

LAB 2

AIDI 2004

Declan Trevor Kintu

100944330 PROFESSOR MOHAMMAD AL-TAWALBEH

Table of Contents

Step 9.	2
Step 2 and Step 3.	2
<i>First Model:</i>	2
<i>Second Model:</i>	10
Step 3, Step 4, and Step 5.	21
Step 6.	23
Step 7.	24
Step 8.	25

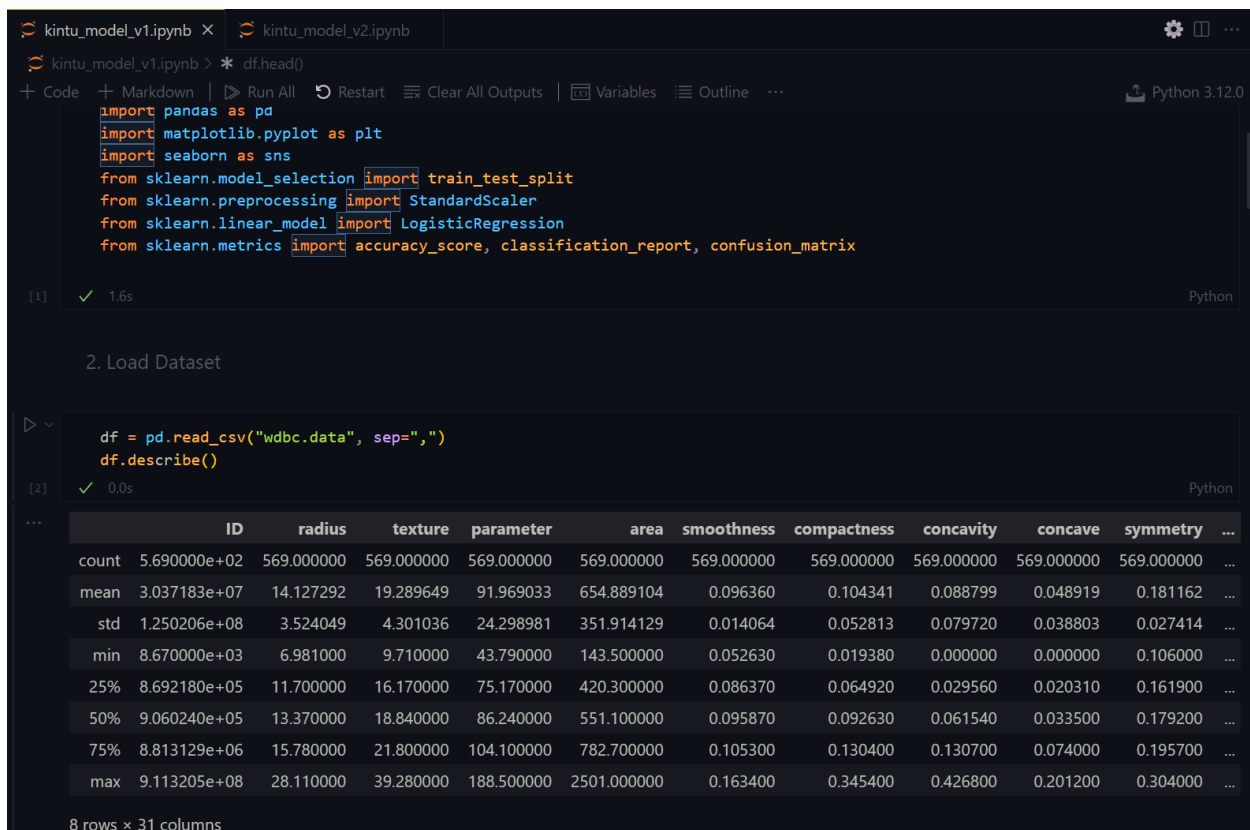
Step 9.

You can view the repository by following this link: [MrKintu/test-lab2: Machine Learning Models use Logistic Linear Regression and Decision Trees to analyse and predict breast cancer in women. \(github.com\)](https://github.com/MrKintu/test-lab2).

Step 2 and Step 3.

First Model:

This model uses Logistic Linear Regression to perform machine learning on the dataset. The first step is to import the libraries and perform an exploratory analysis of the data.



The screenshot shows a Jupyter Notebook interface with two tabs: 'kintu_model_v1.ipynb' and 'kintu_model_v2.ipynb'. The active tab is 'kintu_model_v1.ipynb'. The code cell [1] contains the following imports:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

The code cell [2] contains the following code to load the dataset:

```
df = pd.read_csv("wdbc.data", sep=",")
df.describe()
```

The output of the code cell [2] is a summary statistics table for the dataset. The table has 11 columns: ID, radius, texture, parameter, area, smoothness, compactness, concavity, concave, symmetry, and ... (truncated). The rows show count, mean, std, min, 25%, 50%, 75%, and max for each column.

	ID	radius	texture	parameter	area	smoothness	compactness	concavity	concave	symmetry	...
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...

8 rows x 31 columns

kintu_model_v1.ipynb

kintu_model_v2.ipynb

⚙️ □ ...

kintu_model_v1.ipynb

> M*Explore Current Data Set

+ Code

+ Markdown

▶ Run All

↺ Restart

☰ Clear All Outputs

📄 Variables

☰ Outline

...

Python 3.12.0

min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...

8 rows × 31 columns

▶ ▾

df.head()

[3] ✓ 0.0s

Python

...

	ID	diagnosis	radius	texture	parameter	area	smoothness	compactness	concavity	concave	...	radius_bad	textu
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	25.38	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54	

5 rows × 32 columns

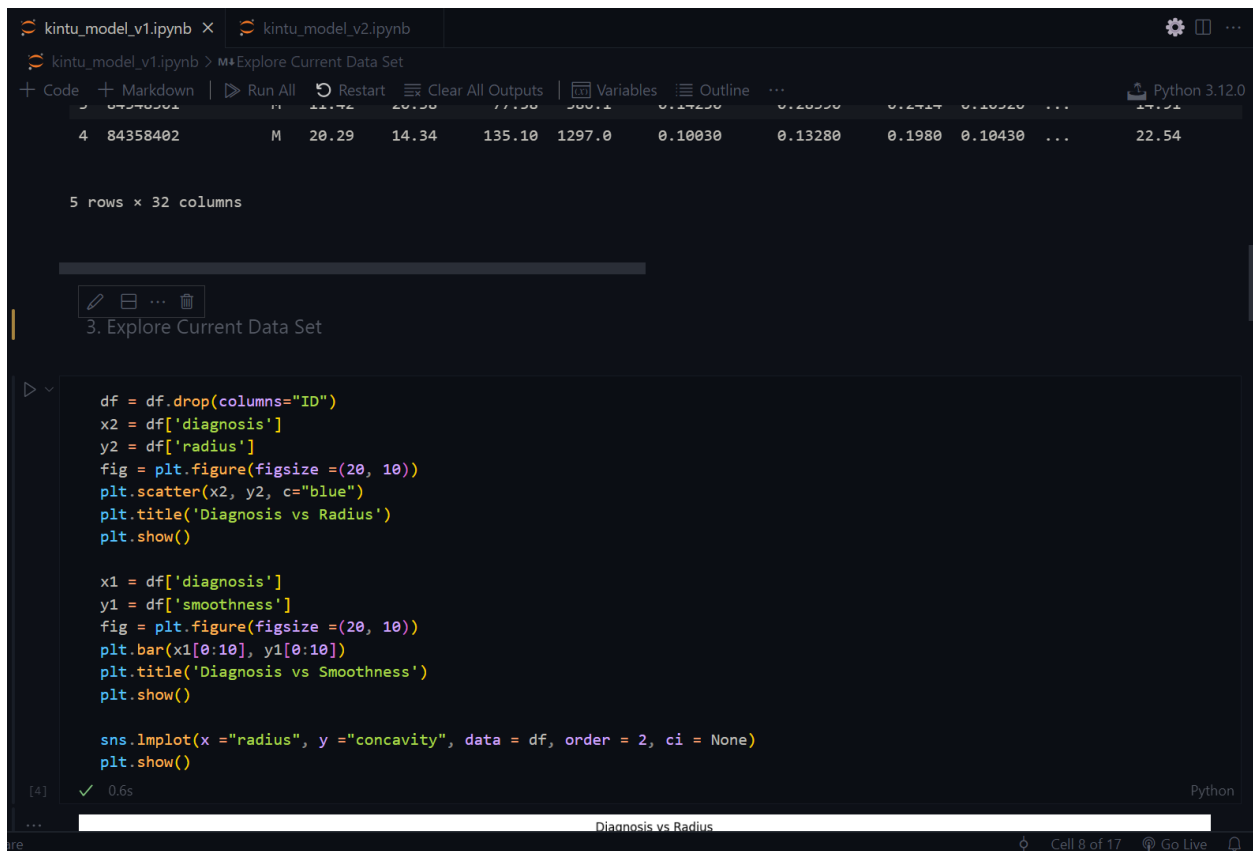
pre

Cell 8 of 17

Go Live

🔔

The second step is to plot the graphs describing the data.



