APPENDIX 1

CLASSIFICATION OF REAL AND FAKE BANK NOTES

PROJECT REPORT

CA1

Submitted by

CECILIA SAGWATI

12020304

INT578: ADVANCED MACHINE LEARNING

Submitted to

DR. DHANPRATAP SINGH

FEBRUARY, 2022

APPENDIX 2

TABLE OF CONTENTS

APPENDIX 3

DECLARATION

I, Cecilia Sagwati, hereby declare that the work presented herein is genuine work originally done by me and it has not been published or submitted elsewhere for the requirement of a degree programme.

APPENDIX 4

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. Dhanpratap Singh of Lovely Professional University, for teaching me machine learning language which enabled me to undertake this project. Machine Learning knowledge is very important because in the society that we live in, there are so many complex problems that can be solved easily by machine learning algorithms.

APPENDIX 5

LIST OF FIGURES

LIST OF ABBREVIATIONS

SVC	Support Vector Machine Model
KNN	K-Nearest Neighbor Model
IDE	Integrated Development Environment
PPTN	Perceptron model
LR	Logistic Regression Model
DT	Decision Tree Model

CHAPTER1

TITLE: CLASSIFICATION OF REAL AND FAKE BANK NOTES

1.0 INTRODUCTION

1.1 ABSTRACT AND OBJECTIVE

This is a supervised machine learning project regarding authenticating banknotes. It is a binary classification, and the main objective of the project is to classify real and fake banknotes denoted by 0 and 1 in the dataset. When depositing money in the bank, the bank teller loads the money in a machine to verify whether a banknote is real or fake. This verification task can also be performed by a machine learning algorithm. This project report presents how machine learning models were trained to classify real and fake banknotes.

The dataset which I used to train the models was downloaded from UCI online repository through web link https://archive.ics.uci.edu/ml/machine-learning-databases/00267/.
This dataset contains 4 features that were extracted from images of real and fake banknotes. The features were digitalized by an industrial camera, and wavelet transform tool, was used to extract the features from the said banknotes' images. The features of the banknote dataset are:

- Variance of wavelet of transformed image denoted as **variance of wavelet**
- Skewness of wavelet of transformed image denoted as **skewness of wavelet**
- curtosis of wavelet of transformed image denoted as **curtosis of wavelet**
- entropy of transformed image denoted as **entropy of image**

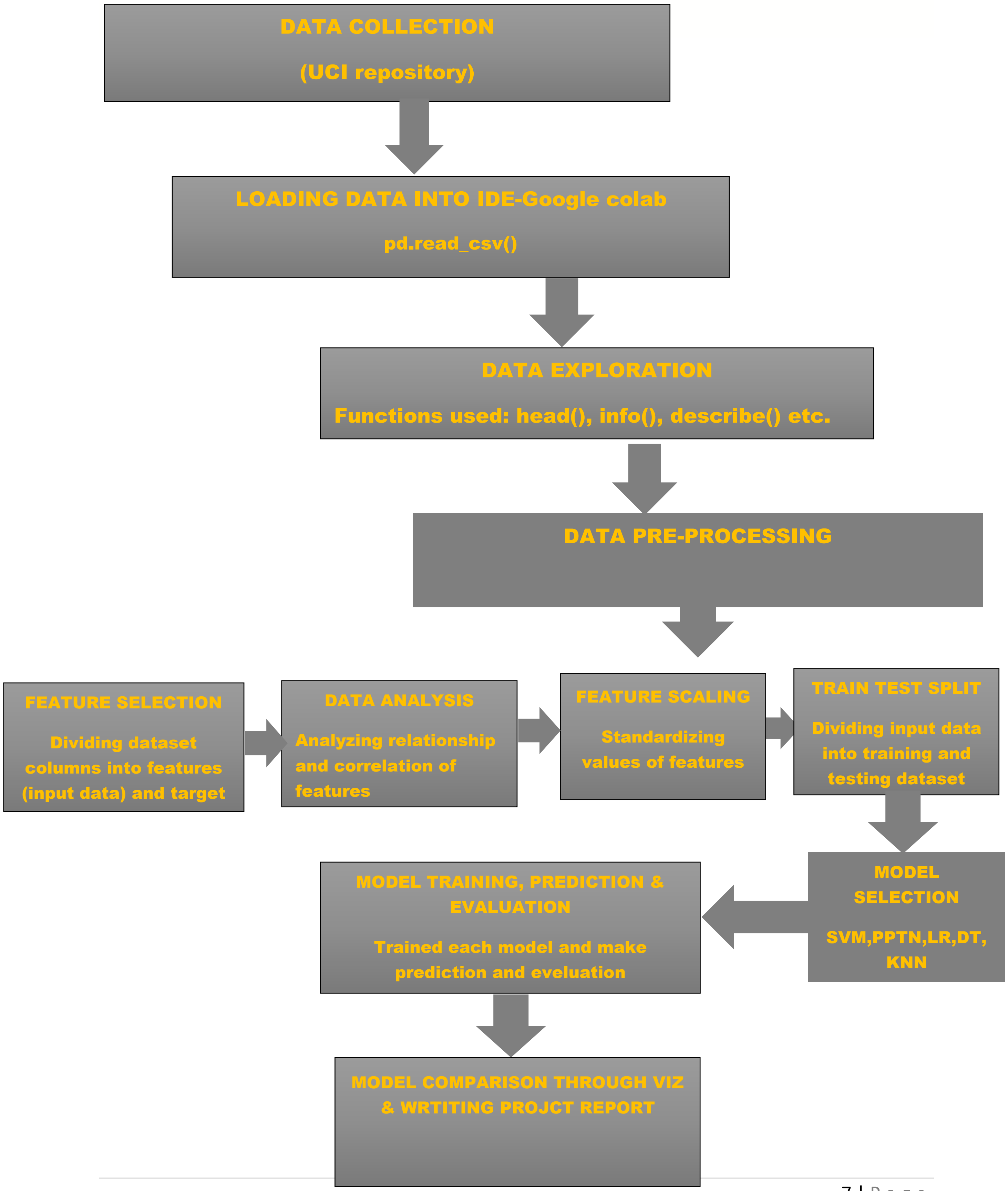
The fifth column in the dataset is "class" which contains class of a banknote denoted as either 0 or 1. The banknote dataset has 1372 instances, 762 for class 0 and 610 for class 1. Thus, its shape is 1372 by 5. For purposes of training and testing, the dataset was split in the ratio of 70 to 30. Thus, the training dataset denoted as data_train had 960 instances of banknotes whereas the testing dataset denoted as data_test had 412 instances of banknotes.

I classified the banknotes using 5 machine learning classification models namely Support Vector Machine, Perceptron, Logistic Regression, Decision Tree and K-Nearest Neighbors. **The machine learning model with 100% accuracy** during both training and testing **was Support Vector Machine model**. The accuracy for the rest of the models during testing and training was also good as it was above 96%, which means there were few misclassified banknotes. **Perceptron model** had 8 misclassified banknotes for class 0 and its testing and training accuracy was 98.06% and 99.48% respectively. **Logistic Regression model** had testing accuracy of 98.3% and training accuracy of 98.13% with 7 misclassified banknotes for class 0. **Decision Tree model** had testing accuracy of 97.82% and training accuracy of 100% with 6 misclassified banknotes for class 0 and 3 misclassified banknotes for class 1. Finally, **K-Nearest Neighbors** model had testing accuracy of 99.76% and training accuracy of 99.9% with 1 misclassified banknote for class 0.

Based on the evaluation report of the models, it was concluded that class 0 banknotes of this dataset are more difficult to classify than class 1. Thus, out of the 5 trained models, only **Decision Tree model** had misclassified banknotes (3) in class 1 and it performed poorly in comparison with the other models. The rest of the models classified class 1 bank notes correctly. **Support Vector Machine model** was recommended as the best model for classifying the banknotes in the dataset followed by **K-Nearest Neighbors model** and at last **Decision Tree model.**

CHAPTER2

2.1 DESCRIPTION OF THE PROJECT


DATA COLLECTION
(UCI repository)
LOADING DATA INTO IDE-Google colab
pd.read_csv()
DATA EXPLORATION
Functions used: head(), info(), describe() etc.
DATA PRE-PROCESSING
FEATURE SELECTION
Dividing dataset columns into features (input data) and target
DATA ANALYSIS
Analyzing relationship and correlation of features
FEATURE SCALING
Standardizing values of features
TRAIN TEST SPLIT
Dividing input data into training and testing dataset
MODEL SELECTION
SVM,PPTN,LR,DT, KNN
MODEL TRAINING, PREDICTION & EVALUATION
Trained each model and make prediction and eveluation
MODEL COMPARISON THROUGH VIZ & WRTITING PROJCT REPORT

CHAPTER3

3.0 PROJECT IMPLEMENTATION

3.1 DATA COLLECTION

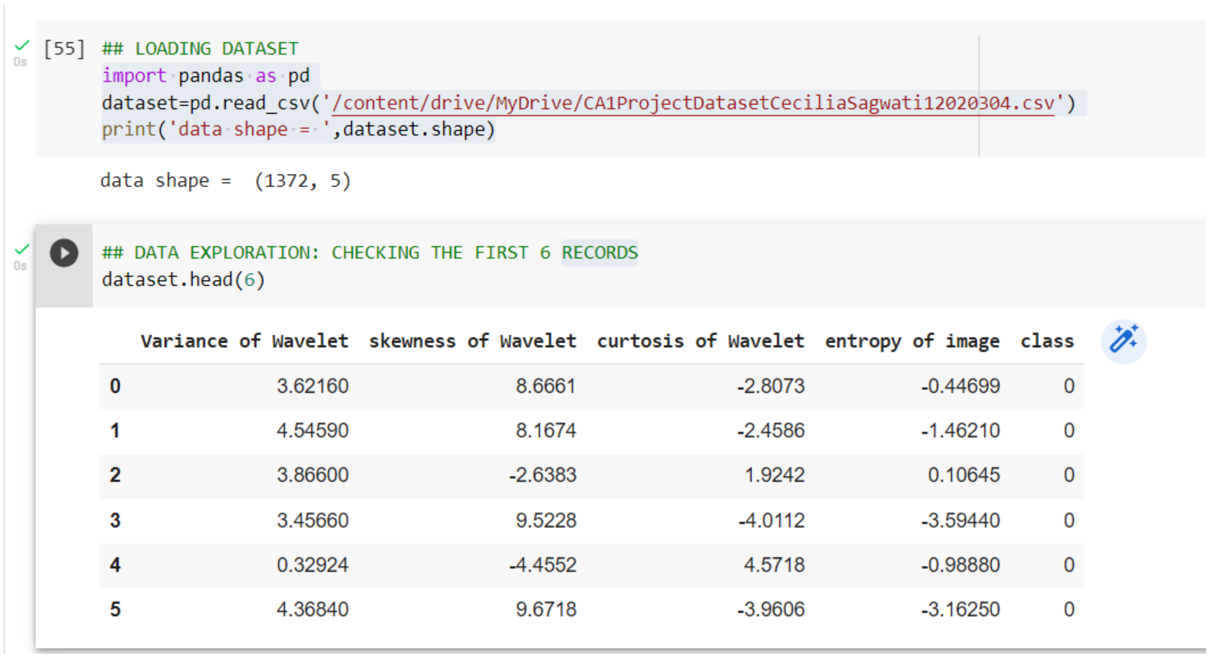
The dataset which I used to train the models was downloaded from UCI online repository through web link https://archive.ics.uci.edu/ml/machine-learning-databases/00267/.

