

Data and Artificial Intelligence

Cyber Shujaa Program

Week 1 Assignment

Web Scraping and Data Handling in Python

Student Name: Deborah Kwamboka Omae

Student ID: CS-DA02-25075

Table of Contents

Data and Artificial Intelligence	1
Cyber Shujaa Program.....	1
Week 1 Assignment Web Scraping and Data Handling in Python.....	1
Introduction	4
Tasks Completed	4
Step 1: Load the dataset	4
Step 2: Explore the dataset.....	5
Step 3: Prepare the data	7
Step 4: Build the models and evaluate them.....	8
Logistic regression	8
Decision tree model.....	9
Random forest.....	10
K nearest neighbours	11
Naïve bayes	12
Support vector machine.....	13
Step 5: model comparison	14
Link to code.....	15
Conclusion.....	15

Table of figures

Figure 1: screenshot showing code for loading the wine dataset	5
Figure 2: screenshot displaying x	6
Figure 3: screenshot displaying y	7
Figure 4: screenshot showing the dataset has no missing values.....	7
Figure 5: screenshot to the above code.....	8
Figure 6: screenshot showing the confusion matrix.....	9
Figure 7: screenshot for decision tree model	10
Figure 8: screenshot for random forest model	11
Figure 9: screenshot for k- nearest neighbors	12
Figure 10: screenshot for naive bayes.....	13
Figure 11: screenshot for support vector machine.....	14
Figure 12: screenshot showing model comparison	15

Introduction

This week's assignment was to apply my understanding of **supervised machine learning classification models** by building and evaluating various models. I used the Wine dataset from scikit-learn. The goal was to explore and visualize the data, and train six different models: Logistic Regression, Decision Tree, Random Forest, k-Nearest Neighbors (KNN), Naive Bayes and, Support Vector Machine (SVM). The goal was to compare their performance using standard evaluation metrics and confusion matrices, and to interpret which model works best and why.

This assignment helped me understand not just how to train models, but how to assess and compare their performance on the same dataset under similar conditions.

Tasks Completed

Step 1: Load the dataset

In this step I imported the necessary libraries and loaded the dataset from Scikitlearn. I loaded the dataset as a pandas data frame and the target column which is a dependant variable as a pandas series.

Code:

```
# Import required libraries

import pandas as pd

import numpy as np

import matplotlib as mpl

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load_wine

from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

```

from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

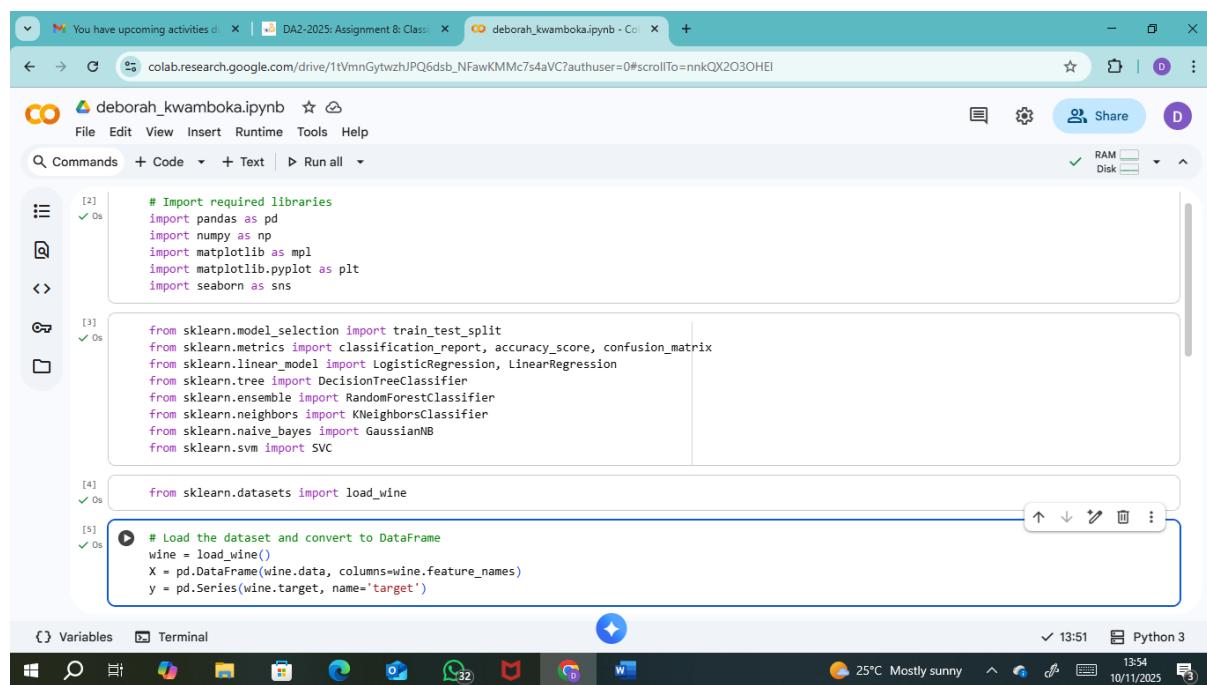
```

Load the dataset and convert to DataFrame