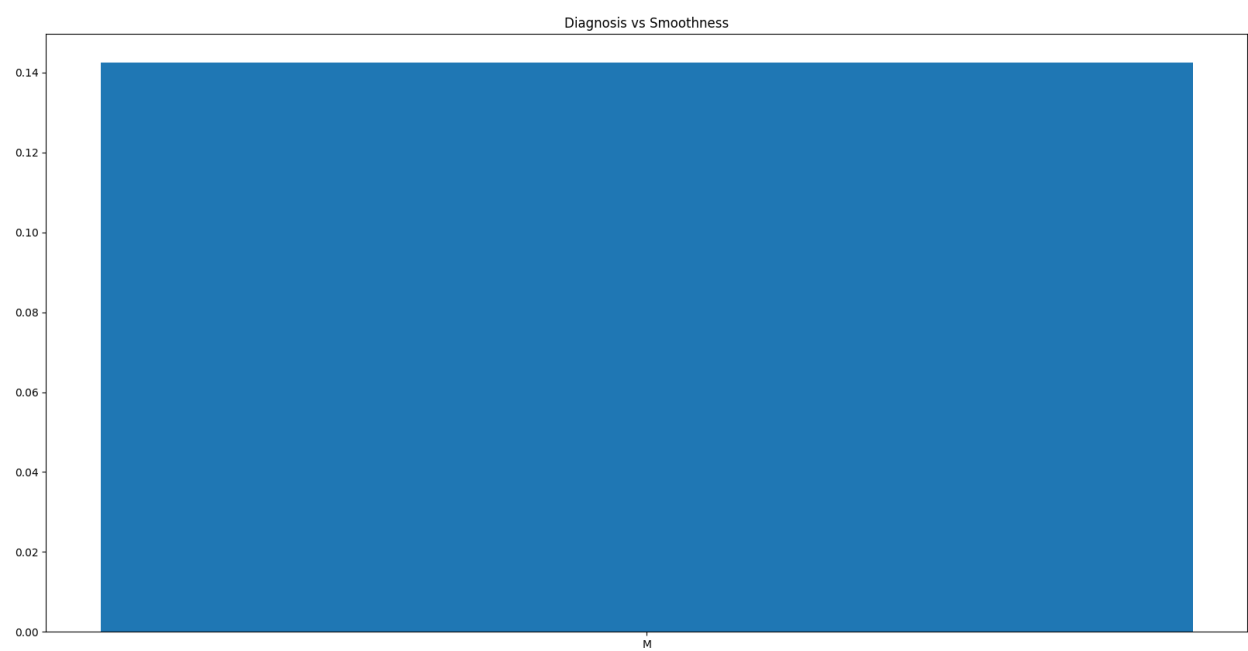
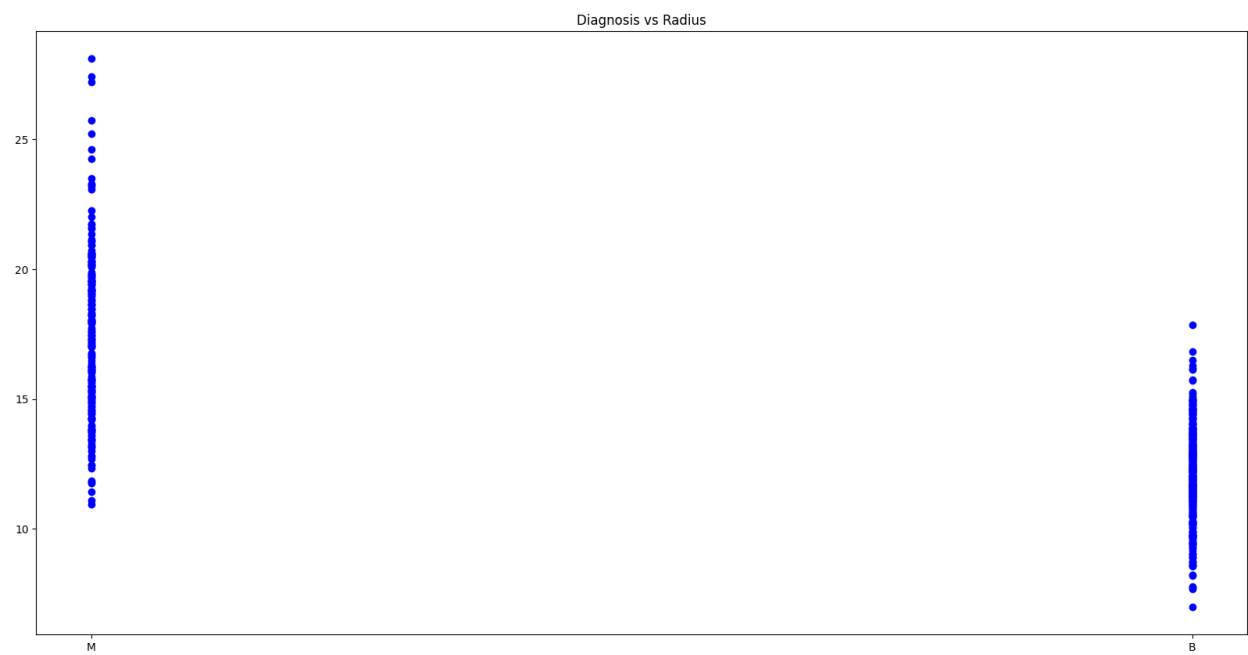
The screenshot shows a Jupyter Notebook with two tabs: 'kintu_model_v1.ipynb' and 'kintu_model_v2.ipynb'. The active tab is 'kintu_model_v2.ipynb', which displays a data preview table with 5 rows and 32 columns. The table includes columns for 'ID', 'diagnosis', 'radius', 'smoothness', 'concavity', and 'area'. Below the table, the text '5 rows x 32 columns' is shown. The notebook interface includes a toolbar with options like 'Code', 'Markdown', 'Run All', 'Restart', 'Clear All Outputs', 'Variables', 'Outline', and 'Python 3.12.0'. The code cell contains the following Python code:

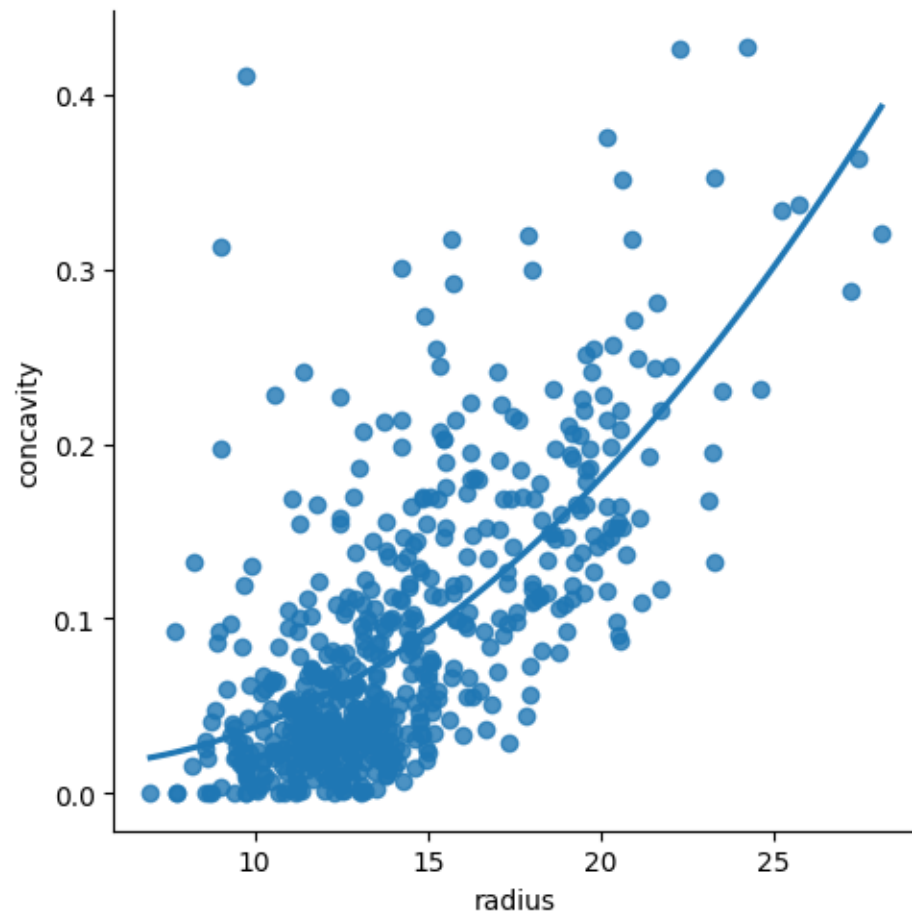
```
df = df.drop(columns="ID")
x2 = df['diagnosis']
y2 = df['radius']
fig = plt.figure(figsize=(20, 10))
plt.scatter(x2, y2, c="blue")
plt.title('Diagnosis vs Radius')
plt.show()

x1 = df['diagnosis']
y1 = df['smoothness']
fig = plt.figure(figsize=(20, 10))
plt.bar(x1[0:10], y1[0:10])
plt.title('Diagnosis vs Smoothness')
plt.show()

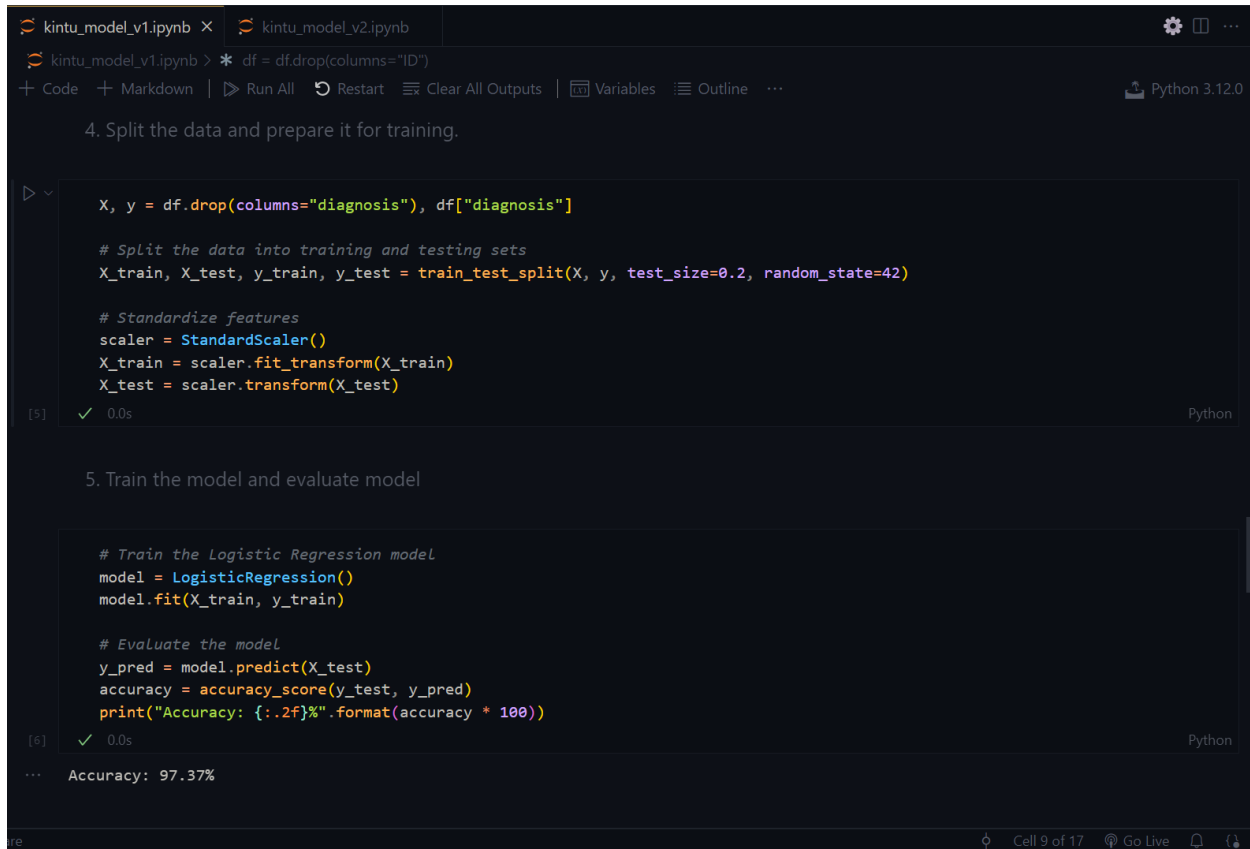
sns.lmplot(x="radius", y="concavity", data = df, order = 2, ci = None)
plt.show()
```

The code cell is executed, showing a status bar with '[4]', a green checkmark, and '0.6s'. The output area at the bottom shows a plot titled 'Diagnosis vs Radius'.





After which, the data is split into training and testing sets. The data that falls under the training set shall be pre-processed by going through a Standard Scaler. We can also make predictions and estimate the accuracy of our model.



```
kintu_model_v1.ipynb x kintu_model_v2.ipynb
kintu_model_v1.ipynb > * df = df.drop(columns="ID")
+ Code + Markdown | ▶ Run All ⏮ Restart ⏭ Clear All Outputs | 📄 Variables 📖 Outline ... Python 3.12.0

4. Split the data and prepare it for training.

X, y = df.drop(columns="diagnosis"), df["diagnosis"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

[5] ✓ 0.0s Python

5. Train the model and evaluate model

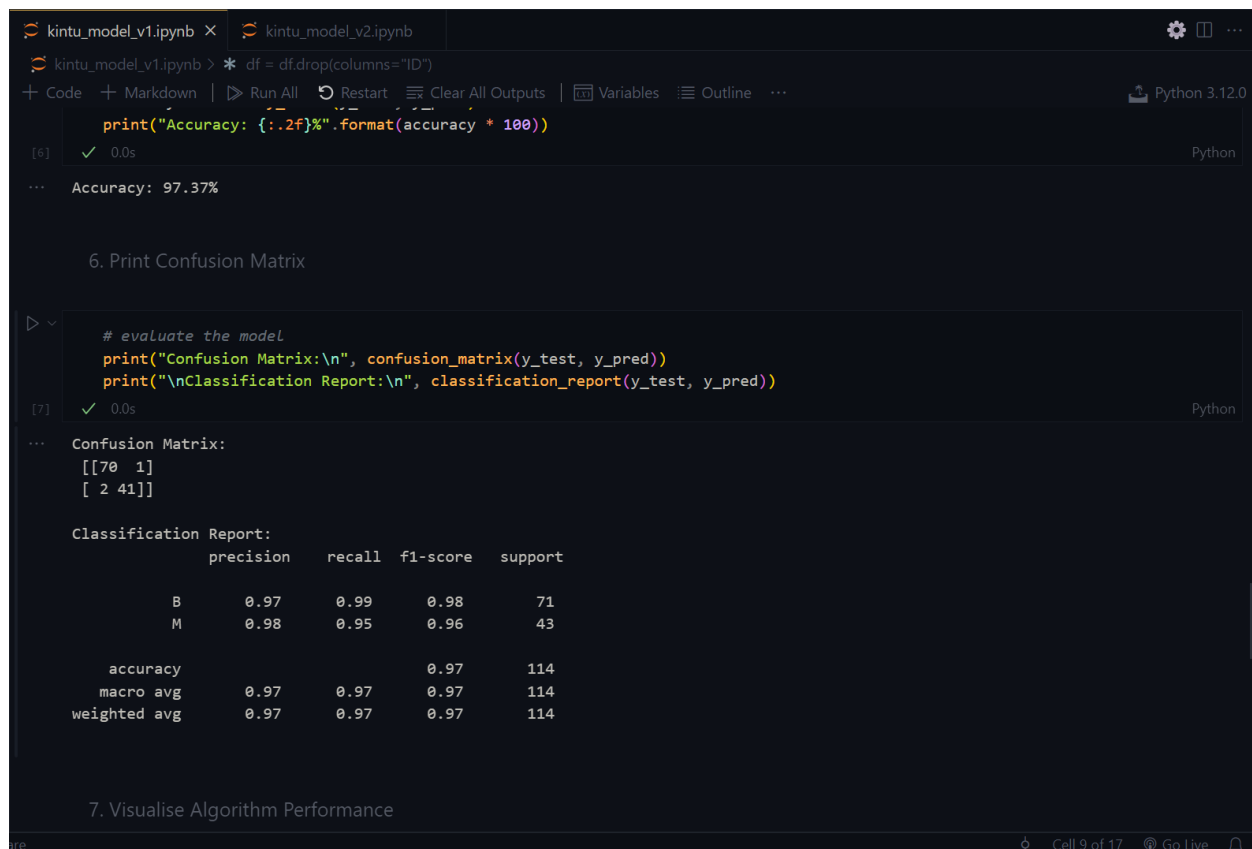
# Train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))

[6] ✓ 0.0s Python

... Accuracy: 97.37%
```


Lastly, we create the confusion matrix and the classification report for the algorithm. Our model is graphed using a scatter plot.



