CONTENT

ABSTRACT

The Loan Application Status Prediction project aims to predict whether a loan application will be approved based on various features such as income, credit history, loan amount, and more. By analyzing the dataset, we build predictive models using Logistic Regression, Random Forest, and Decision Tree classifiers. These models are evaluated using accuracy, confusion matrix, and classification reports to determine the most effective approach for predicting loan approval. The project demonstrates the potential of machine

learning in improving financial decision-making processes.

INTRODUCTION

This project focuses on predicting loan approval for applicants using machine learning techniques. The dataset contains information on applicants' income, credit history, loan amount, and demographic details. By leveraging this data, the goal is to create a model that can accurately predict whether a loan application will be approved. The project involves data preprocessing, model training, and evaluation to identify the best-performing model for this task.

TECHNOLOGIES USED

Programming Language:

- **Python**: Python was used for implementing machine learning models and data manipulation due to its simplicity and extensive library support.

Machine Learning Libraries:

- **Scikit-learn**: Provided tools for model building, data preprocessing, and evaluation.
- **Pandas**: Used for loading, preprocessing, and manipulating the dataset.
- **Seaborn & Matplotlib**: Utilized for data visualization and plotting graphs to understand data distribution and relationships between features.

Development Environment:

- **Google Colab**: Google Colab was used as the development environment, offering free access to powerful GPUs and a cloud-based interface. It facilitated easy collaboration and allowed for running computationally intensive tasks without needing local resources.

DATASET INFORMATION

The dataset contains information about applicants who applied for a loan, including:

- **Loan_ID**: A unique identifier for each loan.
- **Gender**: Gender of the applicant.
- **Married**: Marital status of the applicant.

- **Dependents**: Number of dependents of the applicant.
- **Education**: Education level of the applicant.
- **Self_Employed**: Whether the applicant is self-employed.
- **ApplicantIncome**: Income of the applicant.
- **CoapplicantIncome**: Income of the co-applicant (if any).
- **LoanAmount**: Amount of loan requested.
- **Loan_Amount_Term**: Term of the loan in months.
- **Credit_History**: Credit history of the applicant (1 = Good, 0 = Bad).
- **Property_Area**: The area where the property is located (Urban, Semi-Urban, Rural).
- **Loan_Status**: Whether the loan was approved or not (1 = Yes, 0 = No).

METHODOLOGY

The methodology for predicting loan approval involves the following steps:

Data Preprocessing:

- **Handling Missing Values**: Missing values in the Gender, Married, Dependents, Self_Employed, and Credit_History columns were filled with the mode, while numerical columns like LoanAmount and Loan_Amount_Term were filled with the median.
- **Encoding Categorical Variables**: Categorical variables like Gender, Married, Education, Self_Employed, and Property_Area were label encoded.
- **Feature Engineering**: A new feature TotalIncome was created by combining ApplicantIncome and CoapplicantIncome.
- **Feature Scaling**: Applied StandardScaler to normalize the data.

Model Selection:

- **Logistic Regression**: A simple and interpretable model suitable for binary classification tasks like loan approval prediction.
- **Random Forest**: An ensemble method that builds multiple decision trees to improve prediction accuracy.

- **Decision Tree**: A non-linear model that captures complex relationships between features.

Model Training and Evaluation:

- **Dataset Split**: The data was split into training (70%) and testing (30%) sets to evaluate model performance.
- **Model Evaluation**: The models were evaluated using accuracy, confusion matrix, and classification report.

Hyperparameter Tuning:

- **Grid Search CV**: Used for tuning Logistic Regression and Random Forest models to optimize performance.

CODE SNIPPETS

Importing Necessary Libraries:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from scipy.stats import zscore
```

Loading the Dataset:

```
# Load the dataset
df = pd.read_csv('/content/loan_prediction.csv')
df.head()
```

Data Preprocessing:

```
# Droping unnecessary column
df.drop(columns=['Loan_ID'], inplace=True)  # Dropping the identifier column
```

```
# Checking for missing values
print("Missing Values:\n", df.isnull().sum())
```

```
Missing Values:
 Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
TotalIncome          0
dtype: int64
```

```
# Fill missing values for categorical columns with mode and numerical columns with median
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0], inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].median(), inplace=True)
df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

Encoding categorical variables

```
from sklearn.preprocessing import LabelEncoder
```

```
# Here we are using LabelEncoder to convert categorical variables into numerical values
le = LabelEncoder()
categorical_columns = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Property_Area', 'Loan_Status']
for col in categorical_columns:
    df[col] = le.fit_transform(df[col])
```

