Assignment: 7

Problem Statement:

Assignment on Classification technique

Every year many students give the GRE exam to get admission in foreign Universities. The

data set contains GRE Scores (out of 340), TOEFL Scores (out of 120), University Rating

(out of 5), Statement of Purpose strength (out of 5), Letter of Recommendation strength (out

of 5), Undergraduate GPA (out of 10), Research Experience (0=no, 1=yes), Admitted (0=no,

1=yes). Admitted is the target variable.

Data Set: https://www.kaggle.com/mohansacharya/graduate-admissions

The counselor of the firm is supposed to check whether the student will get an admission or

not based on his/her GRE score and Academic Score. So to help the counselor to take

appropriate decisions, build a machine learning model classifier using a Decision tree to

predict whether a student will get admission or not.

a) Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if

necessary.

b) Perform data-preparation (Train-Test Split)

c) Apply Machine Learning Algorithm

d) Evaluate Model.

Software Library Package:

Python with pandas , numpy and scikit-learn (DecisionTreeClassifier)

1. Theory:

Decision trees are simple yet powerful machine learning models that recursively split data based on features, creating a tree-like structure of decisions. They're used for classification and regression tasks, handling both numerical and categorical data. Decision trees are easy to interpret and visualize, making them valuable for understanding the decision-making process. However, they can overfit, which can be addressed with techniques like pruning or ensemble methods. Overall, decision trees offer a balance between interpretability and predictive performance, making them widely used in various domains.

1.1 Methodology:

The implemented program utilizes several functions and libraries to preprocess the data, train the model, and evaluate its performance.

1.2 Advantages and Applications:

Decision trees offer several advantages:

1. Interpretability: Decision trees are easy to understand and interpret, making them accessible to users without extensive technical knowledge.

2. Versatility: Decision trees can handle both numerical and categorical data, making them suitable for a wide range of applications.

3. Non-parametric: Decision trees make no assumptions about the underlying distribution of the data, providing flexibility in modeling complex relationships.

4. Feature Importance: Decision trees can identify the most important features for prediction, aiding in feature selection and understanding the underlying data patterns.

5. Efficiency: Decision trees have a fast training time compared to some other machine learning algorithms, making them efficient for large datasets.

6. Robustness to Outliers: Decision trees are robust to outliers and can handle noisy data without significantly impacting performance.

7. No Data Normalization Required: Decision trees do not require data normalization or scaling, simplifying preprocessing steps.

1.3 Limitations:

- Dependency on External Libraries: The program relies on external libraries, which may introduce dependencies and compatibility issues.
- Potential Overhead: Utilizing functions from libraries may introduce overhead in terms of memory and computational resources.

2. Working/Algorithm:

The implemented program consists of several functions:

- `read_csv: Reads the CSV file containing the dataset using pandas.
- `describe`: Provides descriptive statistics of the dataset.
- `info`: Prints information about the dataset, including column names, data types, and non-null counts.
- `shape`: Returns the shape (number of rows and columns) of the dataset.
- `isnull().sum()`: Calculates and prints the number of missing values for each column in the dataset.
- `fillna(0, inplace=True)`: Fills missing values with zeros in the dataset.
- `LabelEncoder()`: Encodes categorical variables into numerical labels.
- `train_test_split`: Splits the dataset into training and testing sets.
- `DecisionTreeRegressor`: Initializes a Decision Tree Regressor model.
- `fit`: Fits the model to the training data.
- `predict`: Predicts the target variable using the trained model.

- ` mean_squared_error` : Calculates the mean squared error between the predicted and actual target values.

- ` mean_absolute_error` : Calculates the mean absolute error between the predicted and actual target values.

- ` r2_score` : Calculates the R-squared score of the model.

3. Conclusion:

The program effectively utilizes functions from libraries like pandas, scikit-learn, and matplotlib to preprocess the data, train a Decision Tree Regressor model, and evaluate its performance. These functions enhance code efficiency, modularity, and reusability. However, it's essential to consider the limitations associated with external dependencies and potential overhead. Overall, the program provides a structured approach to building and evaluating machine learning models for regression tasks.