This dataset contains 4 features that were extracted from images of real and fake banknotes. The features were digitalized by an industrial camera and, wavelet transform tool, was used to extract the features from the said banknotes' images. The features of the banknote dataset are:

- Variance of wavelet of transformed image denoted as **variance of wavelet**
- Skewness of wavelet of transformed image denoted as **skewness of wavelet**
- curtosis of wavelet of transformed image denoted as **curtosis of wavelet**
- entropy of transformed image denoted as **entropy of image**

3.2 LOADING DATA INTO IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

I carried out this machine learning project on google colab which is an online IDE for machine learning using python language. I loaded the dataset on google colab and printed the shape of the data set, and also the first 6 rows of the dataset with the below command lines:



```
[55] ## LOADING DATASET
     import pandas as pd
     dataset=pd.read_csv('/content/drive/MyDrive/CA1ProjectDatasetCeciliaSagwati12020304.csv')
     print('data shape = ',dataset.shape)

     data shape =  (1372, 5)
```

```
## DATA EXPLORATION: CHECKING THE FIRST 6 RECORDS
dataset.head(6)
```

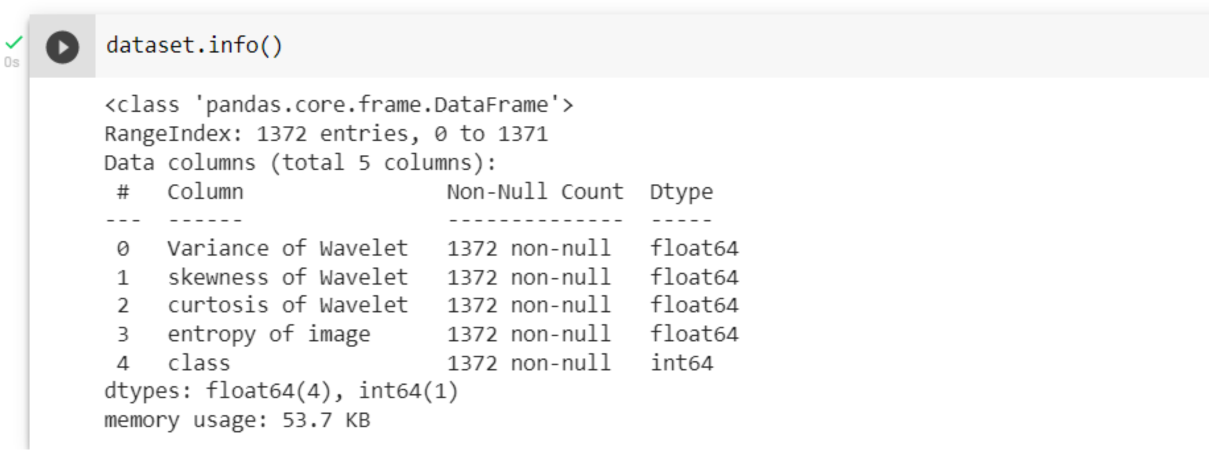
	Variance of Wavelet	skewness of Wavelet	curtosis of Wavelet	entropy of image	class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0
5	4.36840	9.6718	-3.9606	-3.16250	0

As shown in the above sceenshot, there are 1372 rows and 5 columns and the first 6 rows are also shown in this screenshot.

3.3 DATA EXPLORATION

In order to explore the dataset, I used different functions as shown and discussed below:

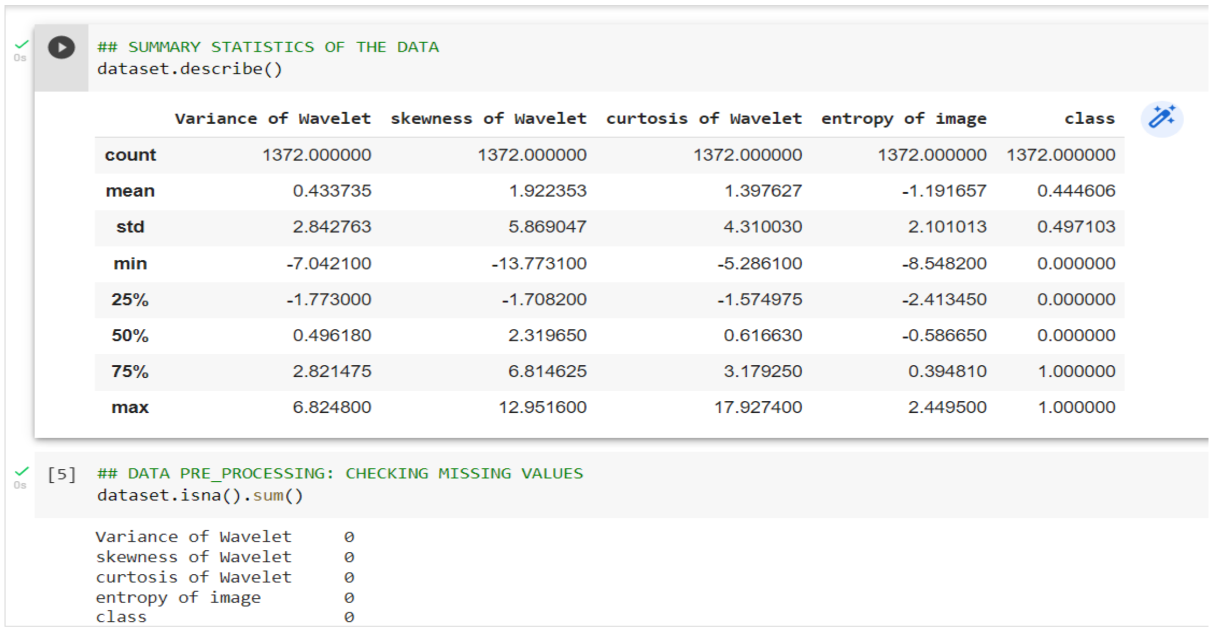
The **info()** function was used to show information about the dataset in terms of total number of columns and the column names, count of values which are not null and the data type. Hence, this data had 1372 instances of the banknotes and 5 columns, including the class/target column. The data types of the first 4 columns were decimal values and that of last column was integer.



```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372 entries, 0 to 1371
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Variance of Wavelet  1372 non-null   float64
 1   skewness of Wavelet  1372 non-null   float64
 2   curtosis of Wavelet  1372 non-null   float64
 3   entropy of image     1372 non-null   float64
 4   class                1372 non-null   int64  
dtypes: float64(4), int64(1)
memory usage: 53.7 KB
```

describe() function, provides count of the records and summary statistics for each column of the dataset such as mean, standard deviation, minimum value, the 25th (first) quartile, 50th (second) quartile and 75th (third) quartile as well as maximum value. These statistics are called 5 number summary. On the second cell, I used **isna()** together with **sum()** function. The **isna()** checks if there are **missing values** and the **sum() add** the missing values, as you can see in the screenshot, the dataset had no missing values.



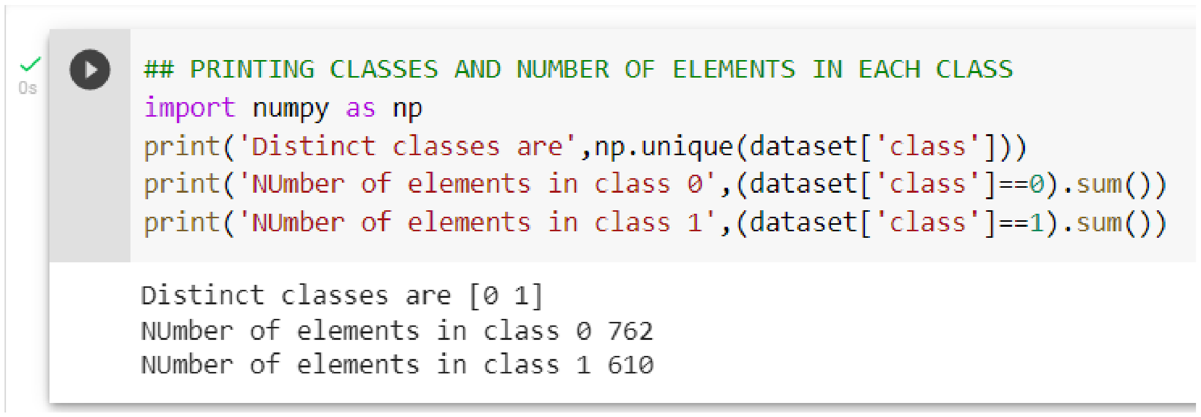
```
## SUMMARY STATISTICS OF THE DATA
dataset.describe()
```