The image shows a Jupyter Notebook interface with two tabs: 'kintu_model_v1.ipynb' and 'kintu_model_v2.ipynb'. The active tab is 'kintu_model_v1.ipynb'. The notebook contains two code cells. The first cell, labeled '[6]', contains the code `print("Accuracy: {:.2f}%".format(accuracy * 100))` and has been executed, resulting in the output 'Accuracy: 97.37%'. The second cell, labeled '[7]', contains the code `# evaluate the model`, `print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))`, and `print("\nClassification Report:\n", classification_report(y_test, y_pred))`. This cell has also been executed, resulting in the output of a Confusion Matrix and a Classification Report. The Confusion Matrix is a 2x2 array: `[[70 1]` and `[2 41]]`. The Classification Report is a table with columns for precision, recall, f1-score, and support, and rows for classes B and M, and overall averages.

```
[6] ✓ 0.0s
... Accuracy: 97.37%
```

6. Print Confusion Matrix

```
[7] ✓ 0.0s
... Confusion Matrix:
[[70 1]
 [ 2 41]]

Classification Report:
      precision    recall  f1-score   support

     B       0.97       0.99       0.98        71
     M       0.98       0.95       0.96        43

   accuracy       0.97       0.97       0.97       114
  macro avg       0.97       0.97       0.97       114
weighted avg       0.97       0.97       0.97       114
```

7. Visualise Algorithm Performance

```
kintu_model_v1.ipynb X kintu_model_v2.ipynb
kintu_model_v1.ipynb > * df = df.drop(columns="ID")
+ Code + Markdown | ▶ Run All ⏹ Restart ⌵ Clear All Outputs | 📄 Variables 📄 Outline ... Python 3.12.0
[7] ✓ 0.0s

... Confusion Matrix:
[[70  1]
 [ 2 41]]

Classification Report:
              precision    recall  f1-score   support

     B       0.97       0.99       0.98         71
     M       0.98       0.95       0.96         43

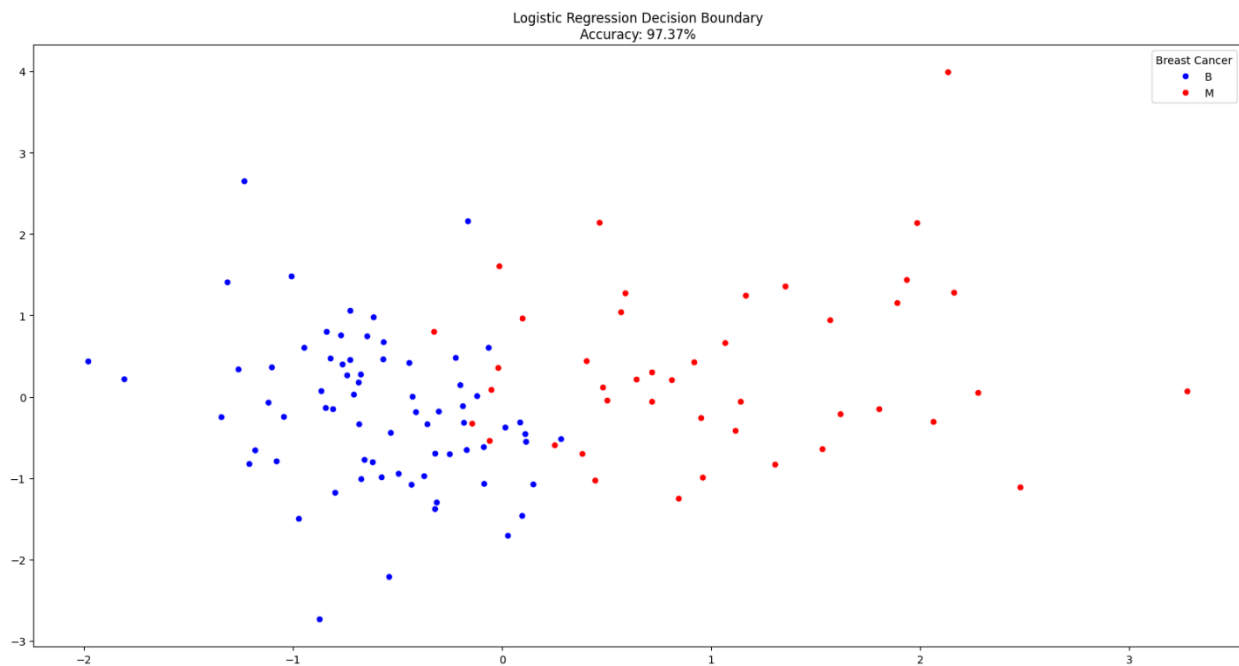
 accuracy       0.97       0.97       0.97        114
  macro avg       0.97       0.97       0.97        114
 weighted avg       0.97       0.97       0.97        114

+ Code + Markdown

7. Visualise Algorithm Performance

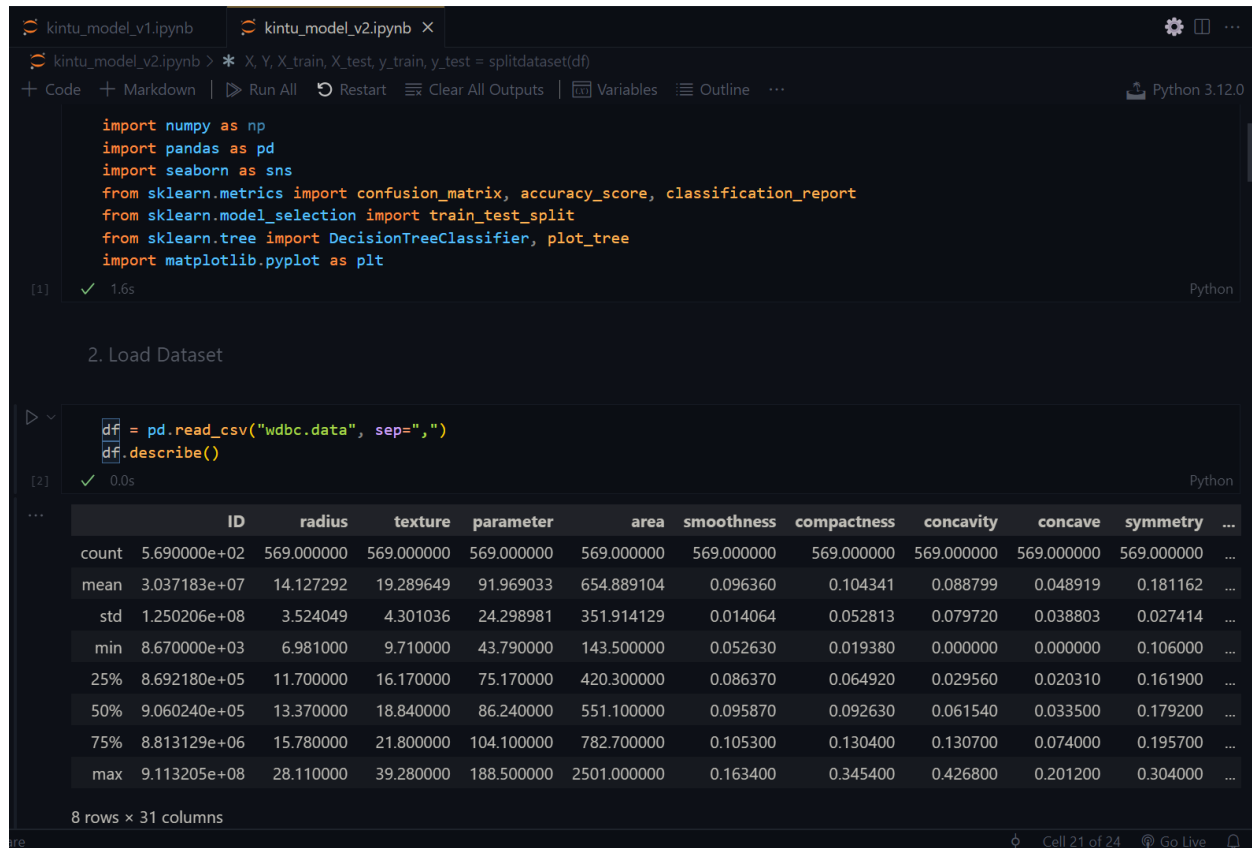
# Visualize the decision boundary with accuracy information
plt.figure(figsize=(20, 10))
sns.scatterplot(x=X_test[:, 2], y=X_test[:, 8], hue=y_test, palette={
    'B': 'blue', 'M': 'red'}, marker='o')
# plt.xlabel("BMI")
# plt.ylabel("Age")
plt.title("Logistic Regression Decision Boundary\nAccuracy: {:.2f}%".format(
    accuracy * 100))
plt.legend(title="Breast Cancer", loc="upper right")
plt.show()

[8] ✓ 0.2s Python
```



