Week 2.2
Using Python Tools for Building and Evaluating Decision Tree and Random Forest Models

Produced By: Abdullah Abdul Majid*(linkedin)*

Completion Date: July 13, 2024

Email: abdullahpisk@gmail.com

Jupyter Notebook Links:

GitHub

Google Drive

Table of Contents:

Problem Statement:

Learn to implement tree-based methods for classification and compare their performance.
Dataset Used: Loan Prediction Dataset from Kaggle

Purpose:

The purpose of this assignment is to inspect and preprocess the dataset. Then use the tools from the scikit-learn library to train Decision Tree and Random Forest models and compare their performance.

Process:

Dataset Loading and Initial Inspection

- Imported all the necessary libraries, i.e pandas, matplotlib and sklearn.
- Downloaded the Loan Prediction dataset from Kaggle.
- Loaded the dataset into a pandas dataframe (loan) using *read_csv* on the *train.csv* file.
- Removed the LoanID column since its not useful.
- Verified the dataset loading using print.

Inspecting and Preprocessing the Dataset

- Inspected the dataset using *info* and *describe*.
- No duplicates were found when *isnull()* was used.
- Null values were found and dealt with appropriately.

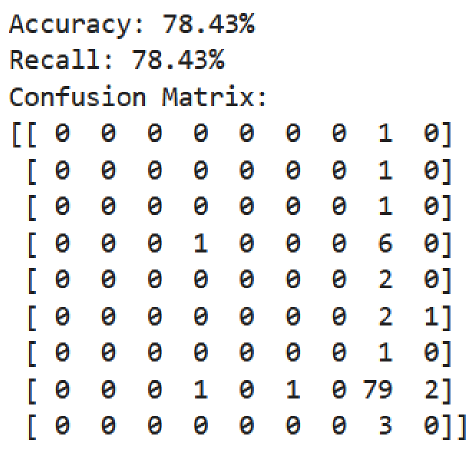
- Used boxplots to check for outliers, a potential outlier in the 'ApplicantIncome" column was terminated using Boolean Indexing.

Building and Training the Decision Tree Model

- Identified dependent (y), i.e "LoanAmount" column and independent (x) variables.
- Split the dataset into test and train sets with a 30-70 split respectively and a random state of 42.
- Used the Decision Tree Model as the model classifier.
- Next up, we train the model by fitting in the train values using *fit.*

Model Testing and Evaluation 1

- We *predict* the dependent 'y' variable using the independent 'x' test values to evaluate the model.
- Now, we compare the predicted and test data to verify that they are similar using *print*.
- Next, we calculate the *accuracy*,and *recall* (*weighted*) *scores*, *F1-score* and print the *confusion_matrix* which are as follows:



```
Accuracy: 78.43%
Recall: 78.43%
Confusion Matrix:
[[ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  1  0  0  0  6  0]
 [ 0  0  0  0  0  0  0  2  0]
 [ 0  0  0  0  0  0  0  2  1]
 [ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  1  0  1  0 79  2]
 [ 0  0  0  0  0  0  0  3  0]]
```

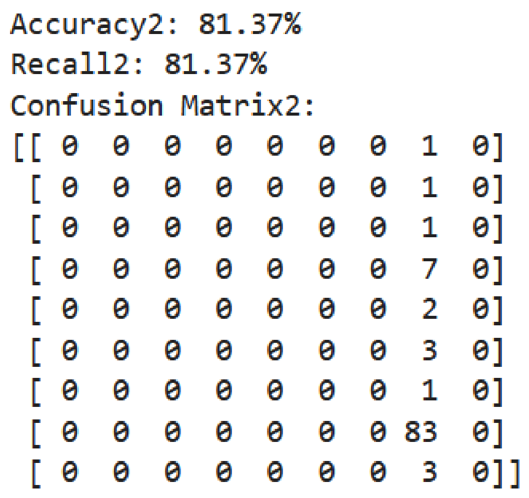
The results are good.

Building and Training the Random Forest Model

- Used same test/train setup as in the previous model.
- Used the Random Forest Model as the model classifier.
- Next up, we train the model by fitting in the train values using *fit.*

Model Testing and Evaluation 2

- We *predict* the dependent 'y2' variable using the independent 'x' test values to evaluate the model.
- Now, we compare the predicted and test data to verify that they are similar using *print*.
- Next, we calculate the *accuracy*,and *recall* (*weighted*) *scores*, *F1-score* and print the *confusion_matrix* which are as follows:



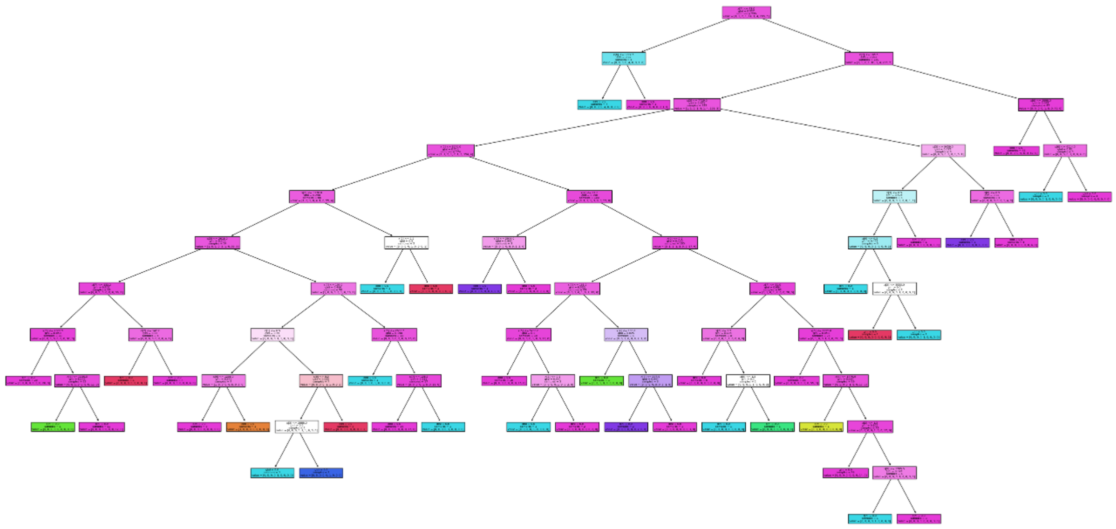
```
Accuracy2: 81.37%
Recall2: 81.37%
Confusion Matrix2:
[[ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  7  0]
 [ 0  0  0  0  0  0  0  2  0]
 [ 0  0  0  0  0  0  0  3  0]
 [ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0 83  0]
 [ 0  0  0  0  0  0  0  3  0]]
```

The results are better.

- Bothe models had an F1-Score of 0.73.

Decision Tree Plotting

- Used *plot_tree* and *matplotlib* to plot the Decision Tree which is as follows:



- A better figure is provided in the GitHub directory.

Conclusion:

I was able to preprocess and prepare a dataset for the Decision Tree and Random Forest model training and building using sklearn from scikit. The model's performance was also evaluated and was proved to be satisfactory, and the latter model proved to be superior. A figure of the Decision Tree has also been plotted.