	Variance of Wavelet	skewness of Wavelet	curtosis of Wavelet	entropy of image	class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	0.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103
min	-7.042100	-13.773100	-5.286100	-8.548200	0.000000
25%	-1.773000	-1.708200	-1.574975	-2.413450	0.000000
50%	0.496180	2.319650	0.616630	-0.586650	0.000000
75%	2.821475	6.814625	3.179250	0.394810	1.000000
max	6.824800	12.951600	17.927400	2.449500	1.000000

```
[5] ## DATA PRE_PROCESSING: CHECKING MISSING VALUES
dataset.isna().sum()
```

```
Variance of Wavelet    0
skewness of Wavelet    0
curtosis of Wavelet    0
entropy of image       0
class                  0
```

I used print() and unique() function to print distinct classes of the class column. As u can see, there were two classes, class 1 and class 0. I also printed number of elements in each class using print() and sum() function. As u can see, class 0 had 762 elements while class 1 had 610 elements.



```
## PRINTING CLASSES AND NUMBER OF ELEMENTS IN EACH CLASS
import numpy as np
print('Distinct classes are',np.unique(dataset['class']))
print('NUmber of elements in class 0',(dataset['class']==0).sum())
print('NUmber of elements in class 1',(dataset['class']==1).sum())
```

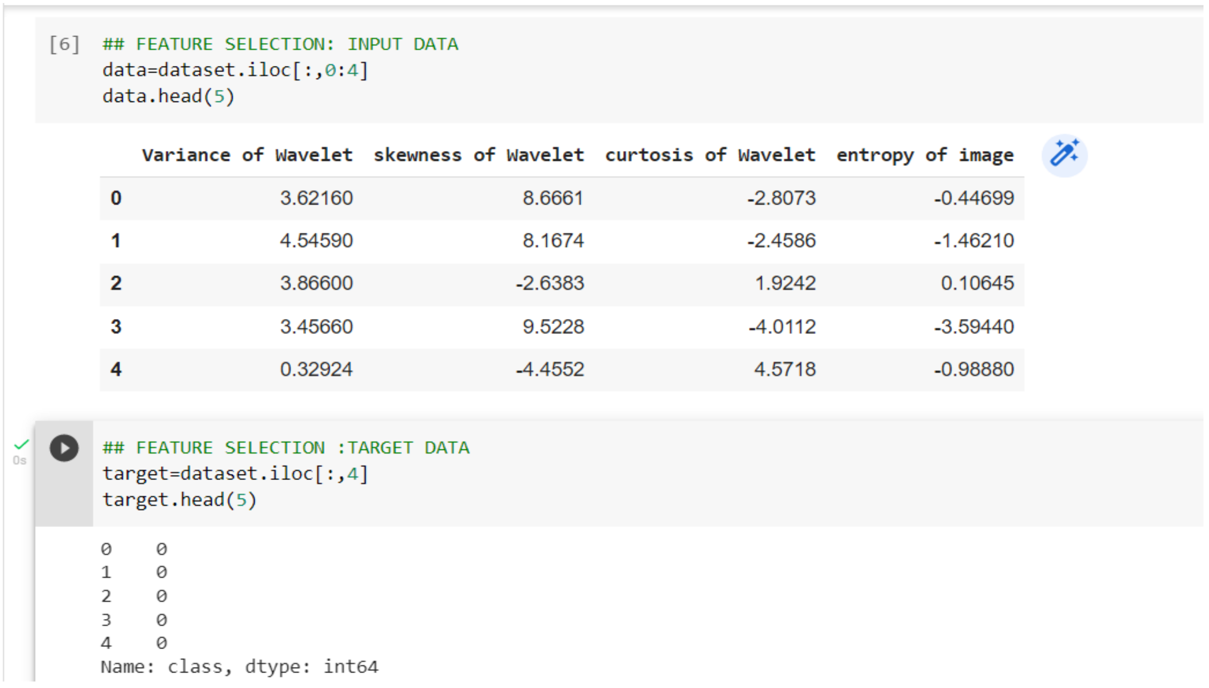
```
Distinct classes are [0 1]
NUmber of elements in class 0 762
NUmber of elements in class 1 610
```

4.0 DATA PRE-PROCESSING

I pre-processed the data and converted it into the right format for training and testing a machine learning algorithm. To achieve this, I used different functions as shown below:

4.1 FEATURE SELECTION

Machine learning algorithm requires data with a demarcation of independent variables which are also called independent features and dependent variable which is the label to be predicted and it is also called target or class label. Thus, class labels are predicted using independent features. Hence, I separated the data using index location method, and I stored the independent features in a variable called 'data' and the dependent feature /target in a variable called 'target'. I then printed the 5 records of each variable using head() function.



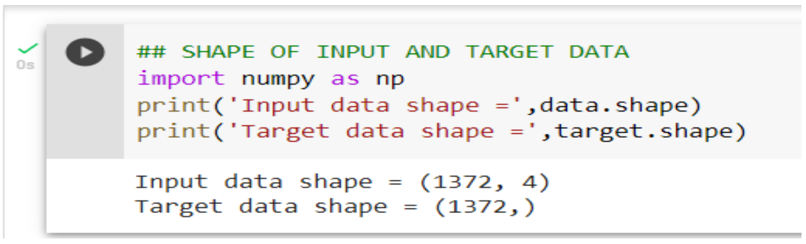
```
[6] ## FEATURE SELECTION: INPUT DATA
    data=dataset.iloc[:,0:4]
    data.head(5)
```

	Variance of Wavelet	skewness of Wavelet	curtosis of Wavelet	entropy of image
0	3.62160	8.6661	-2.8073	-0.44699
1	4.54590	8.1674	-2.4586	-1.46210
2	3.86600	-2.6383	1.9242	0.10645
3	3.45660	9.5228	-4.0112	-3.59440
4	0.32924	-4.4552	4.5718	-0.98880

```
## FEATURE SELECTION :TARGET DATA
target=dataset.iloc[:,4]
target.head(5)
```

```
0    0
1    0
2    0
3    0
4    0
Name: class, dtype: int64
```

I printed the shape of the input data and the target data as show below:



```
## SHAPE OF INPUT AND TARGET DATA
import numpy as np
print('Input data shape =',data.shape)
print('Target data shape =',target.shape)
```

```
Input data shape = (1372, 4)
Target data shape = (1372,)
```

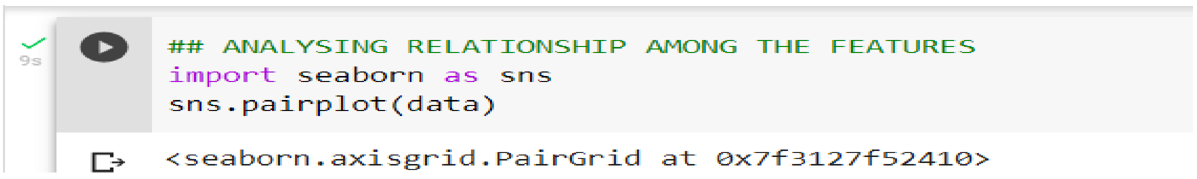
4.2 DATA ANALYSIS

pairplot() in seaborn(sns) library

I imported seaborn library in order to use pairplot() function which plot pairwise relationships of the columns in a dataset. As you can see in fig1 below, the pairplot function produced a grid in which the features were ploted against each other such that each feature had a shared y-axis across a single row and a shared x-axis across a single column.

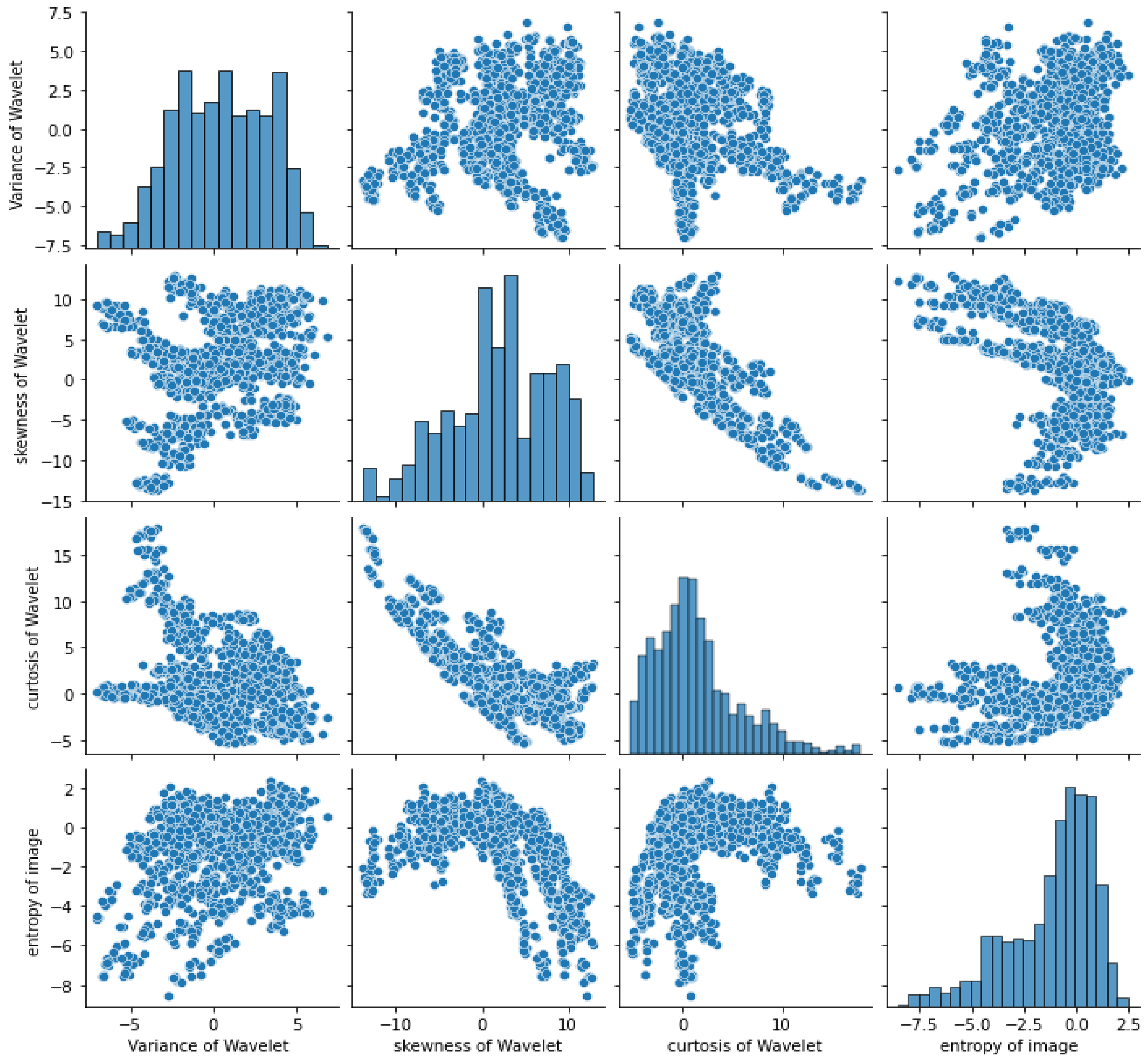
After analysing the above plot, it was discovered that the skewness of wavelet and curtosis of wavelet has an inverse linear relationship (negative correlation), meaning when value of one variable decreases, the value of the other variable increases and vice versa. Whereas entropy of image and variance of wavelet have a non-inverse linear relationship (positive correlation), meaning when value of one variable increases, the value of the other variable also increases and vice versa.