```

wine = load_wine()
X = pd.DataFrame(wine.data, columns=wine.feature_names)
y = pd.Series(wine.target, name='target')  
screenshot:

```



The screenshot shows a Google Colab notebook titled "deborah_kwamboka.ipynb". The code cell [2] contains imports for pandas, numpy, matplotlib, and seaborn. The code cell [3] contains imports for train_test_split, classification_report, accuracy_score, confusion_matrix, LogisticRegression, LinearRegression, DecisionTreeClassifier, RandomForestClassifier, KNeighborsClassifier, GaussianNB, and SVC from various sklearn modules. The code cell [4] imports load_wine from sklearn.datasets. The code cell [5] loads the wine dataset, converts it to a DataFrame, and stores the target variable y. The status bar at the bottom indicates Python 3, 13:51, and 25°C Mostly sunny.

Figure 1: screenshot showing code for loading the wine dataset

Step 2: Explore the dataset

In this step I got an overview of my data set by printing x and y. I checked for the columns in x. I also checked for missing values and found that there were no missing values in the dataset.

Code:

```
print(X)

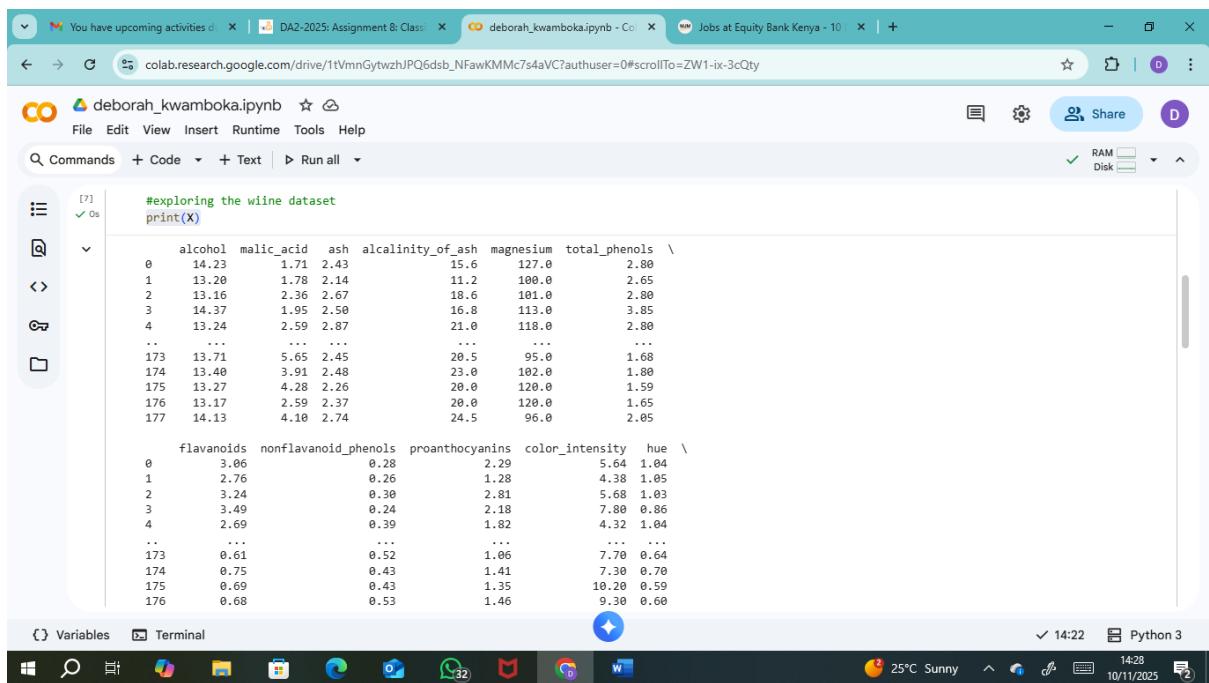
print("Columns in X:", X.columns)

print(y)

print("Missing values in X:\n", X.isnull().sum())

print("\nMissing values in y:\n", y.isnull().sum())
```

Screenshot:



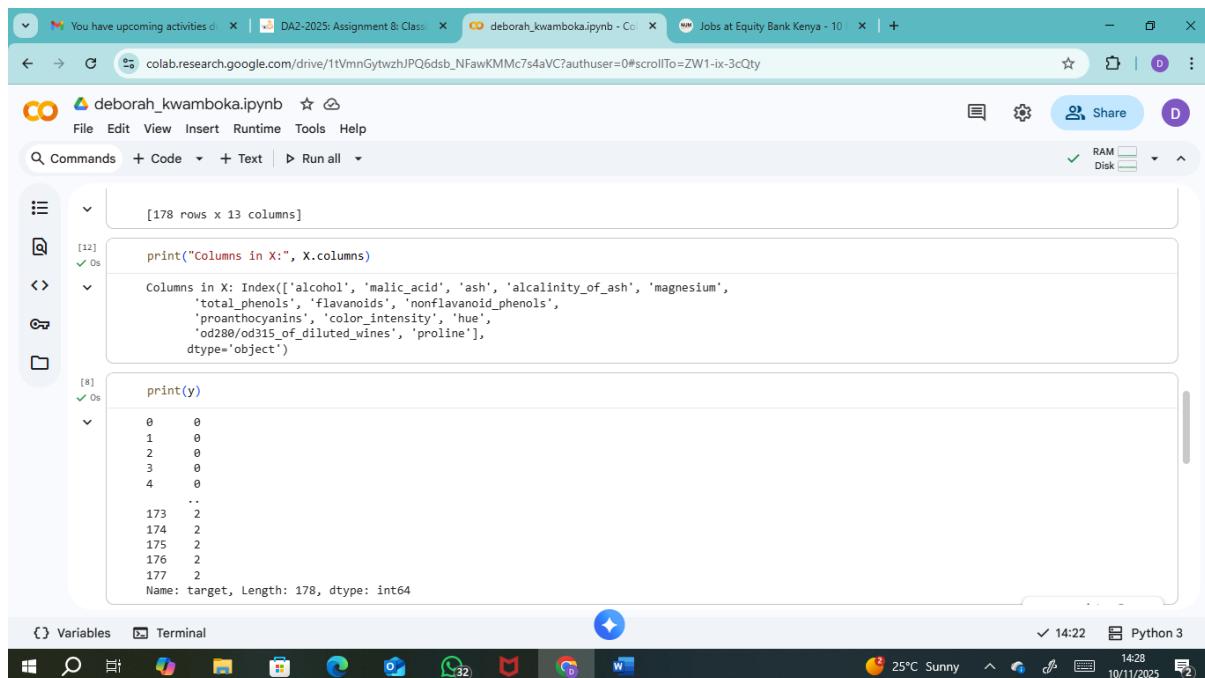
The screenshot shows a Jupyter Notebook interface with the following code and output:

```
#exploring the wine dataset
print(X)
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols
0	14.23	1.71	2.43	15.6	127.0	2.80
1	13.20	1.78	2.14	11.2	100.0	2.65
2	13.16	2.36	2.67	18.6	101.0	2.80
3	14.37	1.95	2.50	16.8	113.0	3.85
4	13.24	2.59	2.87	21.0	118.0	2.80
..
173	13.71	5.65	2.45	20.5	95.0	1.68
174	13.40	3.91	2.48	23.0	102.0	1.80
175	13.27	4.28	2.26	20.0	120.0	1.59
176	13.17	2.59	2.37	20.0	120.0	1.65
177	14.13	4.10	2.74	24.5	96.0	2.05

	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue
0	3.06	0.28	2.29	5.64	1.04
1	2.76	0.26	1.28	4.38	1.05
2	3.24	0.30	2.81	5.68	1.03
3	3.49	0.24	2.18	7.80	0.86
4	2.69	0.39	1.82	4.32	1.04
..
173	0.61	0.52	1.06	7.70	0.64
174	0.75	0.43	1.41	7.30	0.70
175	0.69	0.43	1.35	10.20	0.59
176	0.68	0.53	1.46	9.30	0.60

Figure 2: screenshot displaying x



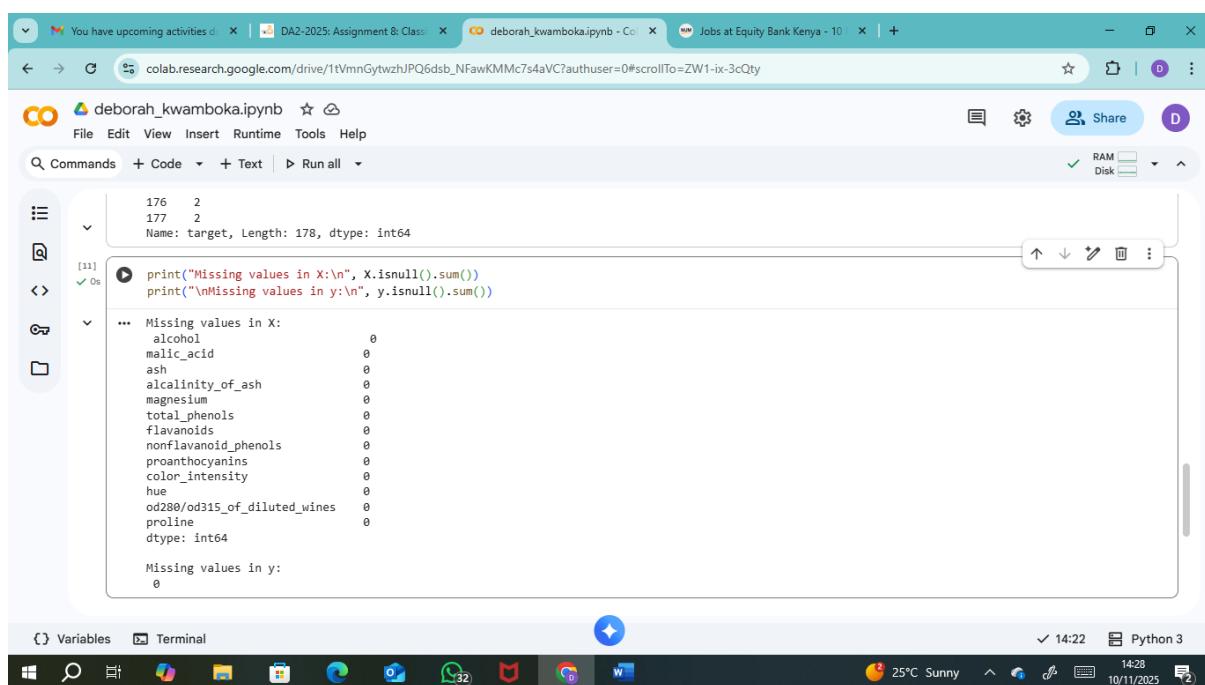
The screenshot shows a Jupyter Notebook interface on Google Colab. The code cell at the top prints the columns of the dataset X:

```
Columns in X: Index(['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline'], dtype='object')
```

The code cell below prints the target variable y:

```
print(y)
0    0
1    0
2    0
3    0
4    0
 ..
173   2
174   2
175   2
176   2
177   2
Name: target, Length: 178, dtype: int64
```

