FINAL TERM PROJECT

CS 634: DATA MINING
OPTION 1: SUPERVISED DATA MINING (CLASSIFICATION)

VISHAL BRAHMBHATT
NJIT UCID: vdb2
vdb2@njit.edu

DR. JASON WANG
DEPARTMENT OF COMPUTER SCIENCE
NEW JERSEY INSTITUTE OF TECHNOLOGY

Contents

Introduction

- The aim of the project (option 1) is to apply two algorithms and compare them with each other to find out the more accurate one.
- The algorithms were applied on one dataset.
- By doing so we could know which one is more efficient algorithm.

Algorithms Used:

a) Random Forest (Category 2)
b) Decision Tree (Category 3)

Software Tools Used:

a) Scikit-learn (Category 10)

Programming Language Used:

a) Python

System Requirements:

Software Requirements:

a) Python
b) Jupyter Notebook

Dataset Description

Title:

Loan Prediction Problem Dataset

URL:

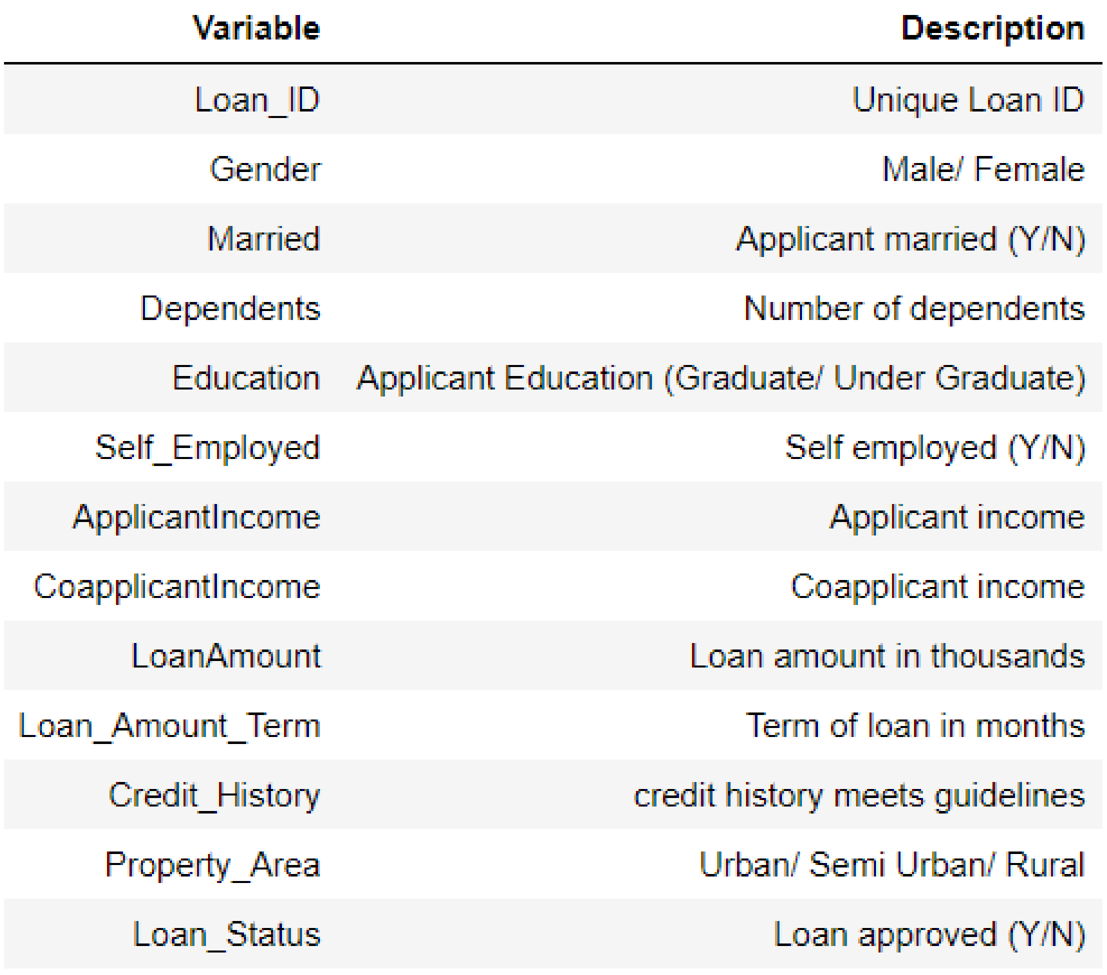
https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset

Description:

This dataset contains 614 unique values which consists of details like Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History, and others.