Scaling the data using StandardScaler

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Feature Engineering:

```
# Adding TotalIncome column
# Here we are adding a new feature 'TotalIncome' by combining 'ApplicantIncome' and 'CoapplicantIncome'
df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']
```

```
# Dropping the original 'ApplicantIncome' and 'CoapplicantIncome' columns as they are now redundant
df.drop(['ApplicantIncome', 'CoapplicantIncome'], axis=1, inplace=True)
```

Model Training and Evaluation:

```
#Logistic Regression model evaluation
logreg = LogisticRegression(max_iter=500)
logreg.fit(X_train_scaled, y_train)
logreg_accuracy = evaluate_model(logreg, X_train_scaled, X_test_scaled, y_train, y_test, "Logistic Regression")
```

```
Logistic Regression Accuracy: 0.8286
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.41      0.58        51
           1       0.81      1.00      0.89       124

    accuracy                           0.83       175
   macro avg       0.90      0.71      0.74       175
weighted avg       0.86      0.83      0.80       175

Logistic Regression Confusion Matrix:
 [[ 21  30]
 [  0 124]]
```

```
# Random Forest model evaluation
rf = RandomForestClassifier()
rf_accuracy = evaluate_model(rf, X_train, X_test, y_train, y_test, "Random Forest")
```

```
Random Forest Accuracy: 0.8000
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.47      0.58        51
           1       0.81      0.94      0.87       124

    accuracy                           0.80       175
   macro avg       0.78      0.70      0.72       175
weighted avg       0.79      0.80      0.78       175

Random Forest Confusion Matrix:
 [[ 24  27]
 [  8 116]]
```

```
# Decision Tree model evaluation
dt = DecisionTreeClassifier()
dt_accuracy = evaluate_model(dt, X_train, X_test, y_train, y_test, "Decision Tree")
```

```
Decision Tree Accuracy: 0.7029
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.49      0.53      0.51        51
           1       0.80      0.77      0.79       124

    accuracy                           0.70       175
   macro avg       0.65      0.65      0.65       175
weighted avg       0.71      0.70      0.71       175

Decision Tree Confusion Matrix:
 [[27 24]
 [28 96]]
```

After Hyperparameter Tunning

```
# Logistic Regression Hyper Parameter Tunning
param_grid_logreg = {'C': [0.1, 1, 10, 500],'solver': ['liblinear', 'lbfgs'],'max_iter': [200, 500, 1000]}  # Increased i
grid_logreg = GridSearchCV(LogisticRegression(), param_grid_logreg, cv=5)
grid_logreg.fit(X_train_scaled, y_train)
best_logreg = grid_logreg.best_estimator_
logreg_accuracy_tuned = evaluate_model(best_logreg, X_train_scaled, X_test_scaled, y_train, y_test, "Tuned Logistic Regre
```

```
Tuned Logistic Regression Accuracy: 0.8286
Tuned Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.41      0.58        51
           1       0.81      1.00      0.89       124

    accuracy                           0.83       175
   macro avg       0.90      0.71      0.74       175
weighted avg       0.86      0.83      0.80       175

Tuned Logistic Regression Confusion Matrix:
 [[ 21  30]
 [  0 124]]
```

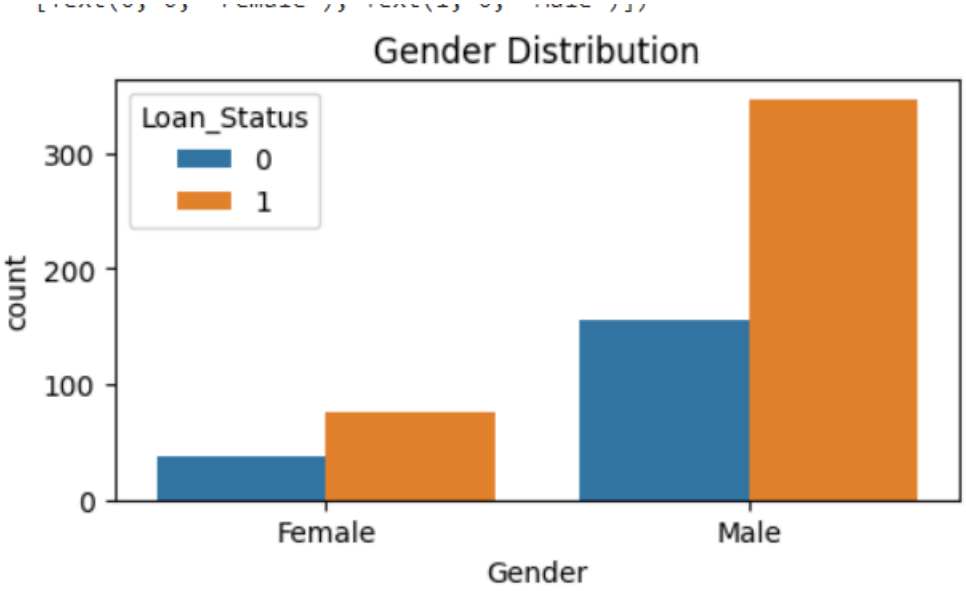
```
# Random Forest Hyperparameter Tuning
param_grid_rf = {'n_estimators': [50, 100, 200], 'max_depth': [10, 20, 30], 'min_samples_split': [2, 5, 10]}
grid_rf = GridSearchCV(RandomForestClassifier(), param_grid_rf, cv=5)
grid_rf.fit(X_train, y_train)
best_rf = grid_rf.best_estimator_
rf_accuracy_tuned = evaluate_model(best_rf, X_train, X_test, y_train, y_test, "Tuned Random Forest")
```