Figure 3: screenshot displaying y



The screenshot shows a Jupyter Notebook interface on Google Colab. The code cell at the top prints the missing values in the target variable y:

```
Missing values in y:
0
```

The code cell below prints the missing values in the features of dataset X:

```
print("Missing values in X:\n", X.isnull().sum())
print("\nMissing values in y:\n", y.isnull().sum())

... Missing values in X:
alcohol          0
malic_acid      0
ash              0
alcalinity_of_ash 0
magnesium        0
total_phenols    0
flavanoids       0
nonflavanoid_phenols 0
proanthocyanins 0
color_intensity  0
hue              0
od280/od315_of_diluted_wines 0
proline          0
dtype: int64

Missing values in y:
0
```

Figure 4: screenshot showing the dataset has no missing values

Step 3: Prepare the data

In this step I prepared my data. I scaled the values of x to ensure all features have a similar scale and then I split my data into training set and testing.

Code:

```
#scaling features in x

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

# Train/Test split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,
random_state=42)
```

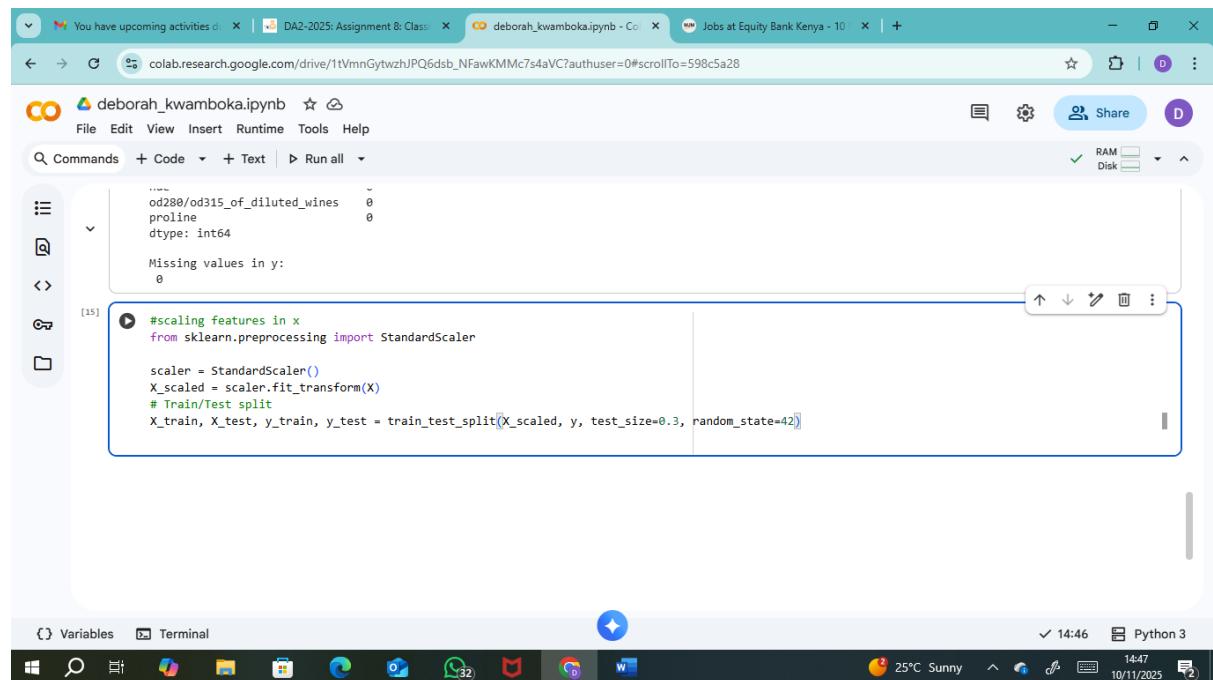
Screenshot:


Figure 5: screenshot to the above code

Step 4: Build the models and evaluate them

Logistic regression

I built the logistic regression model and evaluated it using metrics such as accuracy, recall and F1_score.

Code:

```

lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
print("Logistic Regression\n", classification_report(y_test, y_pred_lr))
plot_conf_matrix(y_test, y_pred_lr, "Logistic Regression")
results.loc[len(results)] = ['Logistic Regression', accuracy_score(y_test, y_pred_lr)]

```

screenshot:

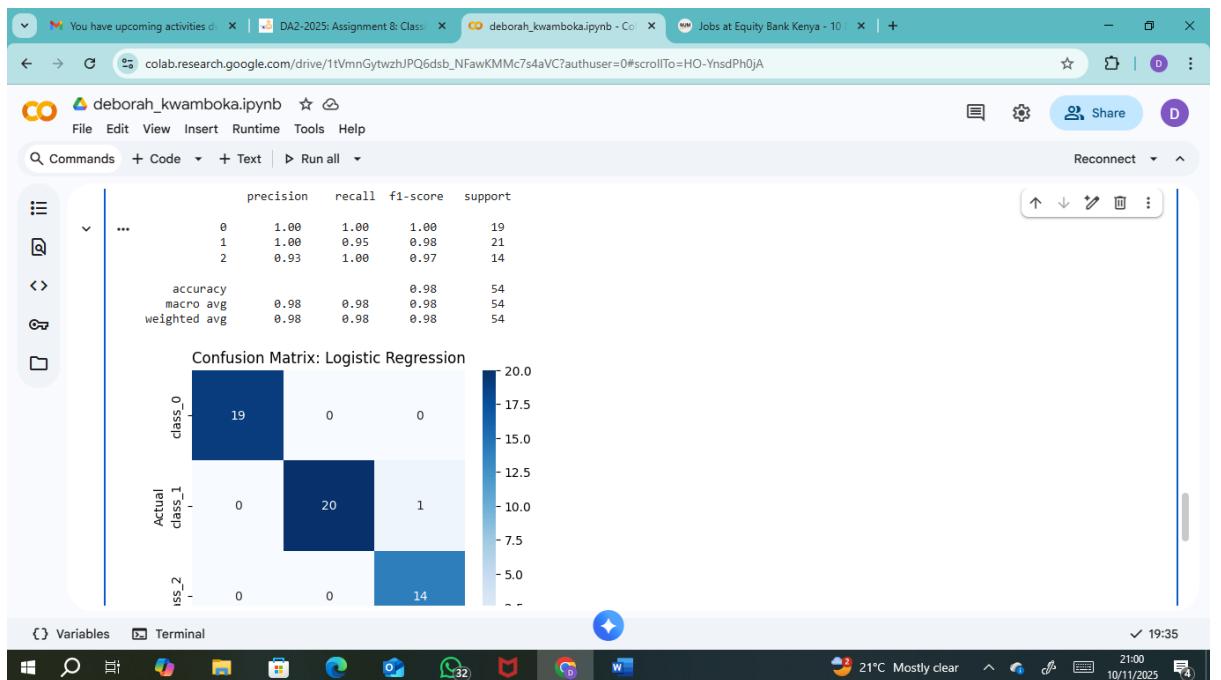


Figure 6: screenshot showing the confusion matrix

Decision tree model

Code:

```

## 2. Decision Tree

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
print("Decision Tree\n", classification_report(y_test, y_pred_dt))
plot_conf_matrix(y_test, y_pred_dt, "Decision Tree")

```

```
results.loc[len(results)] = ['Decision Tree', accuracy_score(y_test, y_pred_dt)]
```

Screenshot:

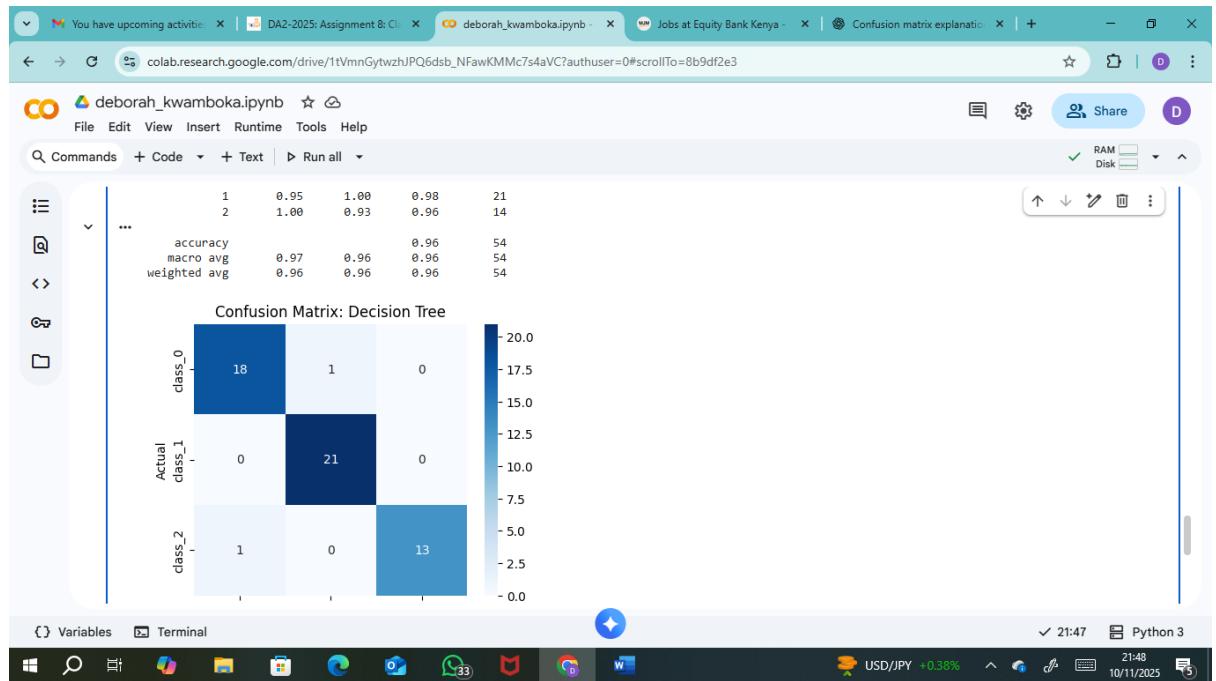


Figure 7: screenshot for decision tree model

Random forest

Code:

```
## 3. Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest\n", classification_report(y_test, y_pred_rf))
plot_conf_matrix(y_test, y_pred_rf, "Random Forest")
results.loc[len(results)] = ['Random Forest', accuracy_score(y_test, y_pred_rf)]
```

Screenshot:

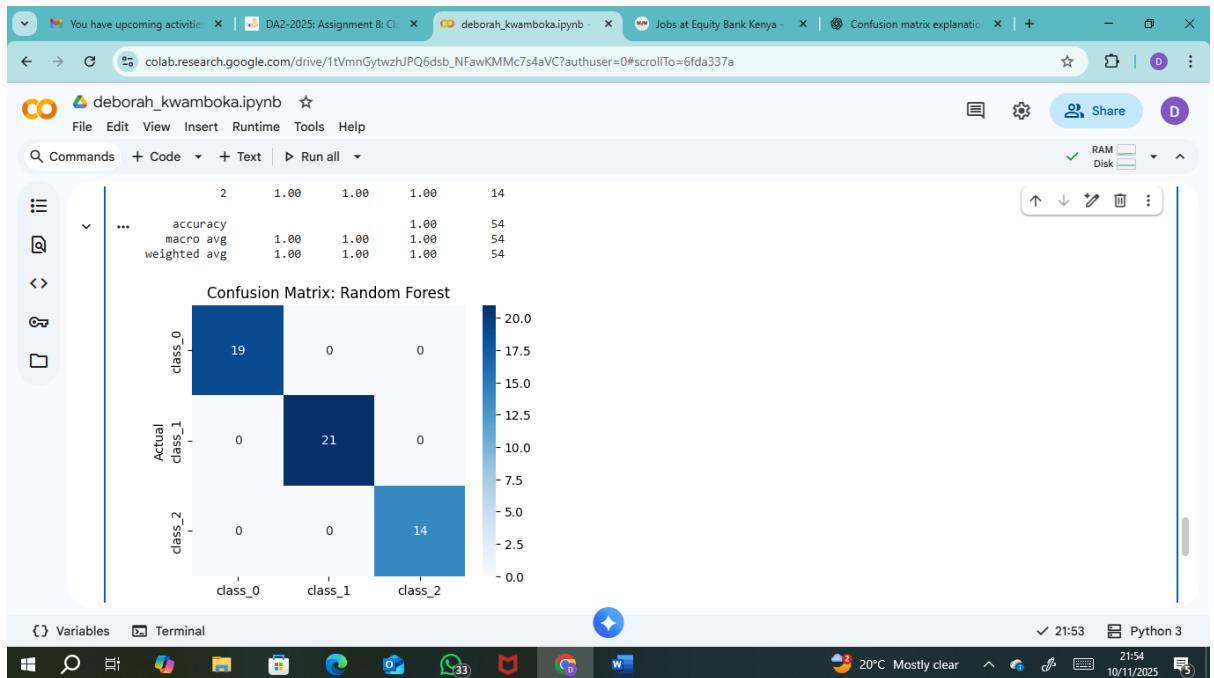


Figure 8: screenshot for random forest model

K nearest neighbours

Code:

```
# 4. K-Nearest Neighbors

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
print("K-Nearest Neighbors\n", classification_report(y_test, y_pred_knn))
plot_conf_matrix(y_test, y_pred_knn, "K-Nearest Neighbors")
results.loc[len(results)] = ['K-Nearest Neighbors', accuracy_score(y_test, y_pred_knn)]
```

screenshot:

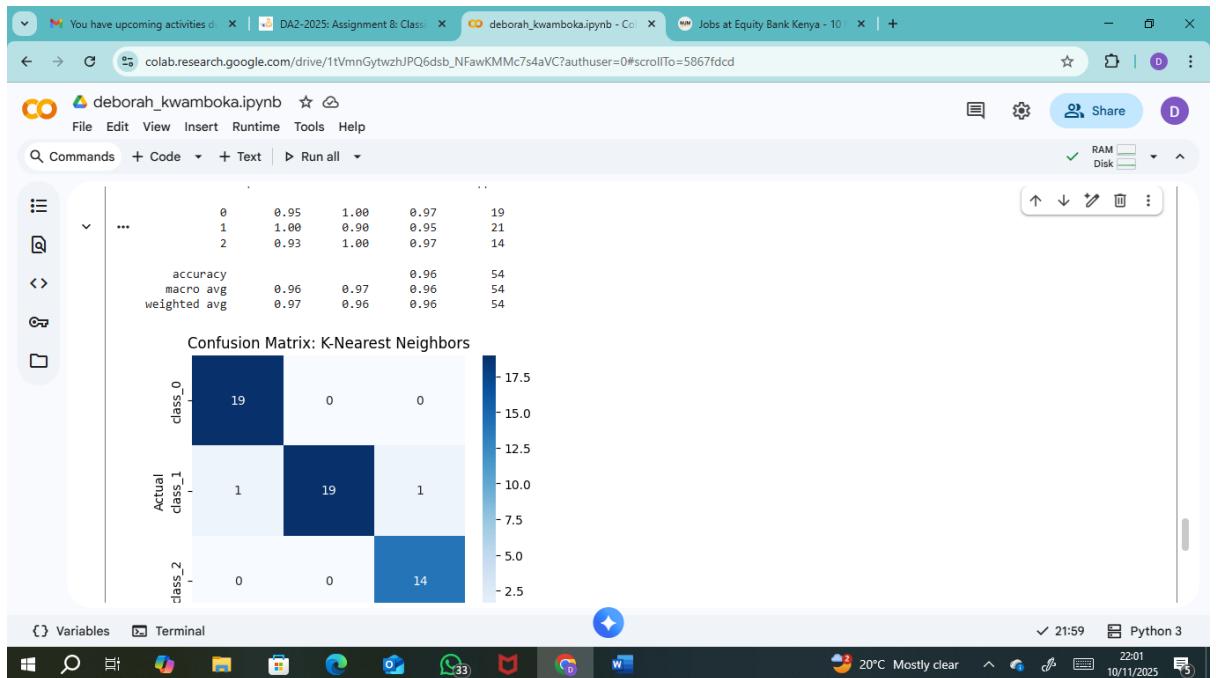


Figure 9: screenshot for k- nearest neighbors

Naïve bayes

Code:

```

# 5. Naive Bayes

nb = GaussianNB()

nb.fit(X_train, y_train)

y_pred_nb = nb.predict(X_test)

print("Naive Bayes\n", classification_report(y_test, y_pred_nb))

plot_conf_matrix(y_test, y_pred_nb, "Naive Bayes")

results.loc[len(results)] = ['Naive Bayes', accuracy_score(y_test, y_pred_nb)]