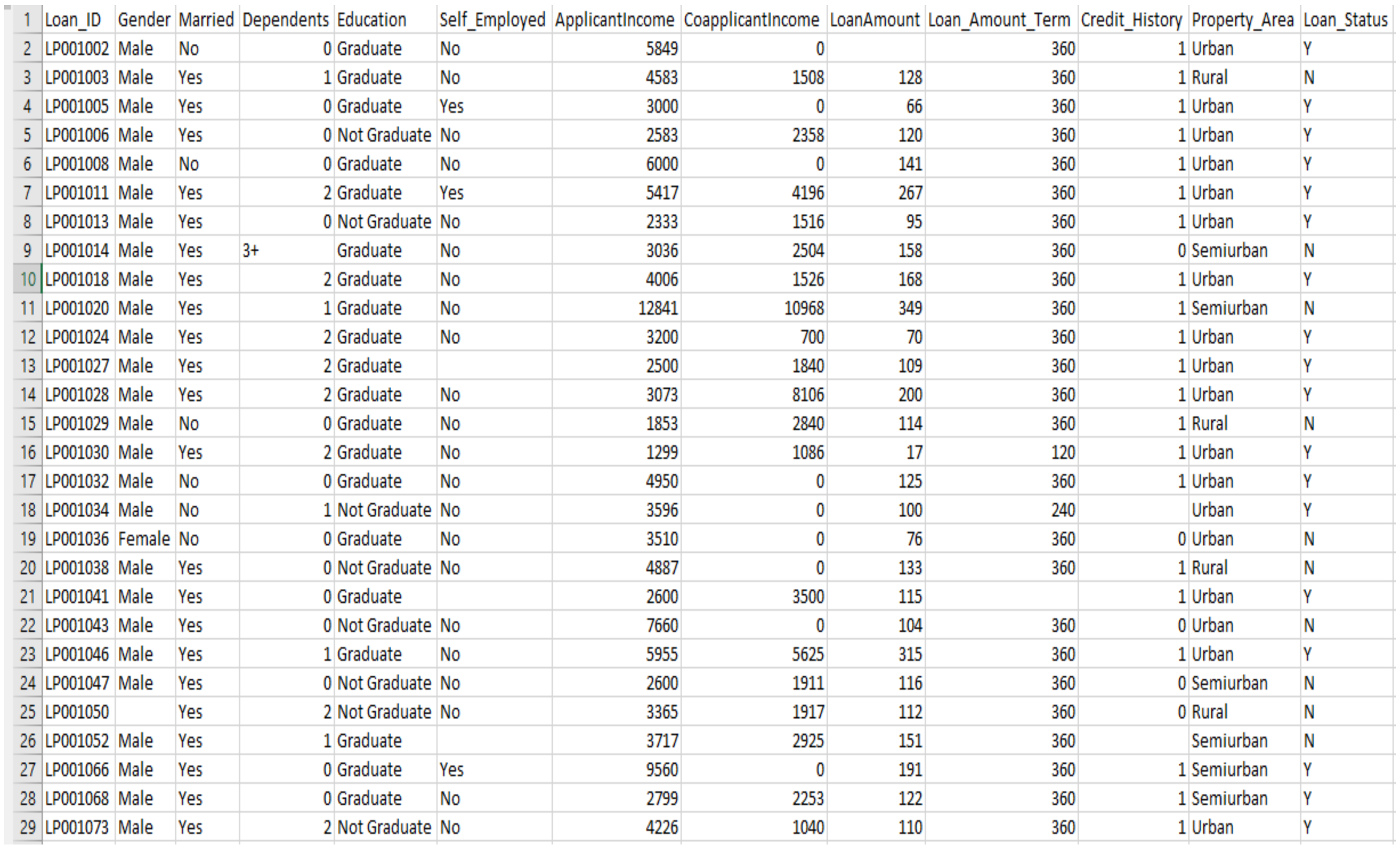
Attributes:

The attributes in the dataset are as follows:



Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural
Loan_Status	Loan approved (Y/N)

Attribute	Datatype
Loan_ID	Object
Gender	Object
Married	Object
Dependents	Object
Education	Object
Self_Employed	Object
ApplicantIncome	Integer
CoapplicantIncome	Float
LoanAmount	Float
Loan_Amount_Term	Float
Credit_History	Float
Property_Area	Object
Loan_Status	Object



1	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
5	LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1	Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
8	LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360	1	Urban	Y
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
13	LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
18	LP001034	Male	No	1	Not Graduate	No	3596	0	100	240		Urban	Y
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
20	LP001038	Male	Yes	0	Not Graduate	No	4887	0	133	360	1	Rural	N
21	LP001041	Male	Yes	0	Graduate		2600	3500	115		1	Urban	Y
22	LP001043	Male	Yes	0	Not Graduate	No	7660	0	104	360	0	Urban	N
23	LP001046	Male	Yes	1	Graduate	No	5955	5625	315	360	1	Urban	Y
24	LP001047	Male	Yes	0	Not Graduate	No	2600	1911	116	360	0	Semiurban	N
25	LP001050		Yes	2	Not Graduate	No	3365	1917	112	360	0	Rural	N
26	LP001052	Male	Yes	1	Graduate		3717	2925	151	360		Semiurban	N
27	LP001066	Male	Yes	0	Graduate	Yes	9560	0	191	360	1	Semiurban	Y
28	LP001068	Male	Yes	0	Graduate	No	2799	2253	122	360	1	Semiurban	Y
29	LP001073	Male	Yes	2	Not Graduate	No	4226	1040	110	360	1	Urban	Y