Fig1: Data Analysis using seaborn plot



```
## ANALYSING RELATIONSHIP AMONG THE FEATURES
import seaborn as sns
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7f3127f52410>
```

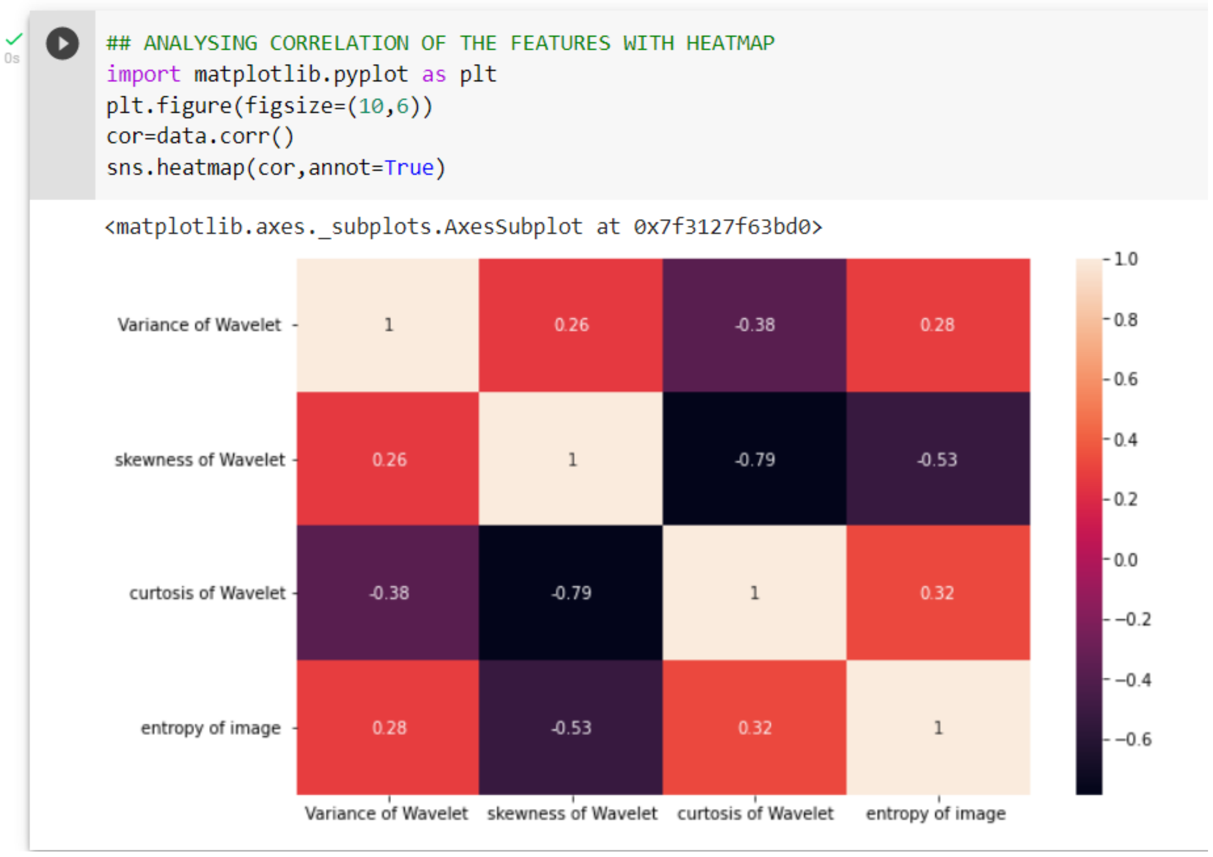


figure(), corr(), heatmap () functions

For correlation analysis, the value ‘+1’ means highly positive correlation while ‘-1’ means highly negative correlation and ‘0’ means no correlation. I also used **corr()** function to find correlation between the features of the input data, **figure()** function to change the size of the plot and **heatmap()** function to produce the plot shown using the output of the corr() function.

Since I already imported the seaborn library in the previous plot, I just Imported matplotlib library. I executed the codes listed below and the output was as shown below:

Fig2: Heatmap



```
## ANALYSING CORRELATION OF THE FEATURES WITH HEATMAP
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
cor=data.corr()
sns.heatmap(cor,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3127f63bd0>
```

Fig2 shows that skewness of wavelet and curtosis of wavelet have a high negative correlation (-0.79) and skewness of wavelet and entropy of image also have a negative correlation of -0.53. Hence, it can be concluded that variance of wavelet is an important feature in determining class of the banknote because it has low correlation with the other features.

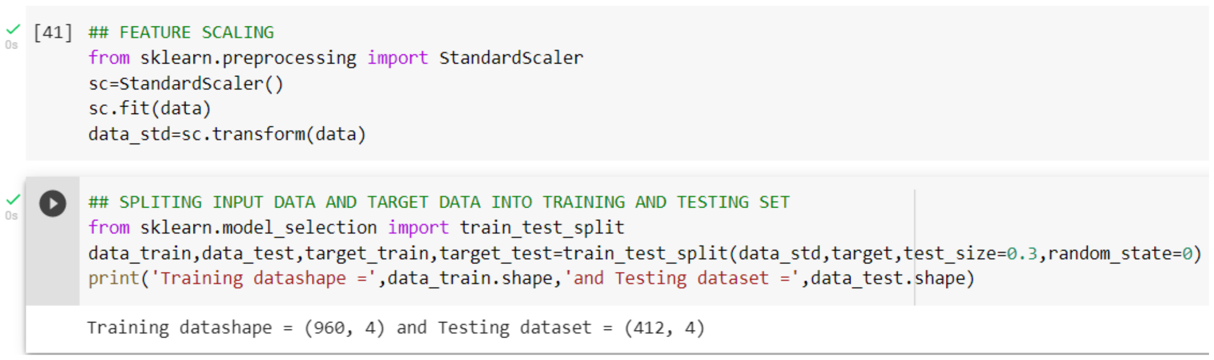
4.3 FEATURE SCAILING

Feature scaling helps to standardize the data so that there is low variation between the values of the features. I standardized the input data with StandardScaler() function, fit() and transform(). The fit() function estimate the mean and standard deviation of the data, while the transform() function replaces the data values with the values estimated by the fit().

4.4 DIVIDING DATA INTO TRAINING AND TESTING DATASET

I divided input data and the target data into training and testing datasets using train-test-split() function from sklean. I reserved 30% of the input and target data for testing while the remaining 70% was used for training. The training dataset is used to train the machine learning model while the testing dataset is used to evaluate the performance of the trained model. Reserving some data for testing is very important as it enables evaluation of performance of the model on unseen data. I also printed the shape of training and testing data.

FEATURE SCALING AND TRAIN-TEST-SPLIT COMMANDS



```
[41] ## FEATURE SCALING
     from sklearn.preprocessing import StandardScaler
     sc=StandardScaler()
     sc.fit(data)
     data_std=sc.transform(data)
```

```
## SPLITING INPUT DATA AND TARGET DATA INTO TRAINING AND TESTING SET
from sklearn.model_selection import train_test_split
data_train,data_test,target_train,target_test=train_test_split(data_std,target,test_size=0.3,random_state=0)
print('Training datashape =',data_train.shape,'and Testing dataset =',data_test.shape)

Training datashape = (960, 4) and Testing dataset = (412, 4)
```