```

Screenshot:

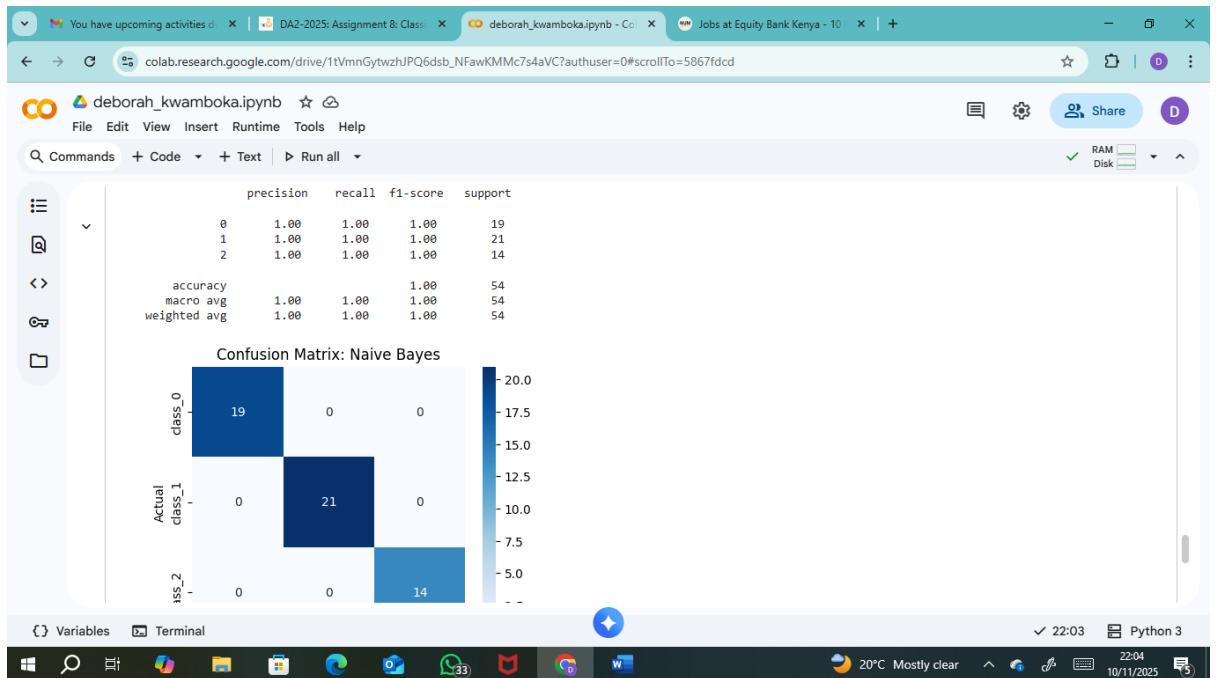


Figure 10: screenshot for naive bayes

Support vector machine

Code:

```
## 6. Support Vector Machine
svc = SVC(random_state=42)
svc.fit(X_train, y_train)
y_pred_svc = svc.predict(X_test)
print("Support Vector Machine\n", classification_report(y_test, y_pred_svc))
plot_conf_matrix(y_test, y_pred_svc, "Support Vector Machine")
results.loc[len(results)] = ['Support Vector Machine', accuracy_score(y_test,
y_pred_svc)]
```

Screenshot:

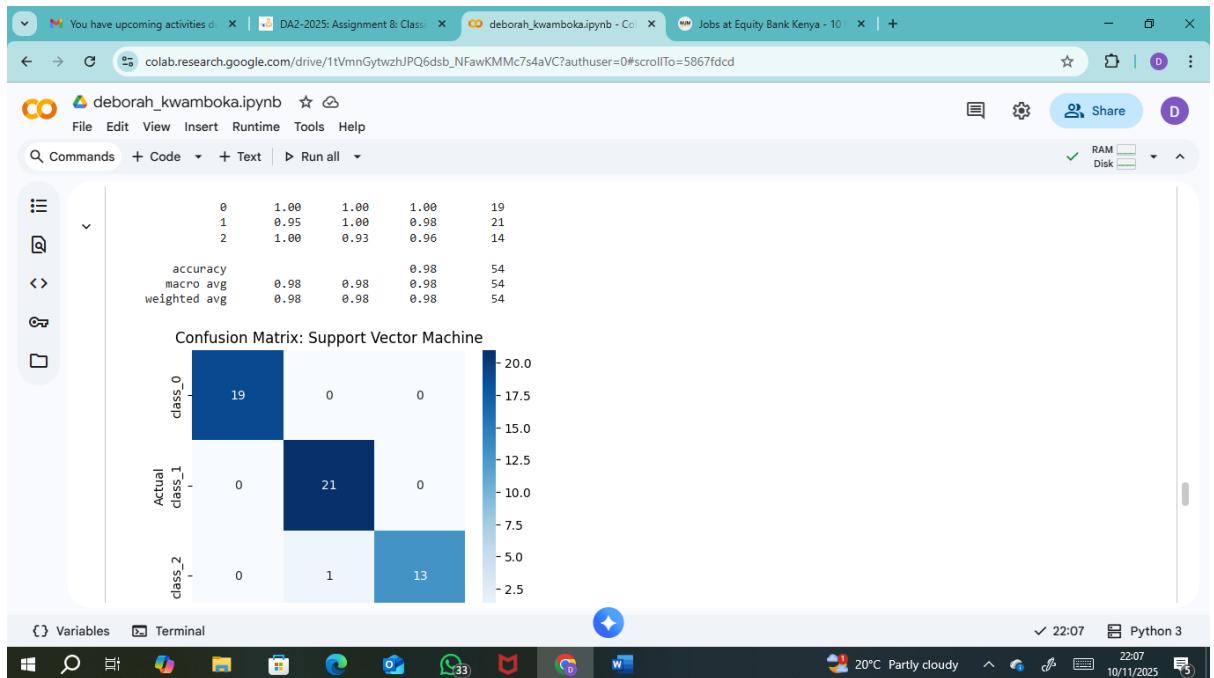


Figure 11: screenshot for support vector machine

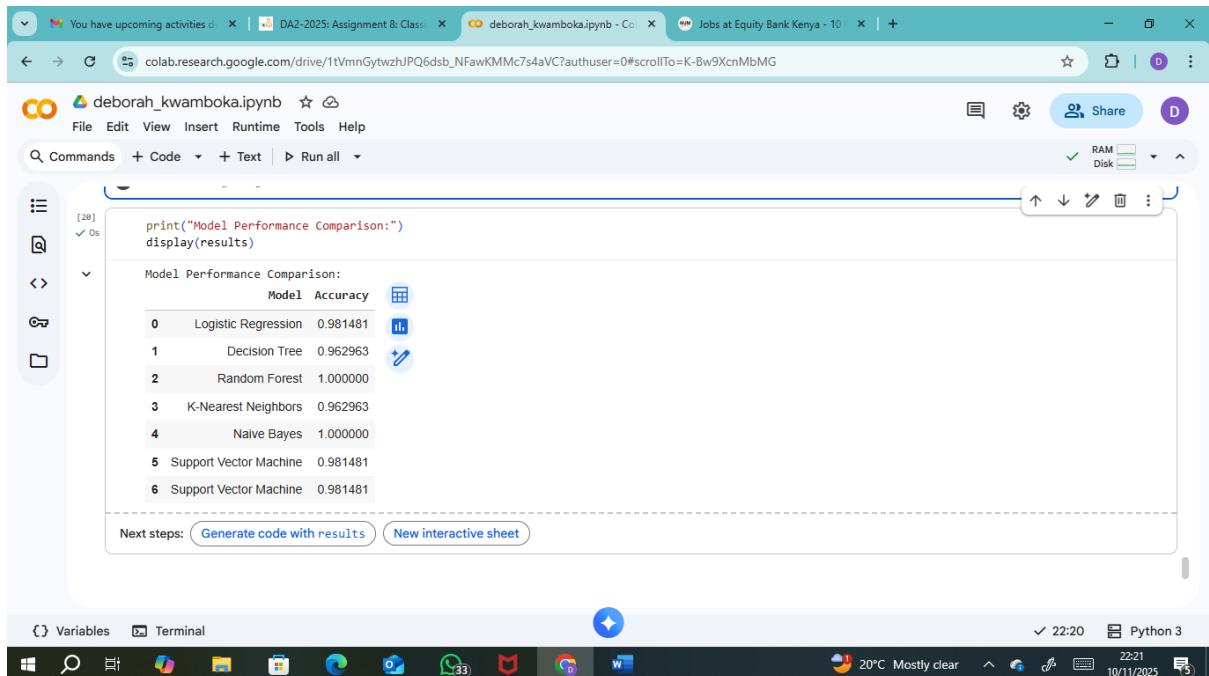
Step 5: model comparison

Both the Random Forest and Naive Bayes models achieved the highest accuracy, with a score of 1.00 on the test set. This means they correctly classified all instances in the test set.

The Logistic Regression and Support Vector Machine models also performed very well, with an accuracy of 0.981481.

The Decision Tree and K-Nearest Neighbors models had slightly lower accuracies of 0.962963.

Therefore, based solely on the accuracy metric on this specific test set, the Random Forest and Naive Bayes models performed the best.



	Model	Accuracy
0	Logistic Regression	0.981481
1	Decision Tree	0.962963
2	Random Forest	1.000000
3	K-Nearest Neighbors	0.962963
4	Naive Bayes	1.000000
5	Support Vector Machine	0.981481
6	Support Vector Machine	0.981481

Figure 12: screenshot showing model comparison

Link to code

Link to Code:

https://colab.research.google.com/drive/1tVmnnGytwzhJPQ6dsb_NFawKMMc7s4aVC?usp=sharing

Conclusion

In this assignment, I successfully applied supervised machine learning techniques to classify wine types using the scikit-learn Wine dataset. The process involved exploring and understanding the dataset through exploratory data analysis (EDA), preparing the data for modelling, and implementing six different classification algorithms: Logistic Regression, Decision Tree, Random Forest, k-Nearest Neighbors (KNN), Naive Bayes, and Support Vector Machine (SVM). This exercise reinforced the importance of model evaluation and comparison in machine learning. It highlighted that while accuracy is important, other metrics such as precision, recall, and the confusion matrix are equally valuable for understanding model behaviour.