Data in CSV file

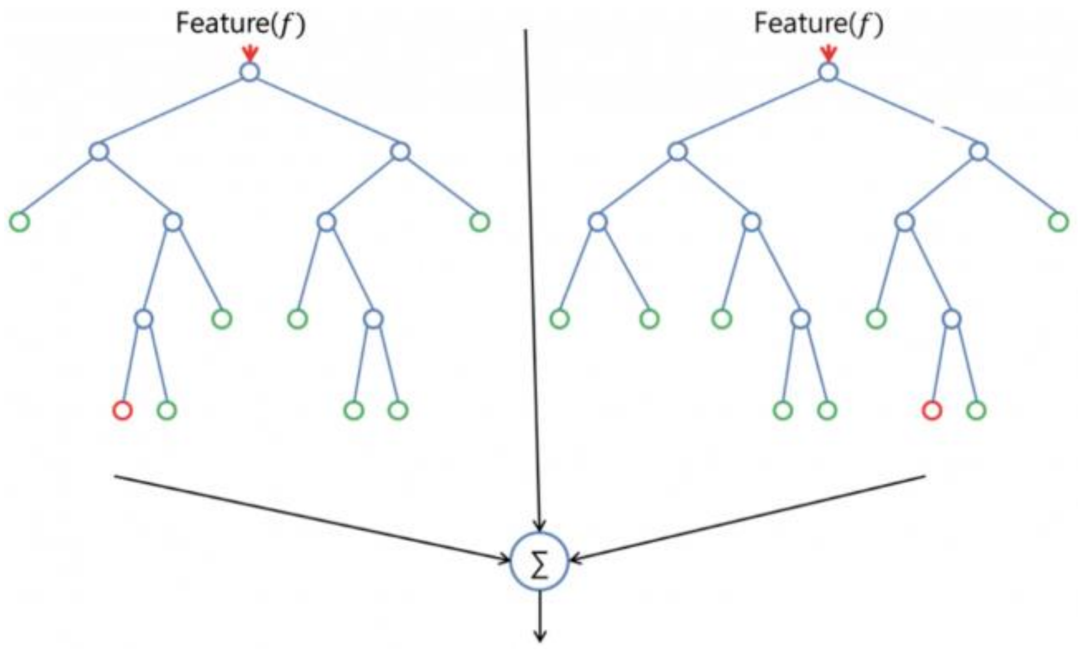
Algorithms

There are various algorithms used for various purposes. The algorithms are Support Vector Machines (SVM), Random Forests, Decision Trees, Bayesian Networks, Naïve Bayes, K-Nearest Neighbors and many more.

The algorithm used for this project are as follows:

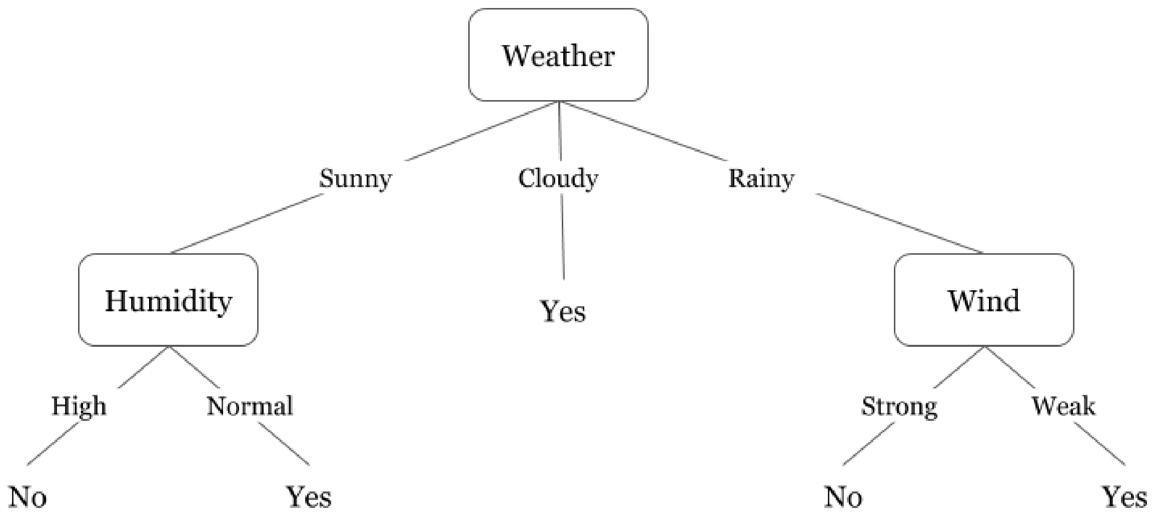
Random Forests:

- Random forest is a supervised learning algorithm.
- The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method.
- The general idea of the bagging method is that a combination of learning models increases the overall result.
- One big advantage of random forest is that it can be used for both classification and regression problems, which form most current machine learning systems.
- Random forest in classification since classification is sometimes considered the building block of machine learning.
- Below you can see how a random forest would look like with two trees:



Decision Trees:

- Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression.
- DecisionTreeClassifier is a class capable of performing multi-class classification on a dataset.
- Decision trees can also be applied to regression problems, using the DecisionTreeRegressor class.
- The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- A tree can be seen as a piecewise constant approximation.
- For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules.
- The deeper the tree, the more complex the decision rules and the fitter the model.



Accuracy, Cross-Validation and K-Fold

Accuracy:

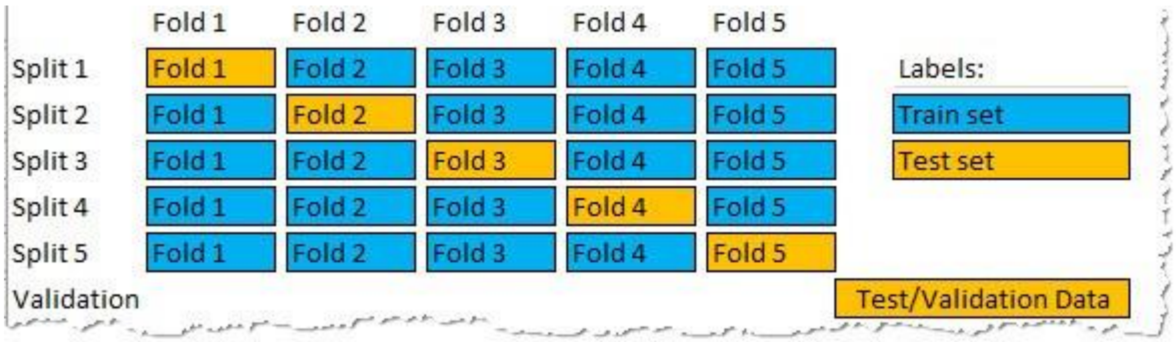
- Accuracy is defined as the percentage of correct predictions for the test data.
- It can be calculated easily by dividing the number of correct predictions by the number of total predictions.
- $Accuracy = \frac{Correct\ Prediction}{All\ prediction}$

Cross-Validation:

- Cross-validation is an important concept in machine learning which helps the data scientists in two major ways:
- It can reduce the size of data.
- It ensures that the model is robust enough.
- Cross validation does that at the cost of resource consumption.

Benefit 1: Data size reduction:

- Normally you split the data into 3 sets.
- **Training**: used to train the model and optimize the model's hyperparameters
- **Testing**: used to check that the optimized model works on unknown data to test that the model generalizes well.
- **Validation**: during optimizing some information about test set leaks into the model by your choice of the parameters so you perform a final check on completely unknown data.



Example of a 5-fold cross-validation data split

Benefit 2: Robust process:

- Even though sklearn's train_test_split method is using a stratified split, which means that the train and test set have the same distribution of the target variable, it is possible that you accidentally train on a subset which doesn't reflect the real world.
- Imagine that you try to predict whether a person is a male or a female by his or her height and weight.
- One would assume that taller and heavier people would rather be males, though if you are very unlucky your train data would only contain dwarf men and tall women.
- Thanks to cross validation you perform multiple train_test split and while one-fold can achieve extraordinarily good results the other might underperform.
- Anytime one of the splits shows unusual results it means that there is an anomaly in your data.

K-Fold:

- K-Folds cross-validator.
- Provides train/test indices to split data in train/test sets. Split dataset into k consecutive folds (without shuffling by default).
- K-Fold divides all the samples in groups of samples, called folds (if k = n, this is equivalent to the Leave One Out strategy), of equal sizes (if possible).
- The prediction function is learned using k-1 folds, and the fold left out is used for test.

Libraries

The libraries used for implementing the project are as follows:

a) Pandas:

- Pandas is a data manipulation and research software library written for the Python programming language.
- It provides data structures and operations for controlling numerical tables and time series.
- It is free software distributed under the BSD three-clause license.
- A fast and efficient DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format.
- Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form.
- Flexible reshaping and pivoting of data sets.
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets.
- Columns can be inserted and deleted from data structures for size mutability.
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets.
- High performance merging and joining of data sets.
- Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data.
- Highly optimized for performance, with critical code paths written in Cython or C.

- Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

b) NumPy:

- NumPy stands for Numerical Python.
- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- NumPy has the following features:
  1. POWERFUL N-DIMENSIONAL ARRAYS:
     - Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.
  2. NUMERICAL COMPUTING TOOLS:
     - NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.
  3. INTEROPERABLE:
     - NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.
  4. PERFORMANT:
     - The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.
  5. EASY TO USE:

     NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

  6. OPEN SOURCE:
     - Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

c) Seaborn:

- Seaborn is a library for making statistical graphics in Python.
- It builds on top of matplotlib and integrates closely with pandas data structures.
- Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.
- Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

d) Matplotlib:

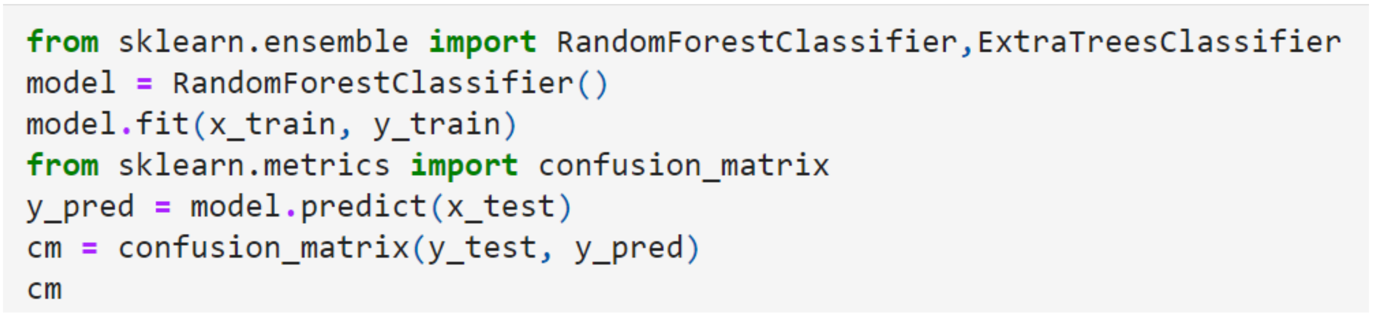
- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.
- Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits.
- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.
- Matplotlib consists of several plots like line, bar, scatter, histogram etc.
- Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.
- It was introduced by John Hunter in the year 2002.

e) Sklearn:

- Scikit-learn is a free machine learning library for Python.
- It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.
- In this tutorial we will learn to code python and apply Machine Learning with the help of the scikit-learn library, which was created to make doing machine learning in Python easier and more robust.

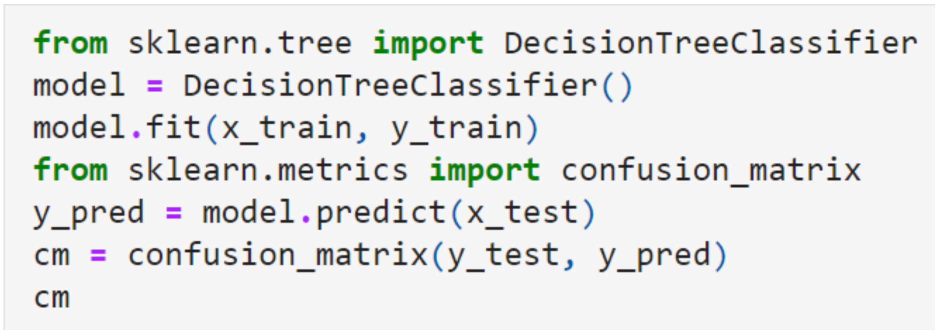
Implementing Algorithms and Functions:

Implementing Random Forest Classifier:



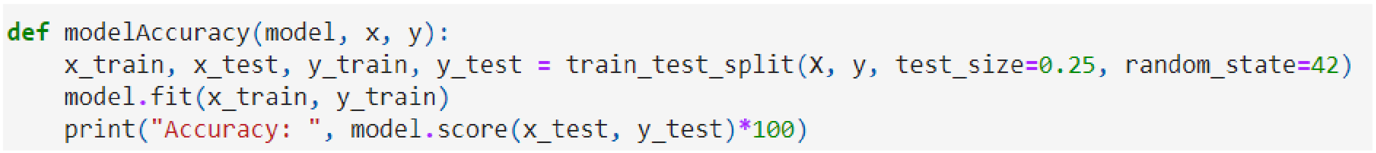
```
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

Implementing Decision Trees Classifier:



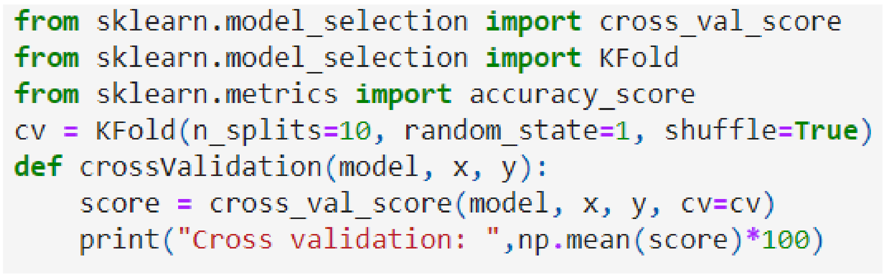
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

Implementing Accuracy Function:



```
def modelAccuracy(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
    model.fit(x_train, y_train)
    print("Accuracy: ", model.score(x_test, y_test)*100)
```

Implementing Cross Validation Function:

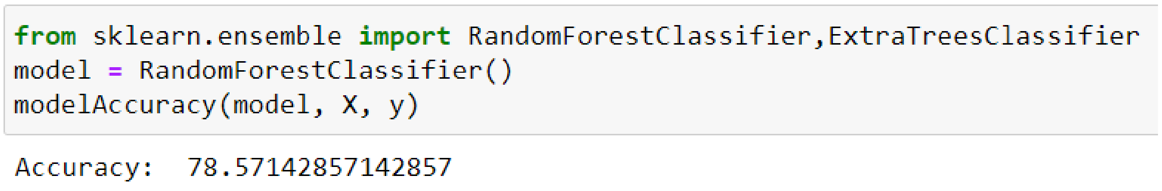


```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
cv = KFold(n_splits=10, random_state=1, shuffle=True)
def crossValidation(model, x, y):
    score = cross_val_score(model, x, y, cv=cv)
    print("Cross validation: ",np.mean(score)*100)
```

Calculating Accuracies and Cross validations of the algorithms

Accuracy:

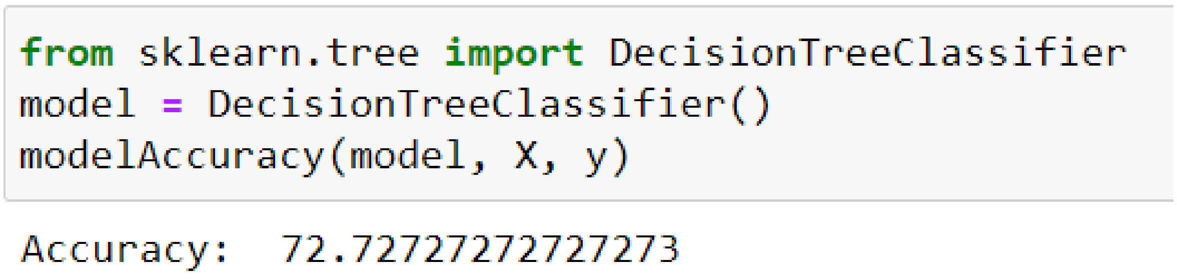
a) Random Forest Accuracy:



```
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier
model = RandomForestClassifier()
modelAccuracy(model, X, y)
```

```
Accuracy:   78.57142857142857
```

b) Decision trees Accuracy:

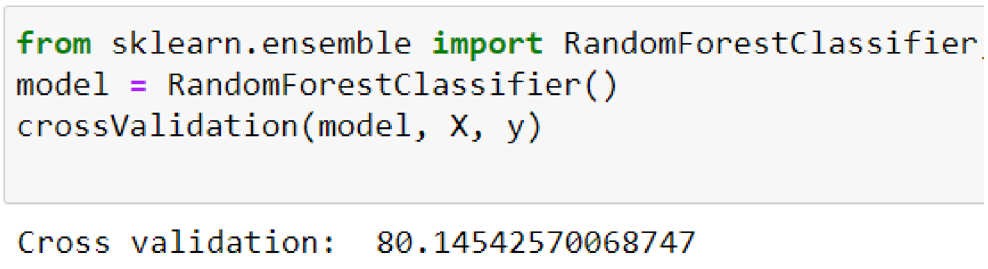


```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
modelAccuracy(model, X, y)
```

```
Accuracy:   72.72727272727273
```

Cross validation:

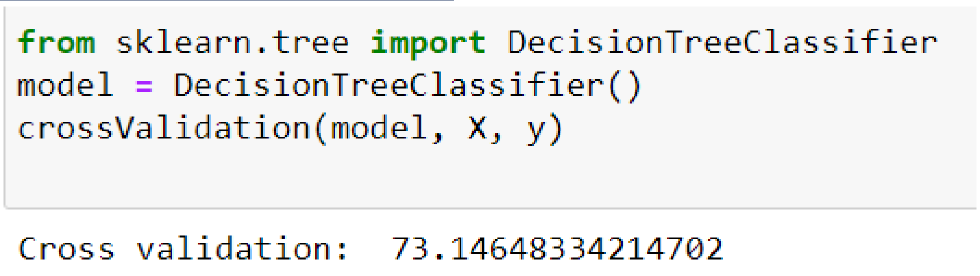
a) Random Forest Cross validation:



```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
crossValidation(model, X, y)
```

```
Cross validation:   80.14542570068747
```

b) Decision trees Cross validation:



```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
crossValidation(model, X, y)
```

```
Cross validation:   73.14648334214702
```

Comparing Accuracies and Cross validations of the algorithms

Accuracy:

Random Forest vs Decision Trees:

On comparing the accuracies of both the classifiers we could say that the accuracy of Random Forest Classifier is more than that of Decision Trees Classifier.

The accuracies are as follow:

Random Forest Classifier is 78.57.

Decision trees Classifier is 72.72.

Therefore, Accuracy of Random Forests > Accuracy of Decision Trees (Since 78.57 > 72.72)

Cross validation:

Random Forest vs Decision Trees:

On comparing the 10-fold cross validation method of both the classifiers we could say that the value of cross validation of Random Forest Classifier is more than that of Decision Trees Classifier.

The values are as follow:

Random Forest Classifier is 80.14.

Decision trees Classifier is 73.14.

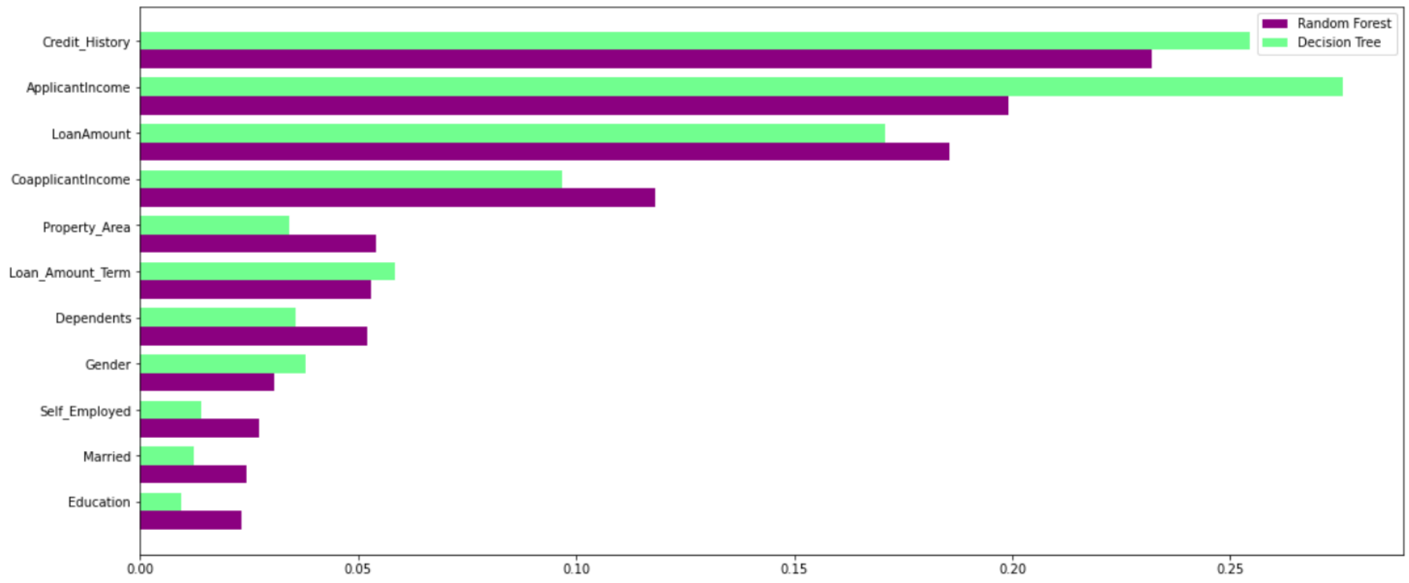
Therefore, The value of cross validation of Random Forests > The value of cross validation of Decision Trees (Since 80.14 > 73.14)

Conclusion

- As from the experimental results we can conclude that the Random Forest classifier is better than the Decision Trees Classifier in terms of accuracy and 10-fold cross validations.
- As the values of accuracies and cross validations of Random Forest Classifier was greater than the values of accuracies and cross validations of the Decision Trees Classifier.

Why Did Our Random Forest Model Outperform the Decision Tree?

- Random forest leverages the power of multiple decision trees. It does not rely on the feature importance given by a single decision tree.



- As you can clearly see in the above graph, the decision tree model gives high importance to a particular set of features.
- But the random forest chooses features randomly during the training process. Therefore, it does not depend highly on any specific set of features.
- This is a special characteristic of random forest over bagging trees.
- Therefore, the random forest can generalize over the data in a better way.
- This randomized feature selection makes random forest much more accurate than a decision tree.

Source Code