5.1 MODEL SELECTION

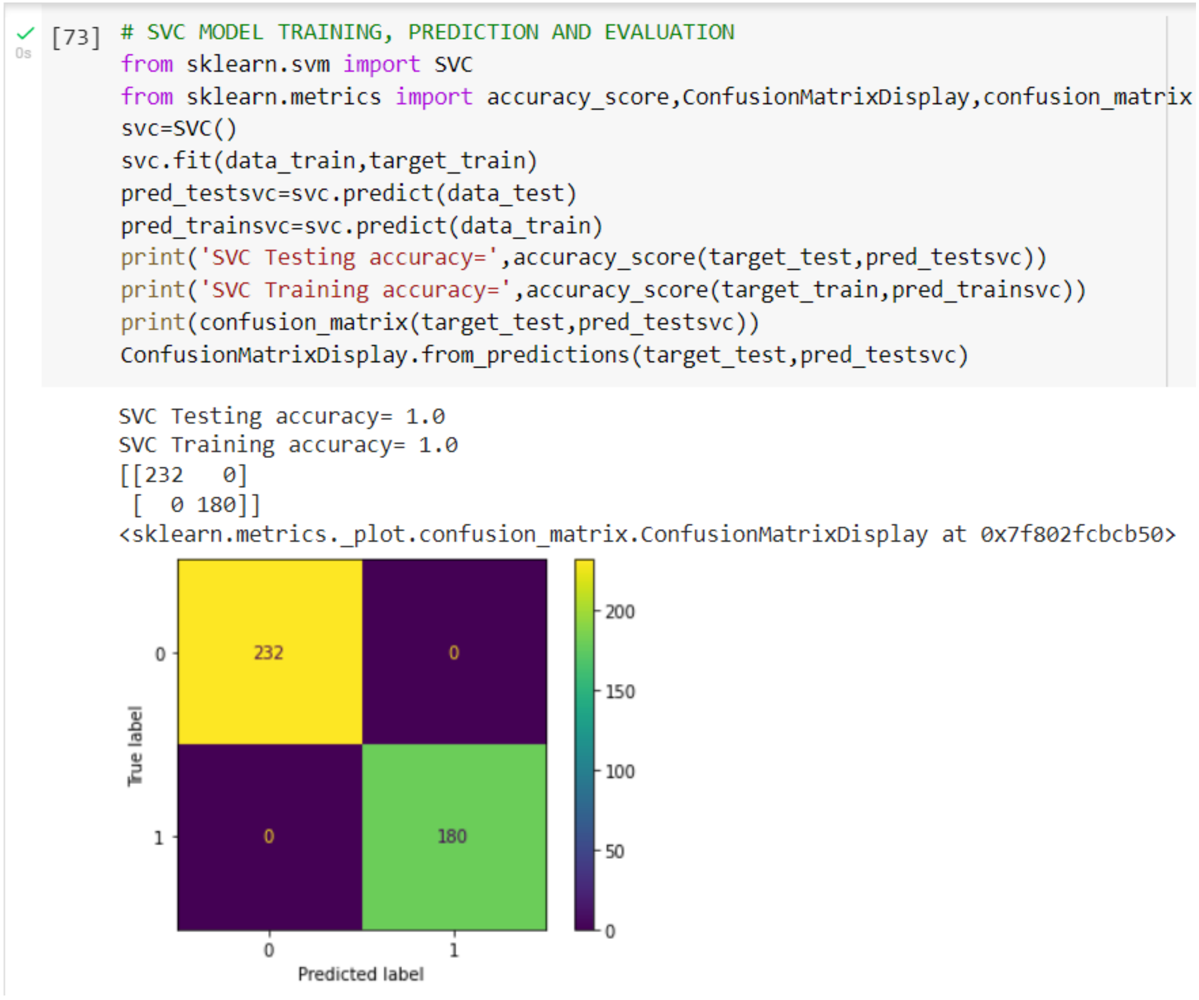
I selected 5 machine learning classifiers namely Support Vector Machine (SVM), Perceptron, Logistic Regression, Decision Tree and K-Nearest Neighbors (KNN). I wanted to train these models and compare their performance, in order to choose the best model among the 5 models.

6.1 MODEL TRAINING, PREDICTION AND EVALUATION

I trained each model separately using respective model function and fit() function. After training the model I evaluated performance of the model on the testing dataset by predicting class labels of the testing dataset using predict(). I also measured performance of the model during both testing and training using accuracy_score() function. I also used confusion_matrix() and ConfusionMatrixDisplay() metric functions to show and display the output of the predicted labels against the true labels. The commands for training, prediction and evaluation of each model is as shown in the section below:

SUPPORT VECTOR MACHINE (SVM) MODEL TRAINING, PREDICTION AND EVALUATION

Fig3: Screenshot of SVM model training, prediction and evaluation



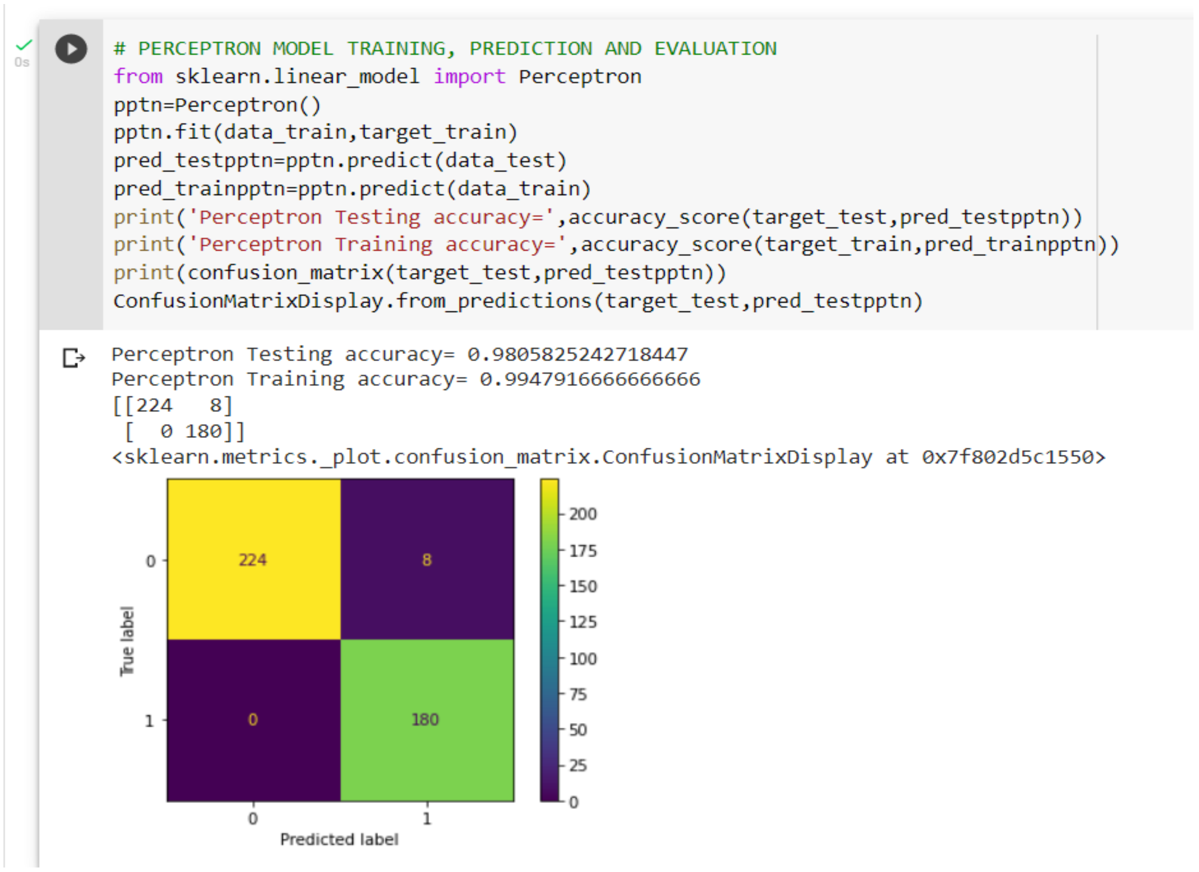
```
[73] # SVC MODEL TRAINING, PREDICTION AND EVALUATION
     from sklearn.svm import SVC
     from sklearn.metrics import accuracy_score,ConfusionMatrixDisplay,confusion_matrix
     svc=SVC()
     svc.fit(data_train,target_train)
     pred_testsvc=svc.predict(data_test)
     pred_trainsvc=svc.predict(data_train)
     print('SVC Testing accuracy=',accuracy_score(target_test,pred_testsvc))
     print('SVC Training accuracy=',accuracy_score(target_train,pred_trainsvc))
     print(confusion_matrix(target_test,pred_testsvc))
     ConfusionMatrixDisplay.from_predictions(target_test,pred_testsvc)
```

```
SVC Testing accuracy= 1.0
SVC Training accuracy= 1.0
[[232   0]
 [  0 180]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f802fcbcb50>
```

As shown in fig3 accuracy of Support Vector Machine model was 100% during both testing and training. Hence, there were no misclassified banknotes.

PERCEPTRON MODEL TRAINING, PREDICTION AND EVALUATION

Fig4: Screenshot of perceptron model training, prediction and evaluation



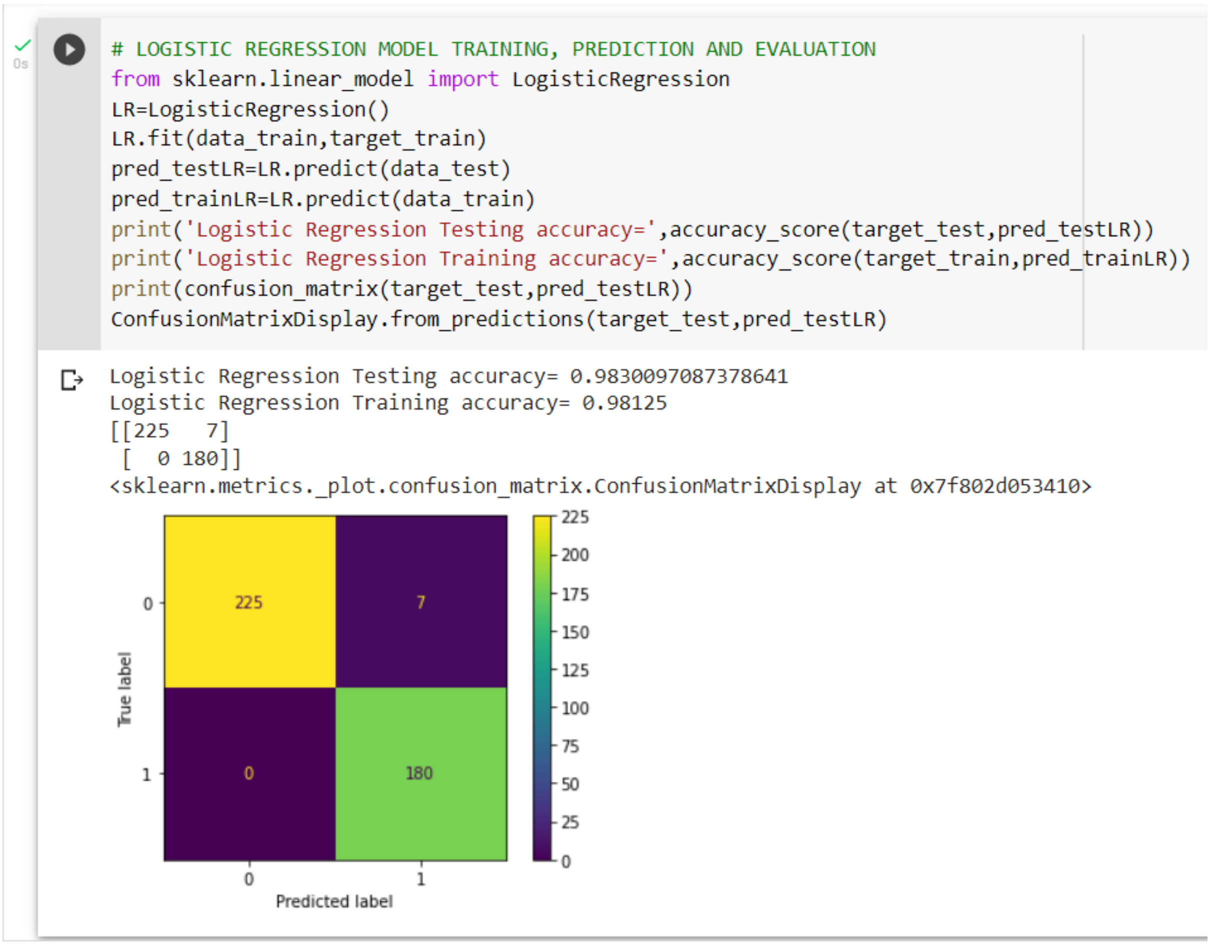
```
# PERCEPTRON MODEL TRAINING, PREDICTION AND EVALUATION
from sklearn.linear_model import Perceptron
pptn=Perceptron()
pptn.fit(data_train,target_train)
pred_testpptn=pptn.predict(data_test)
pred_trainpptn=pptn.predict(data_train)
print('Perceptron Testing accuracy=',accuracy_score(target_test,pred_testpptn))
print('Perceptron Training accuracy=',accuracy_score(target_train,pred_trainpptn))
print(confusion_matrix(target_test,pred_testpptn))
ConfusionMatrixDisplay.from_predictions(target_test,pred_testpptn)
```

```
Perceptron Testing accuracy= 0.9805825242718447
Perceptron Training accuracy= 0.9947916666666666
[[224   8]
 [  0 180]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f802d5c1550>
```

