Assignment No. 8

Problem Statement: Classification using Ensemble Learning Techniques

Objective: To explore and apply Ensemble Learning methods such as Bagging, Boosting, and Random Forest for classification tasks. The goal is to improve model accuracy by combining multiple weak learners into a strong model.

Prerequisite:

1. **Python environment with libraries:** pandas, numpy, sklearn, matplotlib, seaborn.
2. **Classification dataset**
3. Basic understanding of decision trees, classification metrics, and ensemble learning strategies.

Theory:

1. Understanding the Dataset :

- Before building models, perform Exploratory Data Analysis (EDA) on the dataset:
- Dataset Dimensions: Use .shape to find number of records and features.
- Data Types: Use .dtypes to inspect feature types (categorical, numerical).
- Missing Values: Use .isnull().sum() to identify and handle missing data.
- Visualization: Use plots (e.g., pairplot, heatmap, countplot) to explore data patterns.

2. Data Preprocessing

- Handle missing values using appropriate imputation techniques.
- Encode categorical variables using LabelEncoder or OneHotEncoder.
- Feature scaling using StandardScaler if needed.
- Split dataset into training and testing sets (e.g., 80-20 split).

3. Ensemble Learning Techniques

a. Bagging (Bootstrap Aggregating)

- Train multiple base learners (e.g., Decision Trees) on random subsets of data.
- Aggregated result is based on voting (classification) or averaging (regression).

- Example: BaggingClassifier from sklearn.ensemble.

b. Random Forest

- An extension of Bagging that uses decision trees and selects random features at each split.
- Reduces variance and helps prevent overfitting.
- Provides feature importance metrics.

c. Boosting

Trains models sequentially, where each new model focuses on errors made by previous models.

Examples:

1. AdaBoostClassifier
2. GradientBoostingClassifier
3. XGBoost (popular third-party library)

4. Evaluation Metrics

Use the following to evaluate model performance:

- Accuracy
- Precision, Recall, F1-Score
- Confusion Matrix
- ROC Curve and AUC Score
- Compare these metrics across different ensemble techniques.

Code & Output

```
•[1]: import pandas as pd
      import numpy as np
      import seaborn as sns
      import matplotlib.pyplot as plt
```

```
[2]: from sklearn.model_selection import train_test_split, cross_val_score
     from sklearn.preprocessing import LabelEncoder
     from sklearn.impute import SimpleImputer
     from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
     from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
•[3]: train = pd.read_csv("/content/train_u6lujuX_CVtuZ9i.csv")
      test = pd.read_csv("/content/test_Y3wMUE5_7gLdaTN.csv")
```

Data Preprocessing

```
[4]: train['source'] = 'train'
     test['source'] = 'test'
     data = pd.concat([train, test], ignore_index=True)
```

```
[5]: # Fill categorical columns with mode
     for col in ['Gender', 'Married', 'Dependents', 'Self_Employed', 'Credit_History', 'Loan_Amount_Term']:
         data[col].fillna(data[col].mode()[0], inplace=True)

     # Fill numeric column
     data['LoanAmount'].fillna(data['LoanAmount'].median(), inplace=True)
```

```
[6]: le = LabelEncoder()
     for col in ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area', 'Dependents']:
         data[col] = le.fit_transform(data[col])
```

```
[7]: # Drop columns not needed
     data.drop(['Loan_ID'], axis=1, inplace=True)

     # Split data back
     train_data = data[data['source'] == 'train'].drop(['source'], axis=1)
     test_data = data[data['source'] == 'test'].drop(['source', 'Loan_Status'], axis=1)

     # Encode target
     train_data['Loan_Status'] = le.fit_transform(train_data['Loan_Status'])

     # Define X and y
     X = train_data.drop('Loan_Status', axis=1)
     y = train_data['Loan_Status']

     # Split
     X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

[8]:
```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

