**Name** – Mrunal Talikoti

**PRN** – 22311431

**Roll No** – 282046

**Year** – SY B

--------------------------------------------------------------------------------------------------------------------------------------

**Assignment No – 7**

**Problem Statement**

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 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 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.

**Objective**

 Data Preprocessing

* Load dataset from Kaggle.
* Check for missing values and handle them.
* Create 'Admitted' (0/1) from 'Chance of Admit ': 1 if >= 0.8, else 0.
* Use 'GRE Score' and 'CGPA' as features; drop others if needed.
* Scale features if necessary.

 Data Preparation (Train-Test Split)

* Split data: 80% train, 20% test (X = ['GRE Score', 'CGPA'], y = 'Admitted').
* Use random\_state=42 for reproducibility.

 Apply Machine Learning Algorithm

* Train a DecisionTreeClassifier on training data.
* Fit model using 'GRE Score' and 'CGPA' to predict 'Admitted'.

 Evaluate Model

* Predict on test data.
* Calculate accuracy, precision, recall, and F1-score.
* Optionally visualize the tree.

**Methodology**

1. Data Pre-processing:
   * Load the dataset (admission.csv) using pandas.
   * Check for missing values and dataset information.
   * Convert the continuous target variable ("Chance of Admit") into a binary classification problem (0 or 1) using a threshold (0.7).
2. Data Preparation:
   * Drop unnecessary columns (e.g., "Serial No.") as they don’t contribute to prediction.
   * Split the dataset into features (x) and target (y).
   * Perform train-test splitting (80% train, 20% test) using train\_test\_split.
   * Apply feature scaling using StandardScaler to normalize the feature values.
3. Apply Machine Learning Algorithm:
   * Train a Decision Tree Classifier (DecisionTreeClassifier) on the scaled training data.
4. Evaluate Model:
   * Predict outcomes on the test set.
   * Evaluate performance using a confusion matrix and accuracy score.

**Main Functions :**

1. pd.read\_csv("admission.csv") (Pandas):
   * Loads the dataset from a CSV file into a DataFrame.
2. dataset.info() (Pandas):
   * Shows data types and non-null counts of the dataset.
3. dataset.describe() (Pandas):
   * Provides statistical summary (mean, std, etc.) of numerical columns.
4. dataset.isnull().sum() (Pandas):
   * Checks for missing values in each column.
5. np.concatenate() (NumPy):
   * Combines predicted and actual values into a single array for comparison.
6. train\_test\_split() (Scikit-learn):
   * Splits data into training (80%) and testing (20%) sets.
7. StandardScaler() (Scikit-learn):
   * Normalizes features by scaling to zero mean and unit variance.
8. DecisionTreeClassifier() (Scikit-learn):
   * Trains a Decision Tree model for classification.
9. classifier.predict() (Scikit-learn):
   * Generates predictions on the test set.
10. confusion\_matrix() (Scikit-learn):
    * Computes a matrix showing true vs. predicted classifications.
11. accuracy\_score() (Scikit-learn):
    * Calculates the accuracy of predictions.

**Advantages and Disadvantages of Libraries**

**Pandas**

* **Adv**: Easy data manipulation, fast loading/analysis.
* **Disadv**: Memory-heavy, limited for advanced modeling.

**NumPy**

* **Adv**: Fast array operations, lightweight.
* **Disadv**: Less intuitive for complex data tasks.

**Scikit-learn**

* **Adv**: Robust ML tools, simple to use.
* **Disadv**: Some functions (e.g., StandardScaler) redundant here, basic metrics may mislead.

**Conclusion:**

* **Libraries**: Pandas, NumPy, Scikit-learn.
* **Task**: Predict student admission using Decision Tree.
* **Strengths**: Easy data handling (Pandas), fast arrays (NumPy), robust ML tools (Scikit-learn).
* **Weaknesses**: Memory use (Pandas), overfitting risk (Decision Tree), scaling redundancy (StandardScaler), limited evaluation (accuracy alone).