As shown in figure 4, testing accuracy of perceptron model was 98.06% and training accuracy was 99.48%. There were 8 misclassified labels of class 0 during testing. Hence, class 0 labels were short of 8 banknotes which were classified as class 1.

LOGISTIC REGRESSION MODEL TRAINING, PREDICTION AND EVALUATION

Fig5: Screenshot of Logistic Regression model training, prediction and evaluation



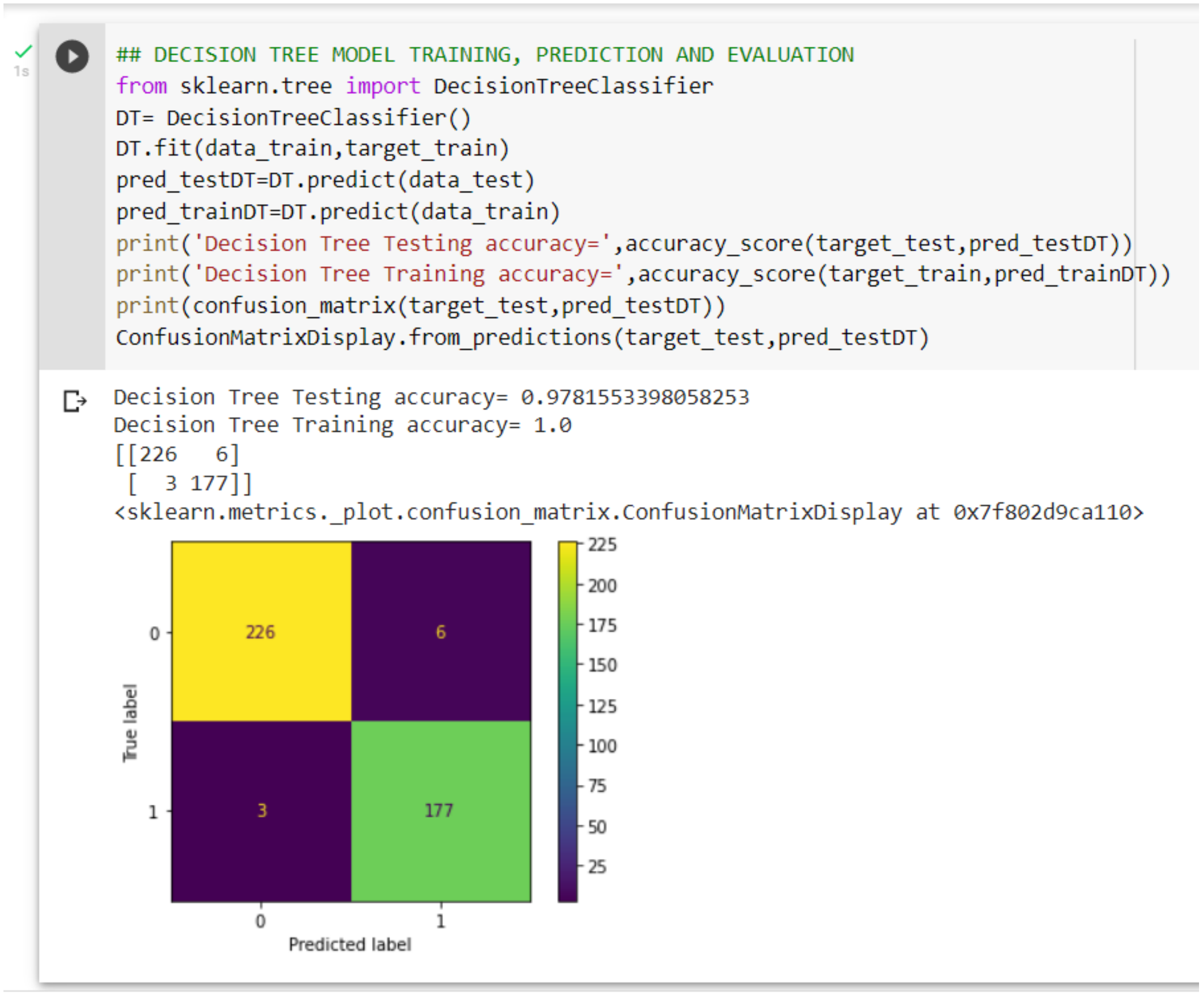
```
# LOGISTIC REGRESSION MODEL TRAINING, PREDICTION AND EVALUATION
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression()
LR.fit(data_train,target_train)
pred_testLR=LR.predict(data_test)
pred_trainLR=LR.predict(data_train)
print('Logistic Regression Testing accuracy=',accuracy_score(target_test,pred_testLR))
print('Logistic Regression Training accuracy=',accuracy_score(target_train,pred_trainLR))
print(confusion_matrix(target_test,pred_testLR))
ConfusionMatrixDisplay.from_predictions(target_test,pred_testLR)
```

```
Logistic Regression Testing accuracy= 0.9830097087378641
Logistic Regression Training accuracy= 0.98125
[[225   7]
 [  0 180]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f802d053410>
```

As shown in figure 5, accuracy of Logistic Regression during testing was 98.30% and 98.13% during training. There were 7 misclassified labels of class 0. Thus, 7 labels of class 0 were classified as class 1 resulting in a short fall of 7 labels in class 0.

DECISION TREE MODEL TRAINING, PREDICTION AND EVALUATION

Fig6: Screenshot of Decision Tree model training, prediction and evaluation



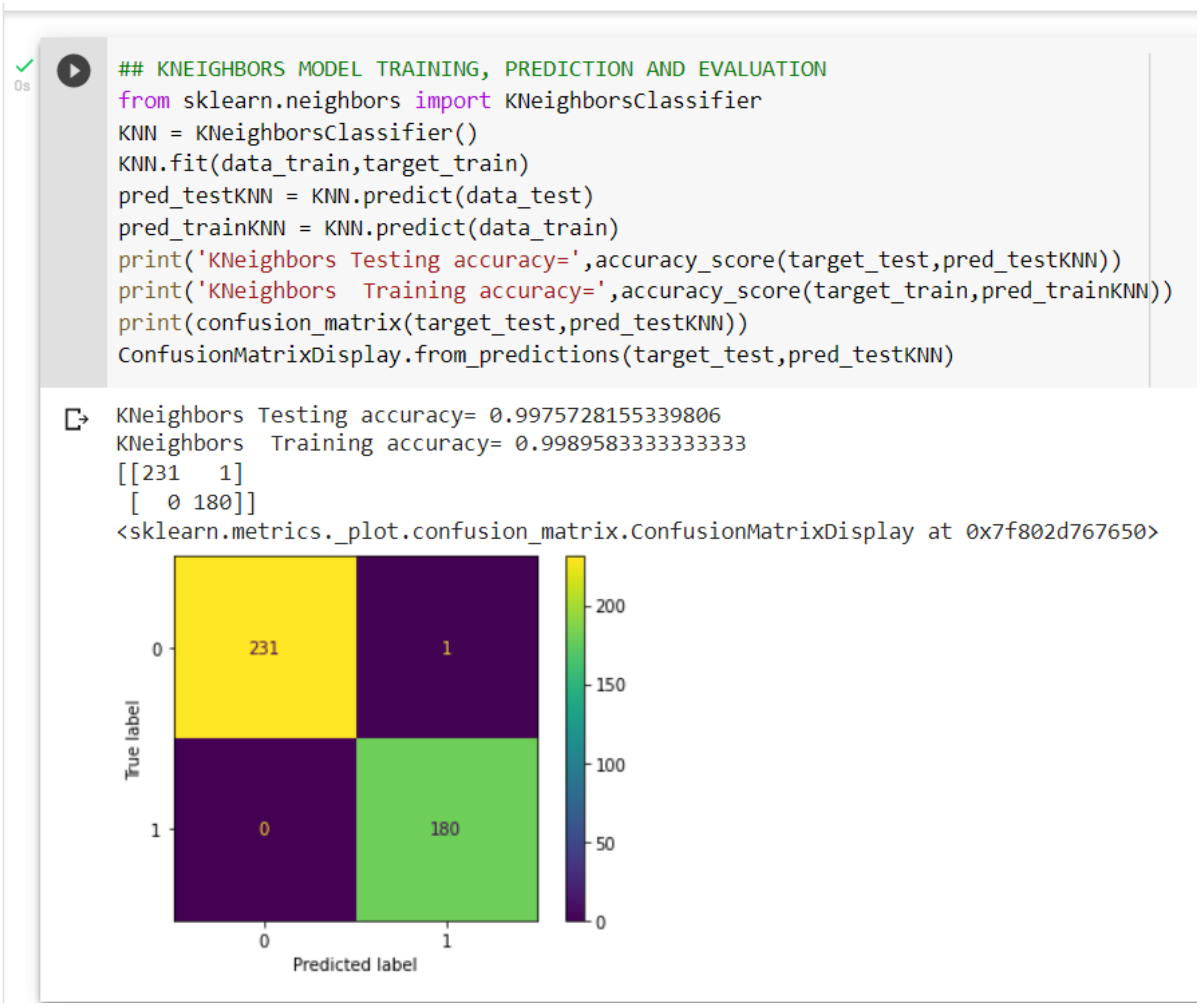
```
## DECISION TREE MODEL TRAINING, PREDICTION AND EVALUATION
from sklearn.tree import DecisionTreeClassifier
DT= DecisionTreeClassifier()
DT.fit(data_train,target_train)
pred_testDT=DT.predict(data_test)
pred_trainDT=DT.predict(data_train)
print('Decision Tree Testing accuracy=',accuracy_score(target_test,pred_testDT))
print('Decision Tree Training accuracy=',accuracy_score(target_train,pred_trainDT))
print(confusion_matrix(target_test,pred_testDT))
ConfusionMatrixDisplay.from_predictions(target_test,pred_testDT)
```

```
Decision Tree Testing accuracy= 0.9781553398058253
Decision Tree Training accuracy= 1.0
[[226   6]
 [  3 177]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f802d9ca110>
```

