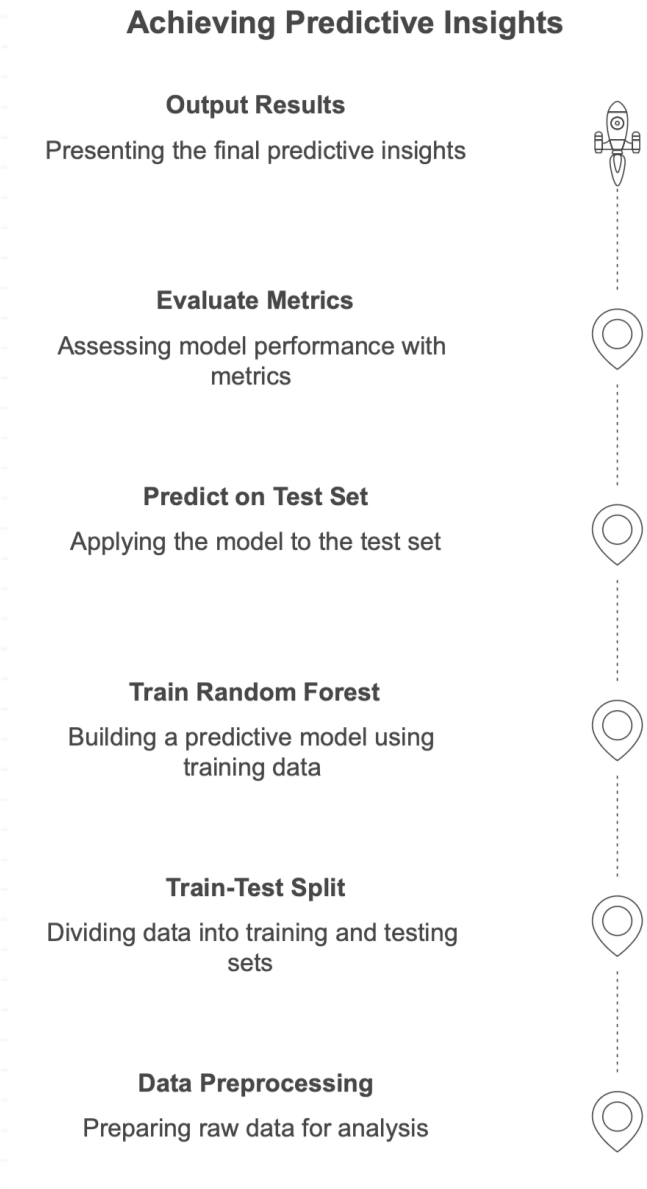
- Marketing Campaigns: Predicting which customers are likely to respond to offers, enabling targeted promotions. For example, a cosmetics shop can send special offers to customers predicted to accept, maximizing ROI.
- Customer Segmentation: Identifying customer groups based on their likelihood to engage with offers.
- Churn Prediction: Predicting which customers might stop purchasing, allowing proactive retention strategies.

Working/Algorithm :

- Load the dataset containing customer attributes like age, purchase history, loyalty score, and response.
- Extract features (X: age, purchase history, loyalty score) and target (y: response).
- Split the data into training (80%) and testing (20%) sets using `train_test_split`.
- Initialize a Random Forest Classifier with a fixed `random_state` for reproducibility.

- Train the model on the training data using the `fit` method.
- Predict customer responses on the test set using the `predict` method.
- Generate the confusion matrix to evaluate true positives, true negatives, false positives, and false negatives.
- Calculate accuracy, precision, recall, and F1 score to assess the model's predictive performance.

Diagram :



Result :

```
[11] print("Confusion Matrix:\n", cm)
     print(f"Accuracy: {acc:.2f}")
     print(f"Precision: {prec:.2f}")
     print(f"Recall: {rec:.2f}")
     print(f"F1 Score: {f1:.2f}")
```

```
Confusion Matrix:
 [[42 11]
 [17 10]]
Accuracy: 0.65
Precision: 0.48
Recall: 0.37
F1 Score: 0.42
```

- True Negatives (TN): 42 (customers correctly predicted to reject the offer).
- False Positives (FP): 11 (customers incorrectly predicted to accept).
- False Negatives (FN): 17 (customers incorrectly predicted to reject).
- True Positives (TP): 10 (customers correctly predicted to accept).

Conclusion :

The Random Forest Classifier was used to predict customer responses to a special offer in a cosmetics shop, achieving 65% accuracy with a precision of 0.48, recall of 0.37, and F1 score of 0.42. While the model gives a basic idea of customer behavior, the results suggest it may struggle due to factors like class imbalance or limited input features. To improve accuracy, adding more customer data, tuning model settings, and handling imbalance could help. Despite its limitations, the model can support better marketing by identifying customers more likely to accept offers.