Second Model:

This machine learning algorithm is implemented using Decision Tree Classification. The first step is to load the necessary libraries and display the current dataset.



The image shows a Jupyter Notebook interface with two tabs: 'kintu_model_v1.ipynb' and 'kintu_model_v2.ipynb'. The active tab is 'kintu_model_v2.ipynb'. The code cell [1] contains the following imports:

```
* X, Y, X_train, X_test, y_train, y_test = splitdataset(df)

import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
```

The code cell [2] contains the following code:

```
df = pd.read_csv("wdbc.data", sep=",")
df.describe()
```

The output of the code cell [2] is a summary statistics table for the dataset. The table has 11 columns: ID, radius, texture, parameter, area, smoothness, compactness, concavity, concave, symmetry, and ... (truncated). The rows represent different statistical measures: count, mean, std, min, 25%, 50%, 75%, and max.

	ID	radius	texture	parameter	area	smoothness	compactness	concavity	concave	symmetry	...
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...

8 rows x 31 columns

kintu_model_v1.ipynb

kintu_model_v2.ipynb X

kintu_model_v2.ipynb

X, Y, X_train, X_test, y_train, y_test = splitdataset(df)

+ Code

+ Markdown

Run All

Restart

Clear All Outputs

Variables

Outline

...

Python 3.12.0

min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...

8 rows x 31 columns

+ Code

+ Markdown

df.head()

[3] ✓ 0.0s Python

	ID	diagnosis	radius	texture	parameter	area	smoothness	compactness	concavity	concave	...	radius_bad	texture_bad
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	25.38	17.33
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99	23.41
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57	25.53
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91	26.50
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54	16.67

5 rows x 32 columns

3. Explore Current Dataset

Cell 21 of 24

Go Live

The second step is to plot graphs describing the current dataset.

```
kintu_model_v1.ipynb | kintu_model_v2.ipynb X
```

```
X, Y, X_train, X_test, y_train, y_test = splitdataset(df)
```

1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99	23.41	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57	25.53	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91	26.50	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54	16.67	

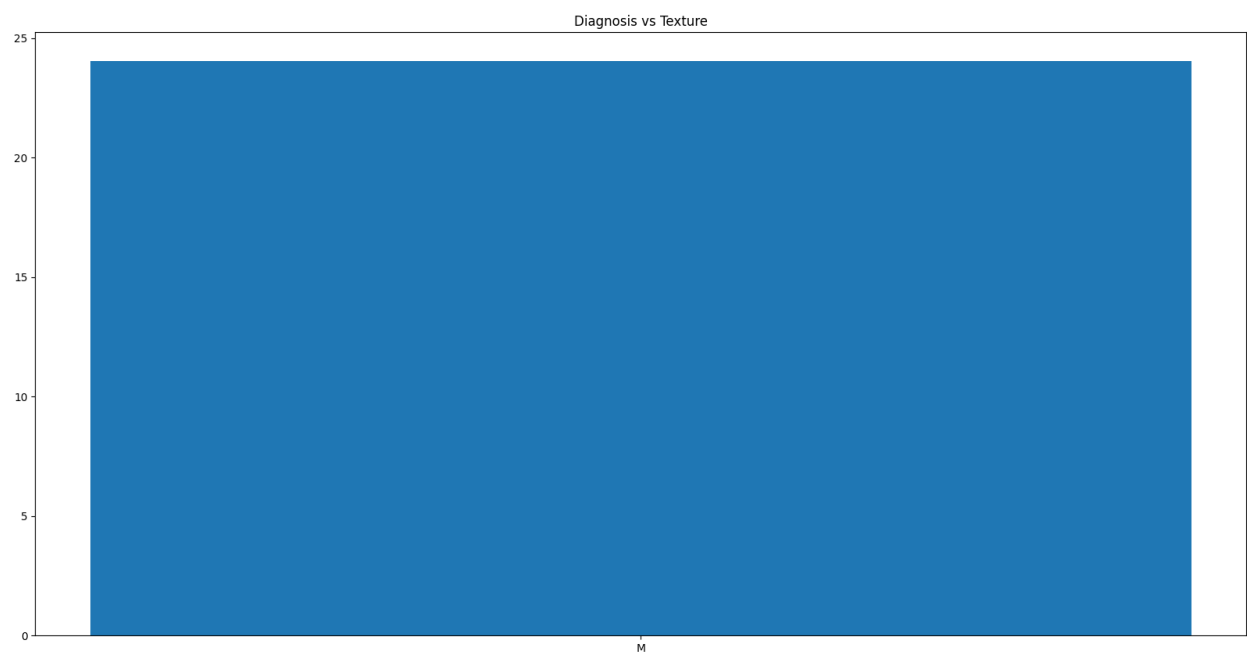
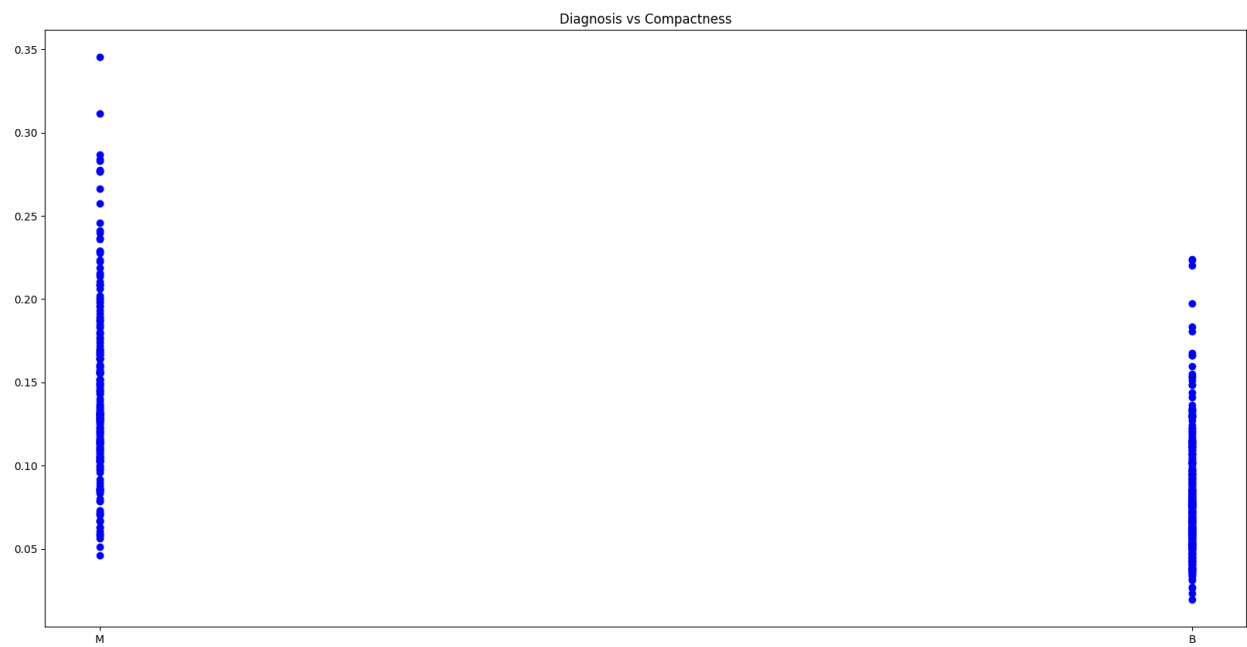
5 rows × 32 columns