As shown in fig6 above, performance of Decision Tree model during testing was 97.82% while its performance during training was 100%. There were 6 misclassified labels of class 0 and 1 misclassified label of class 1. This is the only model with a misclassified label in class 1.

K-NEIGHBORS MODEL TRAINING, PREDICTION AND EVALUATION

Fig7: Screenshot of K-Nearest neighbors model training, prediction and evaluation



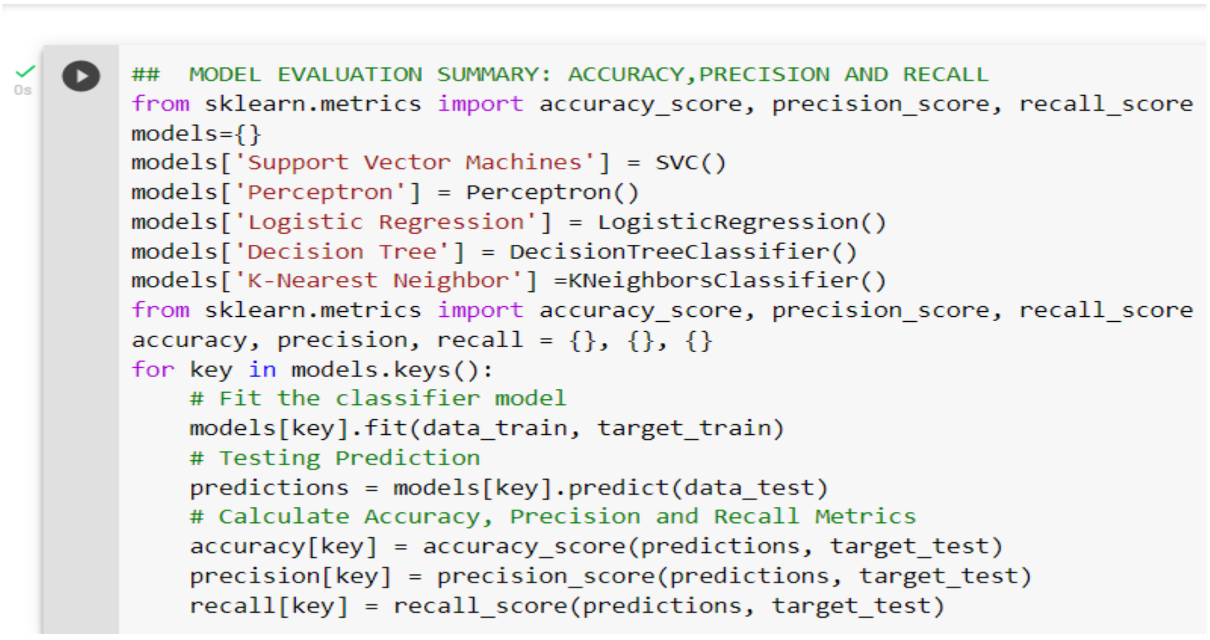
```
## KNEIGHBORS MODEL TRAINING, PREDICTION AND EVALUATION
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier()
KNN.fit(data_train,target_train)
pred_testKNN = KNN.predict(data_test)
pred_trainKNN = KNN.predict(data_train)
print('KNeighbors Testing accuracy=',accuracy_score(target_test,pred_testKNN))
print('KNeighbors  Training accuracy=',accuracy_score(target_train,pred_trainKNN))
print(confusion_matrix(target_test,pred_testKNN))
ConfusionMatrixDisplay.from_predictions(target_test,pred_testKNN)
```

```
KNeighbors Testing accuracy= 0.9975728155339806
KNeighbors  Training accuracy= 0.9989583333333333
[[231   1]
 [  0 180]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f802d767650>
```

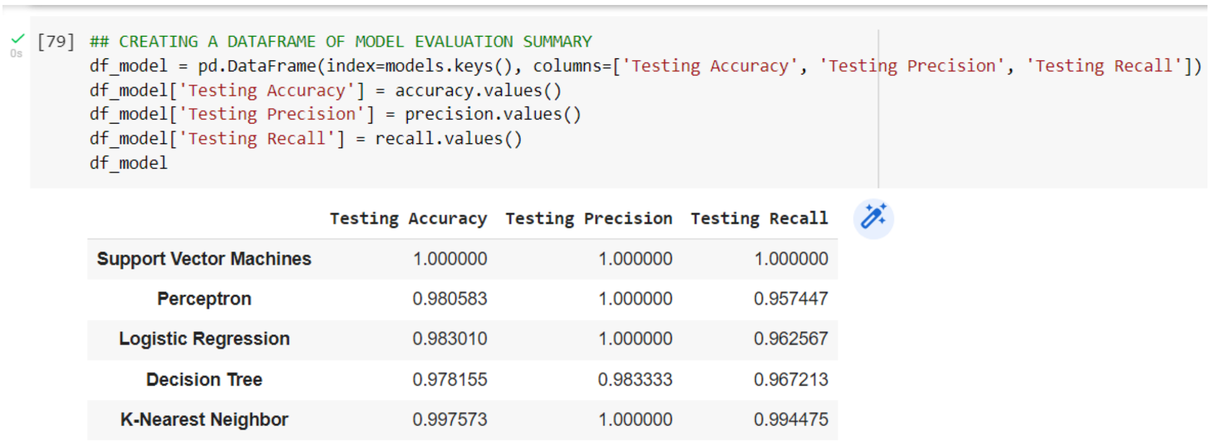
As shown in fig7, performance of K-Nearest Neighbors model was 99.76% during testing and 99.9% during training. There was only 1 misclassified label in class 0. Performance of this model is very close to performance of Support Vector Machine Model.

7.1 MODEL EVALUATION SUMMARY DURING TESTING

This is the summary of evaluation of all the 5 models during testing. Support Vector Machine model classified the class labels correctly and had 100% accuracy, precision and recall. The second-best model in terms of performance, is K-Nearest Neighbor with an accuracy of 99.76%, precision of 100% and a recall of 99.45%. Generally, all the models performed very well as their performance was above 96% in respect of all the metrics used to evaluate them. Decision tree model was the one with lowest performance.



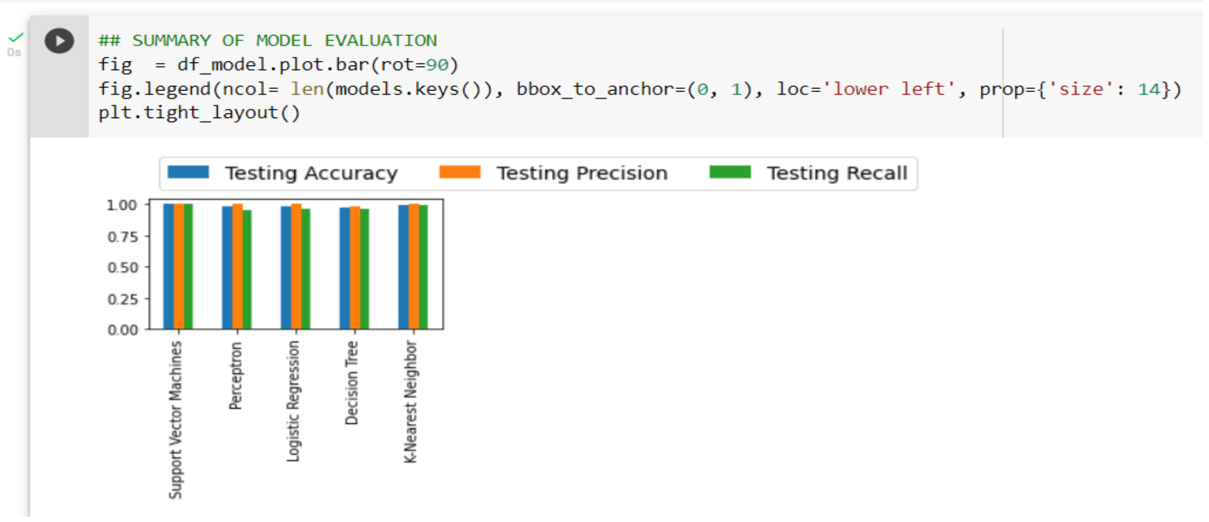
```
##  MODEL EVALUATION SUMMARY: ACCURACY,PRECISION AND RECALL
from sklearn.metrics import accuracy_score, precision_score, recall_score
models={}
models['Support Vector Machines'] = SVC()
models['Perceptron'] = Perceptron()
models['Logistic Regression'] = LogisticRegression()
models['Decision Tree'] = DecisionTreeClassifier()
models['K-Nearest Neighbor'] =KNeighborsClassifier()
from sklearn.metrics import accuracy_score, precision_score, recall_score
accuracy, precision, recall = {}, {}, {}
for key in models.keys():
    # Fit the classifier model
    models[key].fit(data_train, target_train)
    # Testing Prediction
    predictions = models[key].predict(data_test)
    # Calculate Accuracy, Precision and Recall Metrics
    accuracy[key] = accuracy_score(predictions, target_test)
    precision[key] = precision_score(predictions, target_test)
    recall[key] = recall_score(predictions, target_test)
```



```
[79] ## CREATING A DATAFRAME OF MODEL EVALUATION SUMMARY
     df_model = pd.DataFrame(index=models.keys(), columns=['Testing Accuracy', 'Testing Precision', 'Testing Recall'])
     df_model['Testing Accuracy'] = accuracy.values()
     df_model['Testing Precision'] = precision.values()
     df_model['Testing Recall'] = recall.values()
     df_model
```