[8]:
```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

[9]:
```
gb = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, random_state=42)
gb.fit(X_train, y_train)
```

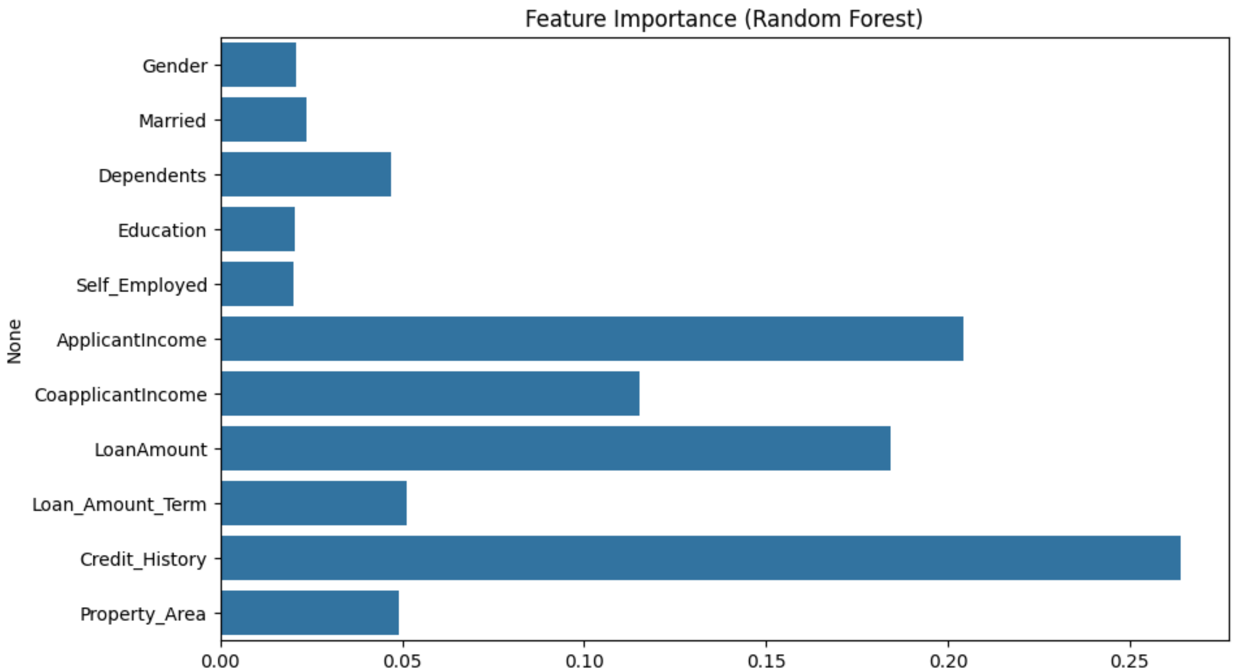
[9]:
```
GradientBoostingClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
Random Forest Accuracy: 0.7561
              precision    recall  f1-score   support

           0       0.78      0.42      0.55        43
           1       0.75      0.94      0.83        80

    accuracy                           0.76       123
   macro avg       0.77      0.68      0.69       123
weighted avg       0.76      0.76      0.73       123


Gradient Boosting Accuracy: 0.7398
              precision    recall  f1-score   support

           0       0.72      0.42      0.53        43
           1       0.74      0.91      0.82        80

    accuracy                           0.74       123
   macro avg       0.73      0.67      0.67       123
weighted avg       0.74      0.74      0.72       123


Voting Classifier Accuracy: 0.7642
              precision    recall  f1-score   support

           0       0.82      0.42      0.55        43
           1       0.75      0.95      0.84        80

    accuracy                           0.76       123
   macro avg       0.79      0.68      0.70       123
weighted avg       0.78      0.76      0.74       123
```

```
[12]: test_predictions = vc.predict(test_data)
      submission = pd.read_csv("test_Y3wMUE5_7gLdaTN.csv")
      submission['Loan_Status'] = le.inverse_transform(test_predictions)
      submission[['Loan_ID', 'Loan_Status']].to_csv("submission.csv", index=False)
```

```
[13]: importances = rf.feature_importances_
      feat_names = X.columns

      plt.figure(figsize=(10,6))
      sns.barplot(x=importances, y=feat_names)
      plt.title("Feature Importance (Random Forest)")
      plt.show()
```



Github : **https://github.com/Vishalgodalkar/Machine-Learning**

Conclusion:

This assignment demonstrated the implementation of various Ensemble Learning techniques to improve classification accuracy. It included:

- Preprocessing and preparing the dataset
- Applying Bagging, Random Forest, and Boosting methods
- Evaluating model performance using key metrics

Ensemble learning methods significantly improve prediction robustness and accuracy by combining multiple learners, making them ideal for practical machine learning applications.