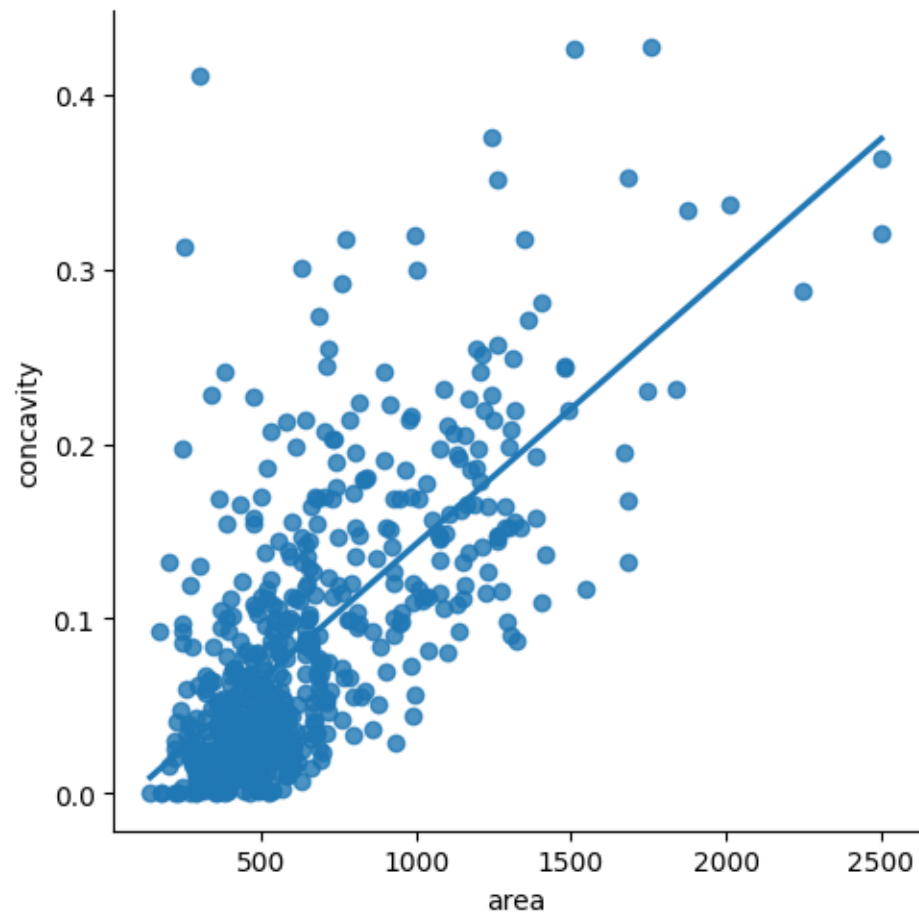
3. Explore Current Dataset

+ Code + Markdown

```
df = df.drop(columns="ID")  
x2 = df['diagnosis']  
y2 = df['compactness']  
fig = plt.figure(figsize=(20, 10))  
plt.scatter(x2, y2, c="blue")  
plt.title('Diagnosis vs Compactness')  
plt.show()  
  
x1 = df['diagnosis']  
y1 = df['texture']  
fig = plt.figure(figsize=(20, 10))  
plt.bar(x1[0:10], y1[0:10])  
plt.title('Diagnosis vs Texture')  
plt.show()  
  
sns.lmplot(x="area", y="concavity", data=df, order=2, ci=None)  
plt.show()
```

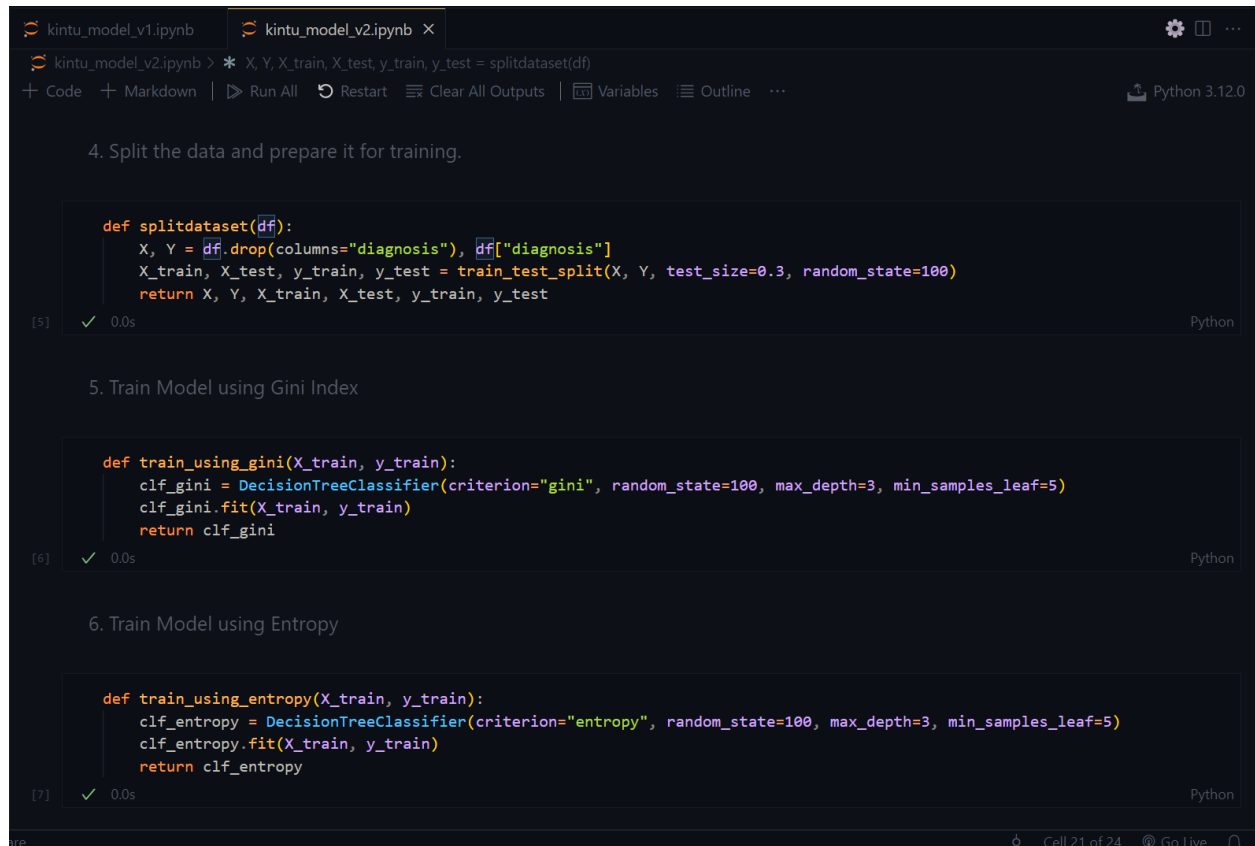
[4] ✓ 0.5s Python





Next, the dataset is split into training and test data. This ML algorithm shall be trained using two variations (Gini Index and Entropy) so that we may be able to compare the performance of the two variations.

The source code that describes the training of the model using both variations is added.



The image shows a Jupyter Notebook interface with three tabs: 'kintu_model_v1.ipynb', 'kintu_model_v2.ipynb' (active), and 'kintu_model_v3.ipynb'. The active tab contains the following code:

```
def splitdataset(df):  
    X, Y = df.drop(columns="diagnosis"), df["diagnosis"]  
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=100)  
    return X, Y, X_train, X_test, y_train, y_test
```

Cell [5] is executed successfully (0.0s).

4. Split the data and prepare it for training.

```
def train_using_gini(X_train, y_train):  
    clf_gini = DecisionTreeClassifier(criterion="gini", random_state=100, max_depth=3, min_samples_leaf=5)  
    clf_gini.fit(X_train, y_train)  
    return clf_gini
```

Cell [6] is executed successfully (0.0s).