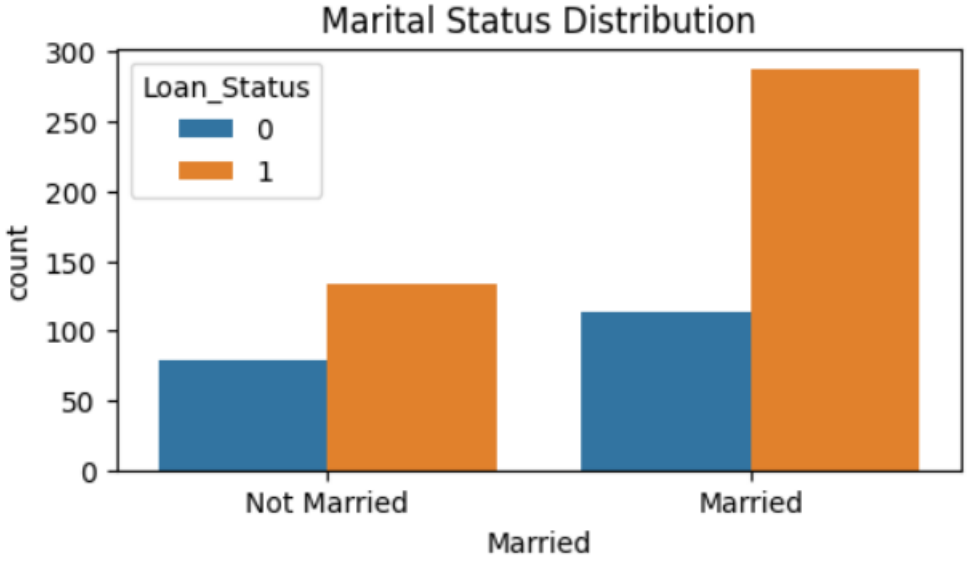
```
Tuned Random Forest Accuracy: 0.8171
Tuned Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.43      0.58        51
           1       0.81      0.98      0.88       124

    accuracy                           0.82       175
   macro avg       0.84      0.70      0.73       175
weighted avg       0.83      0.82      0.79       175

Tuned Random Forest Confusion Matrix:
 [[ 22  29]
 [  3 121]]
```

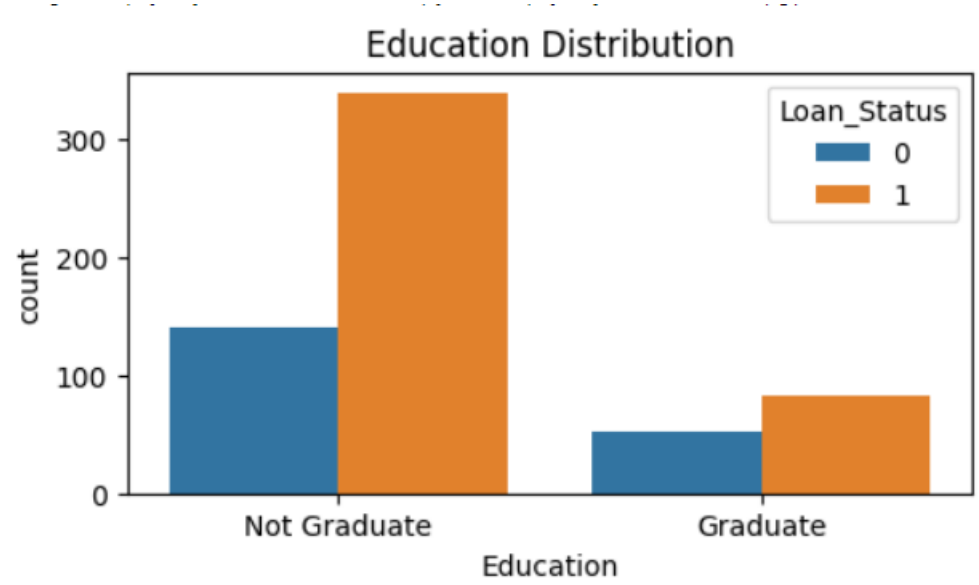
```
# Decision Tree Hyperparameter Tuning
param_grid_dt = {'max_depth': [10, 20, 30], 'min_samples_split': [2, 5, 10]}
grid_dt = GridSearchCV(DecisionTreeClassifier(), param_grid_dt, cv=5)
grid_dt.fit(X_train, y_train)
best_dt = grid_dt.best_estimator_
dt_accuracy_tuned = evaluate_model(best_dt, X_train, X_test, y_train, y_test, "Tuned Decision Tree")
```