```python
## Import modules
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import matplotlib
get_ipython().run_line_magic('matplotlib', 'inline')

## Loading the dataset
df = pd.read_csv("Loan Prediction Dataset.csv")
df.head()
df.describe()
df.info()

## Preprocessing the dataset
# Find the null values
df.isnull().sum()
# Fill the missing values for numerical terms - mean
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term'] =
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
df['Credit_History'] =
df['Credit_History'].fillna(df['Credit_History'].mean())
# fill the missing values for categorical terms - mode
df['Gender'] = df["Gender"].fillna(df['Gender'].mode()[0])
df['Married'] = df["Married"].fillna(df['Married'].mode()[0])
df['Dependents'] = df["Dependents"].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] =
df["Self_Employed"].fillna(df['Self_Employed'].mode()[0])
df.isnull().sum()

## Exploratory Data Analysis
# categorical attributes visualization
sns.countplot(df['Gender'])
sns.countplot(df['Married'])
sns.countplot(df['Dependents'])
sns.countplot(df['Education'])
sns.countplot(df['Self_Employed'])
sns.countplot(df['Property_Area'])
sns.countplot(df['Loan_Status'])
# numerical attributes visualization
sns.distplot(df["ApplicantIncome"])
sns.distplot(df["CoapplicantIncome"])
sns.distplot(df["LoanAmount"])
sns.distplot(df['Loan_Amount_Term'])
sns.distplot(df['Credit_History'])

## Creation of new attributes
# total income
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df.head()
# apply log transformation to the attribute
df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome'])
sns.distplot(df["ApplicantIncomeLog"])
```

```
df['CoapplicantIncomeLog'] = np.log(df['CoapplicantIncome'])
sns.distplot(df["ApplicantIncomeLog"])
df['LoanAmountLog'] = np.log(df['LoanAmount'])
sns.distplot(df["LoanAmountLog"])
df['Loan_Amount_Term_Log'] = np.log(df['Loan_Amount_Term'])
sns.distplot(df["Loan_Amount_Term_Log"])
df['Total_Income_Log'] = np.log(df['Total_Income'])
sns.distplot(df["Total_Income_Log"])

## Coorelation Matrix
corr = df.corr()
plt.figure(figsize=(15,10))
sns.heatmap(corr, annot = True, cmap="BuPu")
df.head()
# drop unnecessary columns
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount",
"Loan_Amount_Term", "Total_Income", 'Loan_ID', 'CoapplicantIncomeLog']
df = df.drop(columns=cols, axis=1)
df.head()

## Label Encoding
from sklearn.preprocessing import LabelEncoder
cols =
['Gender',"Married","Education",'Self_Employed',"Property_Area","Loan_Status"
,"Dependents"]
le = LabelEncoder()
for col in cols:
    df[col] = le.fit_transform(df[col])
df.head()

## Train-Test Split
# specify input and output attributes
X = df.drop(columns=['Loan_Status'], axis=1)
y = df['Loan_Status']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

## Model Training
# Validation and Accuracy function
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
cv = KFold(n_splits=10, random_state=1, shuffle=True)
def crossValidation(model, x, y):
    score = cross_val_score(model, x, y, cv=cv)
    print("Cross validation: ",np.mean(score)*100)

def modelAccuracy(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
    model.fit(x_train, y_train)
    print("Accuracy: ", model.score(x_test, y_test)*100)

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
crossValidation(model, X, y)
```

```
modelAccuracy(model, X, y)

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm

from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier
model = RandomForestClassifier()
crossValidation(model, X, y)
modelAccuracy(model, X, y)

from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)

from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
model = ExtraTreesClassifier()
crossValidation(model, X, y)


## Hyperparameter tuning
model = RandomForestClassifier(n_estimators=100, min_samples_split=25,
max_depth=7, max_features=1)
crossValidation(model, X, y)

## Confusion Matrix
model = RandomForestClassifier()
model.fit(x_train, y_train)
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
sns.heatmap(cm, annot=True)
```

References

a) https://www.jeremyjordan.me/evaluating-a-machine-learning-model/
b) https://towardsdatascience.com/complete-guide-to-pythons-cross-validation-with-examples-a9676b5cac12
c) https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
d) https://en.wikipedia.org/wiki/Pandas_(software)
e) https://numpy.org/
f) https://www.w3schools.com/python/numpy/numpy_intro.asp
g) https://seaborn.pydata.org/introduction.html
h) https://www.geeksforgeeks.org/python-introduction-matplotlib/
i) https://pypi.org/project/matplotlib/
j) https://www.dataquest.io/blog/sci-kit-learn-tutorial/
k) https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/