5. Train Model using Gini Index

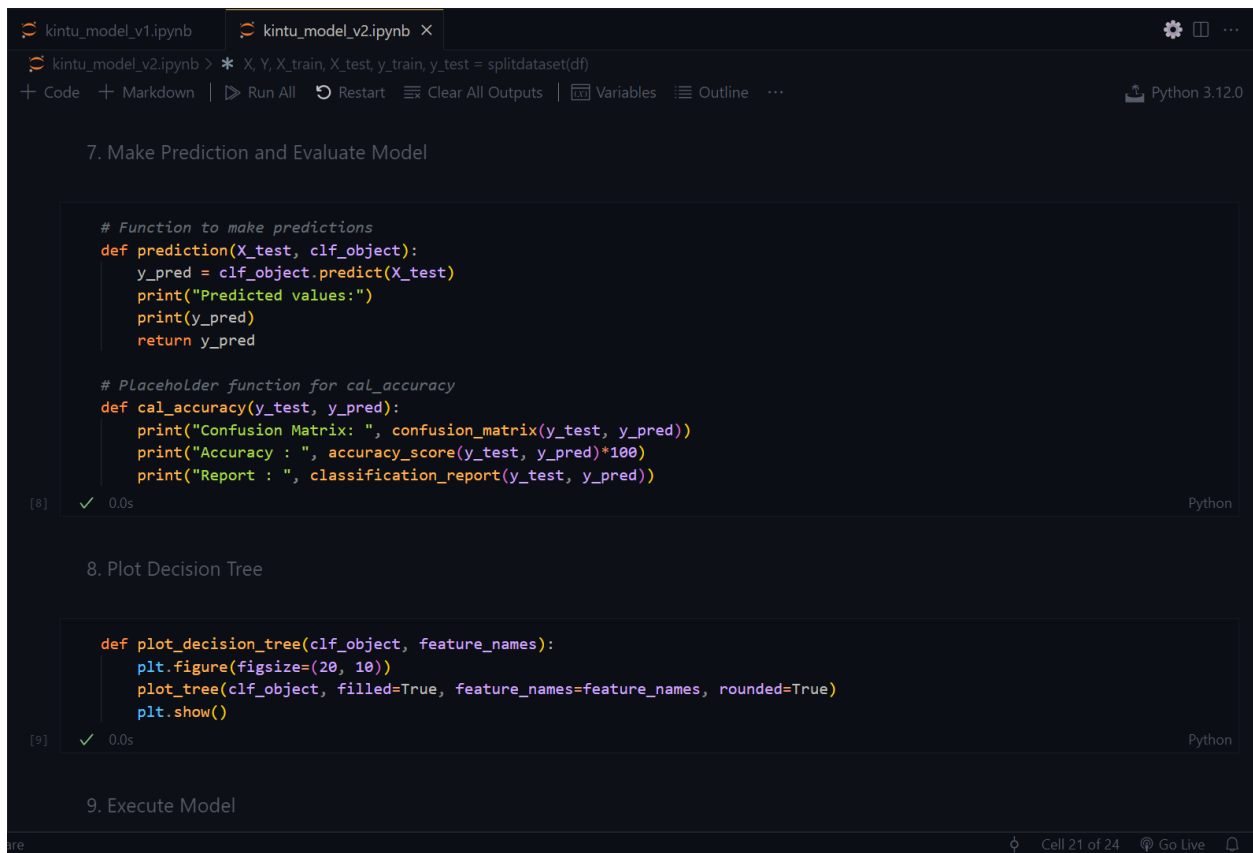
```
def train_using_entropy(X_train, y_train):  
    clf_entropy = DecisionTreeClassifier(criterion="entropy", random_state=100, max_depth=3, min_samples_leaf=5)  
    clf_entropy.fit(X_train, y_train)  
    return clf_entropy
```

Cell [7] is executed successfully (0.0s).

6. Train Model using Entropy

The bottom of the notebook shows 'Cell 21 of 24' and a 'Go Live' button.

The model can then be used to make predictions and have its decision tree plotted.



The image shows a Jupyter Notebook interface with two tabs: 'kintu_model_v1.ipynb' and 'kintu_model_v2.ipynb'. The active tab is 'kintu_model_v2.ipynb'. The notebook has a dark theme. At the top, there's a toolbar with icons for code, markdown, running, restarting, clearing outputs, variables, and outline. Below the toolbar, the notebook content is divided into sections. Section 7, '7. Make Prediction and Evaluate Model', contains a code cell [8] with the following Python code:

```
# Function to make predictions
def prediction(X_test, clf_object):
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred

# Placeholder function for cal_accuracy
def cal_accuracy(y_test, y_pred):
    print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
    print("Accuracy : ", accuracy_score(y_test, y_pred)*100)
    print("Report : ", classification_report(y_test, y_pred))
```

Section 8, '8. Plot Decision Tree', contains a code cell [9] with the following Python code:

```
def plot_decision_tree(clf_object, feature_names):
    plt.figure(figsize=(20, 10))
    plot_tree(clf_object, filled=True, feature_names=feature_names, rounded=True)
    plt.show()
```

Section 9, '9. Execute Model', is currently empty. The bottom of the notebook shows a status bar with 'Cell 21 of 24' and a 'Go Live' button.

```
kintu_model_v1.ipynb kintu_model_v2.ipynb
kintu_model_v2.ipynb > * X, Y, X_train, X_test, y_train, y_test = splitdataset(df)
+ Code + Markdown | ▶ Run All ⏮ Restart ⏹ Clear All Outputs | 📄 Variables 📄 Outline ... Python 3.12.0

# Placeholder function for cal_accuracy
def cal_accuracy(y_test, y_pred):
    print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
    print("Accuracy : ", accuracy_score(y_test, y_pred)*100)
    print("Report : ", classification_report(y_test, y_pred))

[20] ✓ 0.0s Python

8. Plot Decision Tree

def plot_decision_tree(clf_object, feature_names):
    plt.figure(figsize=(20, 10))
    plot_tree(clf_object, filled=True, feature_names=feature_names, rounded=True)
    plt.show()

[21] ✓ 0.0s Python

9. Execute Model

X, Y, X_train, X_test, y_train, y_test = splitdataset(df)

clf_gini = train_using_gini(X_train, y_train)
clf_entropy = train_using_entropy(X_train, y_train)

# Visualizing the Decision Trees
plot_decision_tree(clf_gini, X_train.columns)
plot_decision_tree(clf_entropy, X_train.columns)

[22] ✓ 0.9s Python

Cell 21 of 24 Go Live
```



```

kintu_model_v1.ipynb kintu_model_v2.ipynb
kintu_model_v2.ipynb > * X, Y, X_train, X_test, y_train, y_test = splitdataset(df)
+ Code + Markdown | ▶ Run All ⏮ Restart ⏭ Clear All Outputs | 📄 Variables 📖 Outline ... Python 3.12.0

#Evaluating Entropy model
print("Results Using Entropy:")
y_pred_entropy = prediction(X_test, clf_entropy)
cal_accuracy(y_test, y_pred_entropy)

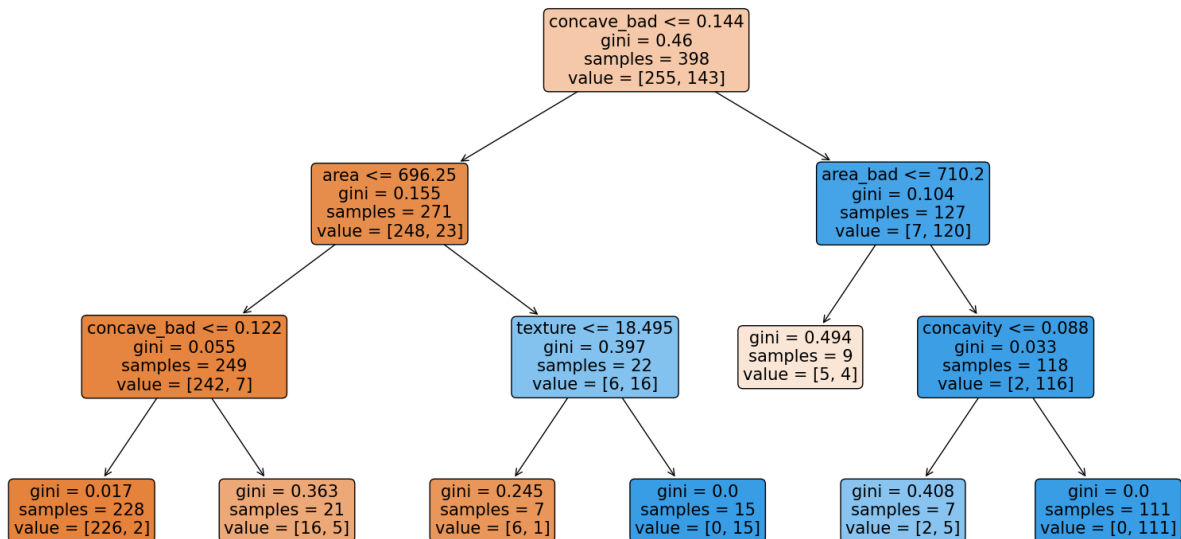
[24] ✓ 0.0s Python

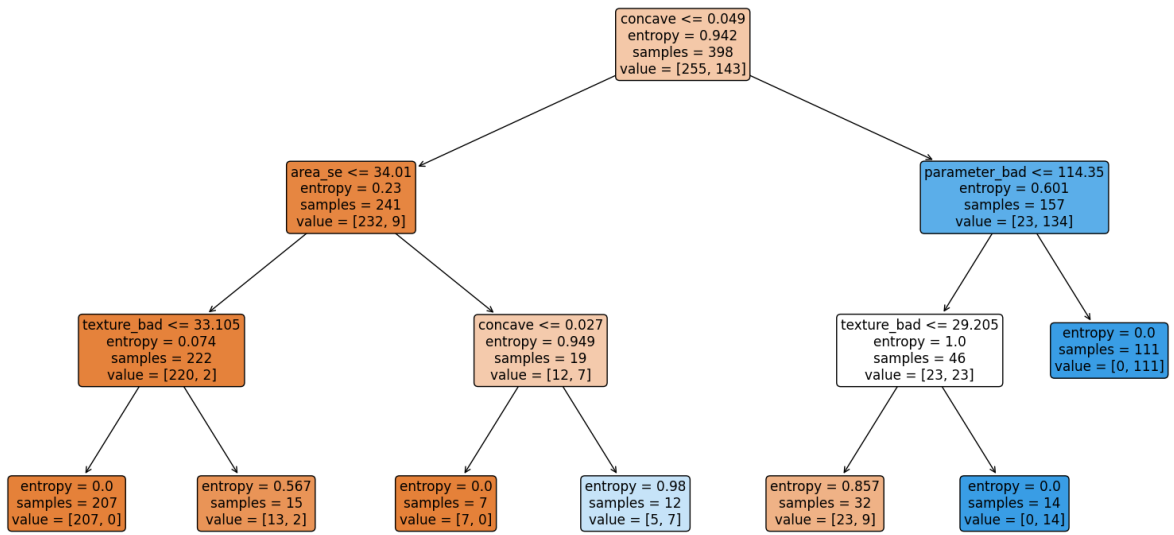
... Results Using Entropy:
Predicted values:
['M' 'M' 'M' 'B' 'B' 'B' 'M' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B'
'B' 'B' 'M' 'B' 'M' 'M' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B'
'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M'
'B' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'M' 'M' 'B' 'M' 'B'
'B' 'B' 'B' 'M' 'B' 'M' 'M' 'B' 'M' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B'
'B' 'B' 'M' 'B' 'M' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'B' 'B'
'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B'
'B' 'B' 'M' 'B' 'M' 'B' 'M' 'B' 'M' 'M' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'B'
'B' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'B']
Confusion Matrix: [[99  3]
 [11 58]]
Accuracy : 91.81286549707602
Report :
      precision    recall  f1-score   support

      B       0.90       0.97       0.93       102
      M       0.95       0.84       0.89        69

   accuracy       0.92       0.92       0.92       171
  macro avg       0.93       0.91       0.91       171
 weighted avg       0.92       0.92       0.92       171