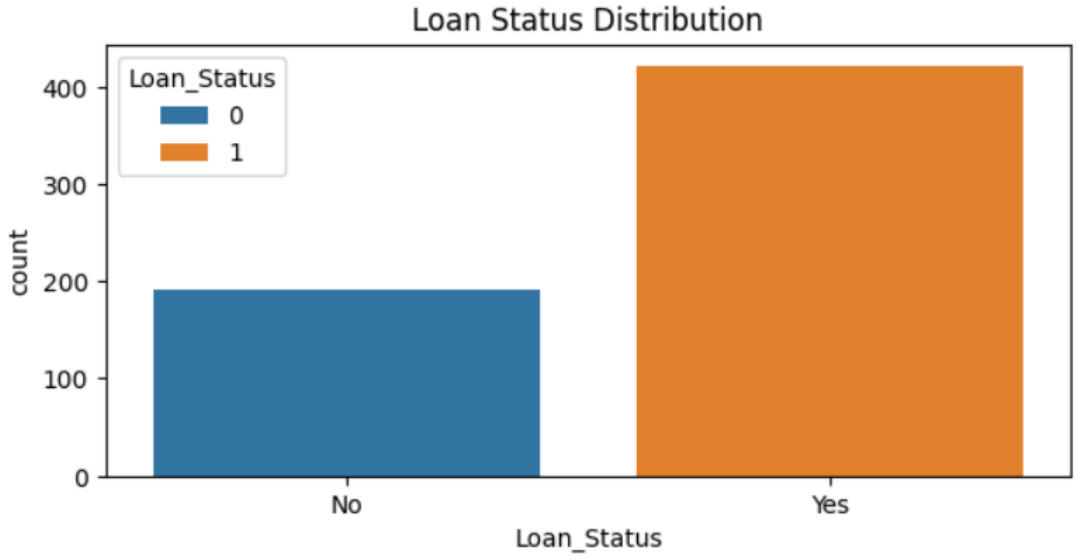
```
Tuned Decision Tree Accuracy: 0.7714
Tuned Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.57      0.59        51
           1       0.83      0.85      0.84       124

    accuracy                           0.77       175
   macro avg       0.72      0.71      0.72       175
weighted avg       0.77      0.77      0.77       175

Tuned Decision Tree Confusion Matrix:
 [[ 29  22]
 [ 18 106]]
```

Data Visualization:


Gender Distribution
Loan_Status
0
1
count
300
200
100
0
Female
Male
Gender


Marital Status Distribution
Loan_Status
0
1
count
300
250
200
150
100
50
0
Not Married
Married
Married





Result:

The results section presents the evaluation metrics for the three machine learning models used in the Loan Application Status Prediction project: Logistic Regression, Random Forest, and Decision Tree.

Logistic Regression:

- Accuracy: 82.86%
- Precision:
  - Class 0: 100%
  - Class 1: 81%
- Recall:
  - Class 0: 41%
  - Class 1: 100%
- F1-Score:
  - Class 0: 58%
  - Class 1: 89%
- Confusion Matrix:
  - True Positives (Class 0): 21
  - False Positives: 0
  - True Negatives (Class 1): 124
  - False Negatives: 30

Random Forest:

- Accuracy: 81.71%
- Precision:
  - Class 0: 88%
  - Class 1: 81%
- Recall:

    - Class 0: 43%
    - Class 1: 98%
- F1-Score:
    - Class 0: 58%
    - Class 1: 88%
- Confusion Matrix:
    - True Positives (Class 0): 22
    - False Positives: 3
    - True Negatives (Class 1): 121
    - False Negatives: 29

Decision Tree:

- Accuracy: 77.14%
- Precision:
    - Class 0: 62%
    - Class 1: 83%
- Recall:
    - Class 0: 57%
    - Class 1: 85%
- F1-Score:
    - Class 0: 59%
    - Class 1: 84%

- Confusion Matrix:
  - True Positives (Class 0): 29
  - False Positives: 18
  - True Negatives (Class 1): 106
  - False Negatives: 22

CONCLUSION

The Loan Application Status Prediction project successfully predicts whether a loan will be approved using various features such as income, credit history, and loan amount. The Logistic Regression model demonstrated the best performance, making it a suitable choice for similar classification problems. The project highlights the importance of feature selection, preprocessing, and model tuning in improving predictive accuracy.