	Testing Accuracy	Testing Precision	Testing Recall
Support Vector Machines	1.000000	1.000000	1.000000
Perceptron	0.980583	1.000000	0.957447
Logistic Regression	0.983010	1.000000	0.962567
Decision Tree	0.978155	0.983333	0.967213
K-Nearest Neighbor	0.997573	1.000000	0.994475

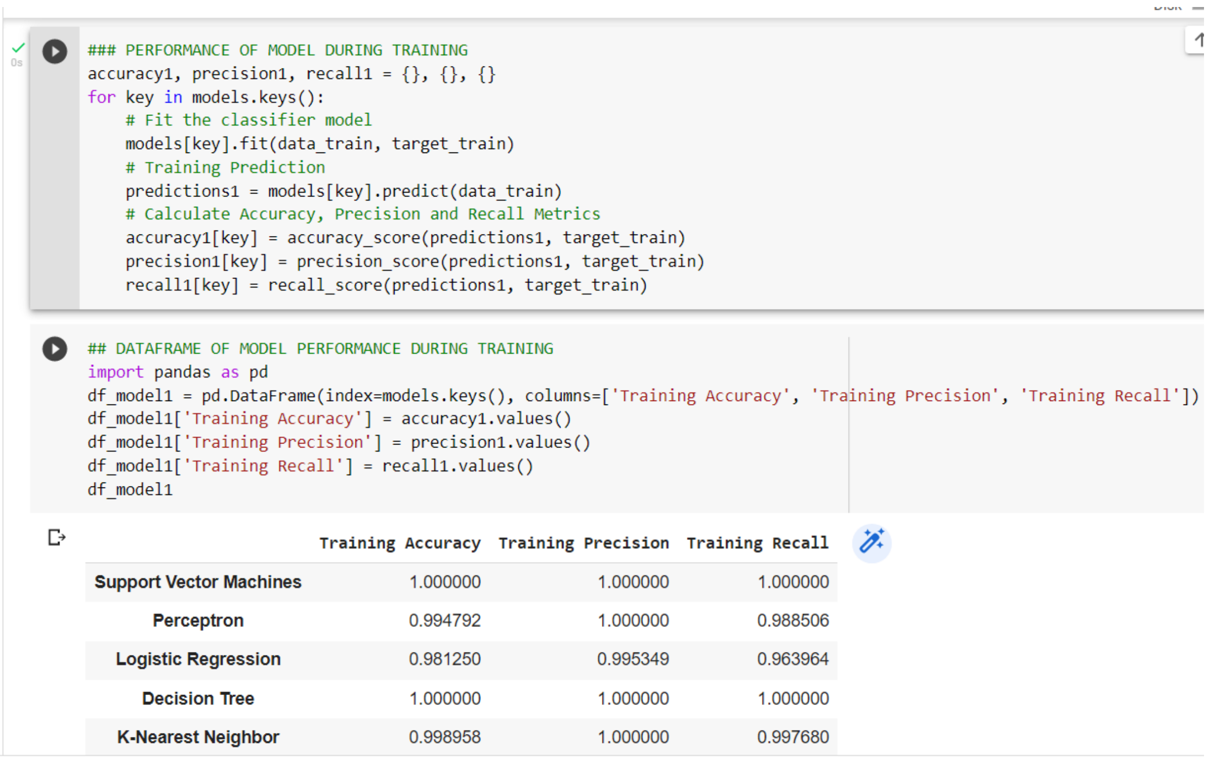
Fig8: Graphical representation of model evaluation summary during testing



```
## SUMMARY OF MODEL EVALUATION
fig  = df_model.plot.bar(rot=90)
fig.legend(ncol= len(models.keys()), bbox_to_anchor=(0, 1), loc='lower left', prop={'size': 14})
plt.tight_layout()
```

8.1 MODEL EVALUATION SUMMARY DURING TRAINING

This is the summary of evaluation of all the 5 models during training. Support Vector Machine model classified the class labels correctly and had 100% accuracy, precision and recall. The second-best model in terms of performance, is K-Nearest Neighbor with an accuracy of 99.9%, precision of 100% and a recall of 99.77%. Generally, all the models performed very well as their performance was above 97% in respect of all the metrics used to evaluate them. Logistic Regression model was the one with lowest performance during training.

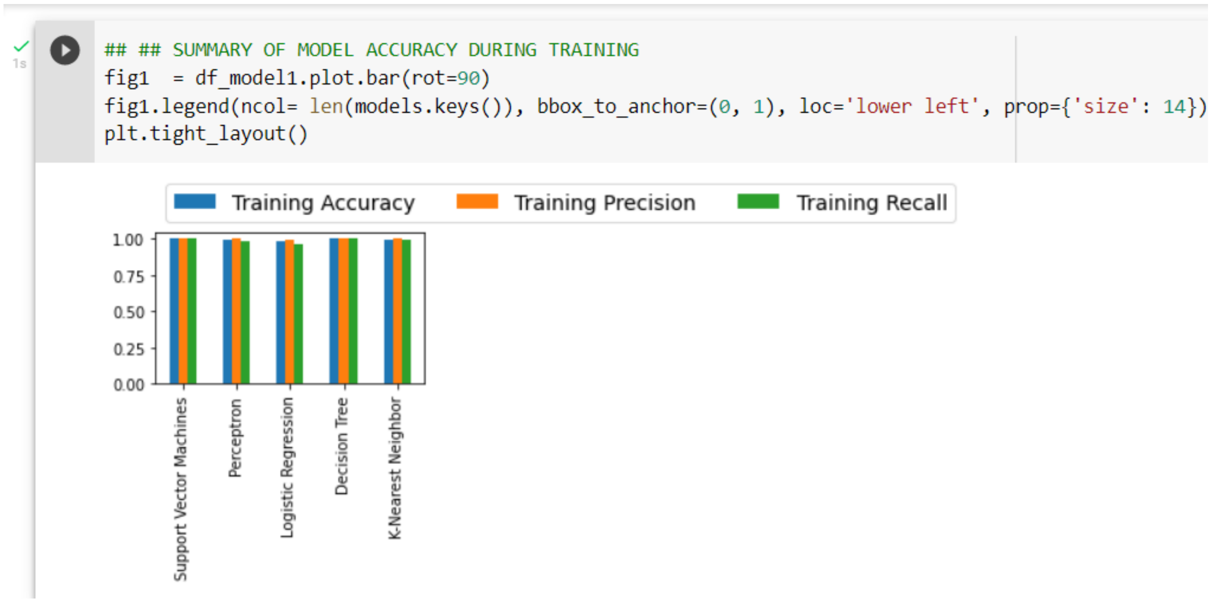


```
### PERFORMANCE OF MODEL DURING TRAINING
accuracy1, precision1, recall1 = {}, {}, {}
for key in models.keys():
    # Fit the classifier model
    models[key].fit(data_train, target_train)
    # Training Prediction
    predictions1 = models[key].predict(data_train)
    # Calculate Accuracy, Precision and Recall Metrics
    accuracy1[key] = accuracy_score(predictions1, target_train)
    precision1[key] = precision_score(predictions1, target_train)
    recall1[key] = recall_score(predictions1, target_train)
```

```
## DATAFRAME OF MODEL PERFORMANCE DURING TRAINING
import pandas as pd
df_model1 = pd.DataFrame(index=models.keys(), columns=['Training Accuracy', 'Training Precision', 'Training Recall'])
df_model1['Training Accuracy'] = accuracy1.values()
df_model1['Training Precision'] = precision1.values()
df_model1['Training Recall'] = recall1.values()
df_model1
```

	Training Accuracy	Training Precision	Training Recall
Support Vector Machines	1.000000	1.000000	1.000000
Perceptron	0.994792	1.000000	0.988506
Logistic Regression	0.981250	0.995349	0.963964
Decision Tree	1.000000	1.000000	1.000000
K-Nearest Neighbor	0.998958	1.000000	0.997680

Fig9: Graphical representation of model evaluation summary during training



```
## ## SUMMARY OF MODEL ACCURACY DURING TRAINING
fig1  = df_model1.plot.bar(rot=90)
fig1.legend(ncol= len(models.keys()), bbox_to_anchor=(0, 1), loc='lower left', prop={'size': 14})
plt.tight_layout()
```

CHAPTER4

CONCLUSION

All the models generally performed very well during both testing and training. Support Vector machine (SVM) model was on position 1 in terms of performance, while Decision tree model was at last. Hence, I recommend SVM as the best classifier for classifying the banknotes in the dataset seconded by KNN. There were many misclassified labels in class 0, therefore, it can be concluded that class 0 labels are more difficult to classify than class 1 labels. The google colab link for this project is shown below:

Google colab link

https://colab.research.google.com/drive/1rNCwskwB0XHXcCnpacgMOJOP1-YBcvYT?usp=sharing

Github link

https://github.com/Cec1989/CeciliaSagwati12020304/branches