Investment Decision Recommendation System Project

NAME: KIRUTHIKA V G
ROLL NO: 21PT10

OBJECTIVE:

To build a recommendation system to predict the best investment decision for new data based on various factors available in the dataset.

1. DATA PREPROCESSING:

- Handling Missing Values and Duplicates using functions **dropna()** and **drop_duplicates().**
- **Feature engineering:**

  The **parse_percent** function is defined to convert percentage strings into numerical values.

  Two new columns, **Income Lower** and **Income Upper**, are created by extracting numerical values from the Household Income column.
- **Label Encoding:**

  The **LabelEncoder** from scikit-learn is used to convert categorical variables into numerical format.

  This includes columns such as City, Gender, Marital Status, Education, Role, Source of Awareness about Investment, Knowledge level about different investment product, Investment Influencer, and Reason for Investment.
- **Train_test_split:**

  The dataset is split into features (X) and the target variable (y) using the drop function to remove the target variable from the feature set. (extract output label - supervised learning)

  Further, the dataset is split into training and testing sets using the **train_test_split** function from scikit-learn.
- **Scaling features:**

  The numerical features are standardized using the **StandardScaler** from scikit-learn. This standardization ensures that all features have a mean of 0 and a standard deviation of 1, which helps in improving model performance.

- **Exploring Correlation Matrix:**
  Analyzing the correlation matrix helped identify relationships between numerical variables.

2. MODEL SELECTION:

- **Random forest:**
  The model used for this problem statement is **Random forest.**

  It handles label encoding easily and captures non-linear relationships between features and the target variable.

  It combines the predictions of multiple individual decision trees to improve the overall performance.

- **Training:**
  The RandomForestClassifier is trained on the training dataset using the fit() method.
  During training, each decision tree in the random forest is built using a random subset of the training data and a random subset of the features.

- Hyperparameters like n_estimators and max_depth can be tuned to optimize model performance.

- **Evaluation:**
  After training, the model's performance is evaluated using metrics such as accuracy, precision, recall, and F1-score on the test dataset.

  Output:

  ```
  Accuracy: 0.19753086419753085
  Precision: 0.17912922550278135
  Recall: 0.18949486978778785
  F1-score: 0.15941317941317942
  ```

3. APPLYING THE MODEL:

- The same preprocessing steps are applied to new data as done for the training data. (parsing percentages, encoding categorical variables, and scaling numerical features.)

- The trained model using random forest is used to predict outcomes on new data.