```





Step 3, Step 4, and Step 5.

In this step, I perform the following activities:

- i. Uploading the first model to the main branch of the repository.
- ii. Creating a new branch on the repository.
- iii. Uploading the second model to the newly created branch on the repository.

```
PowerShell
PowerShell 7.4.1
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git clone https://github.com/MrKintu/test-lab2.git
Cloning into 'test-lab2'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git add .\kintu_model_v1.ipynb
warning: in the working copy of 'kintu_model_v1.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git commit -m "First Model"
[main 816e49d] First Model
1 file changed, 788 insertions(+)
create mode 100644 kintu_model_v1.ipynb
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 119.97 KiB | 15.00 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MrKintu/test-lab2.git
583b3e1..816e49d main -> main
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git checkout -b second
Switched to a new branch 'second'
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git add .\kintu_model_v2.ipynb
warning: in the working copy of 'kintu_model_v2.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git commit -m "Second Model"
[second c3a3ff8] Second Model
1 file changed, 898 insertions(+)
create mode 100644 kintu_model_v2.ipynb
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git push
fatal: The current branch second has no upstream branch.
To push the current branch and set the remote as upstream, use
```

```
PowerShell
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git add .\kintu_model_v1.ipynb
warning: in the working copy of 'kintu_model_v1.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git commit -m "First Model"
[main 816e49d] First Model
 1 file changed, 788 insertions(+)
 create mode 100644 kintu_model_v1.ipynb
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 119.97 KiB | 15.00 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MrKintu/test-lab2.git
 583b3e1..816e49d main -> main
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git checkout -b second
Switched to a new branch 'second'
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git add .\kintu_model_v2.ipynb
warning: in the working copy of 'kintu_model_v2.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git commit -m "Second Model"
[second c3a3ff8] Second Model
 1 file changed, 898 insertions(+)
 create mode 100644 kintu_model_v2.ipynb
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git push
fatal: The current branch second has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin second

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2>
```

```
PowerShell
warning: in the working copy of 'kintu_model_v2.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git commit -m "Second Model"
[second c3a3ff8] Second Model
 1 file changed, 898 insertions(+)
 create mode 100644 kintu_model_v2.ipynb
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git push
fatal: The current branch second has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin second

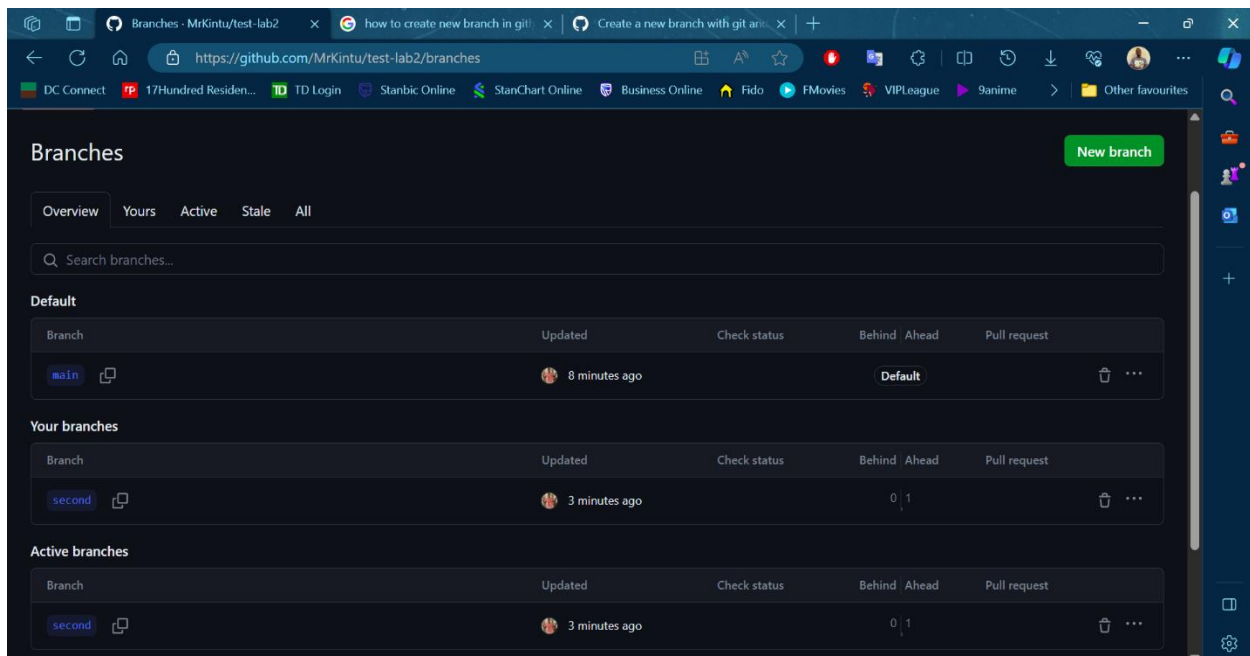
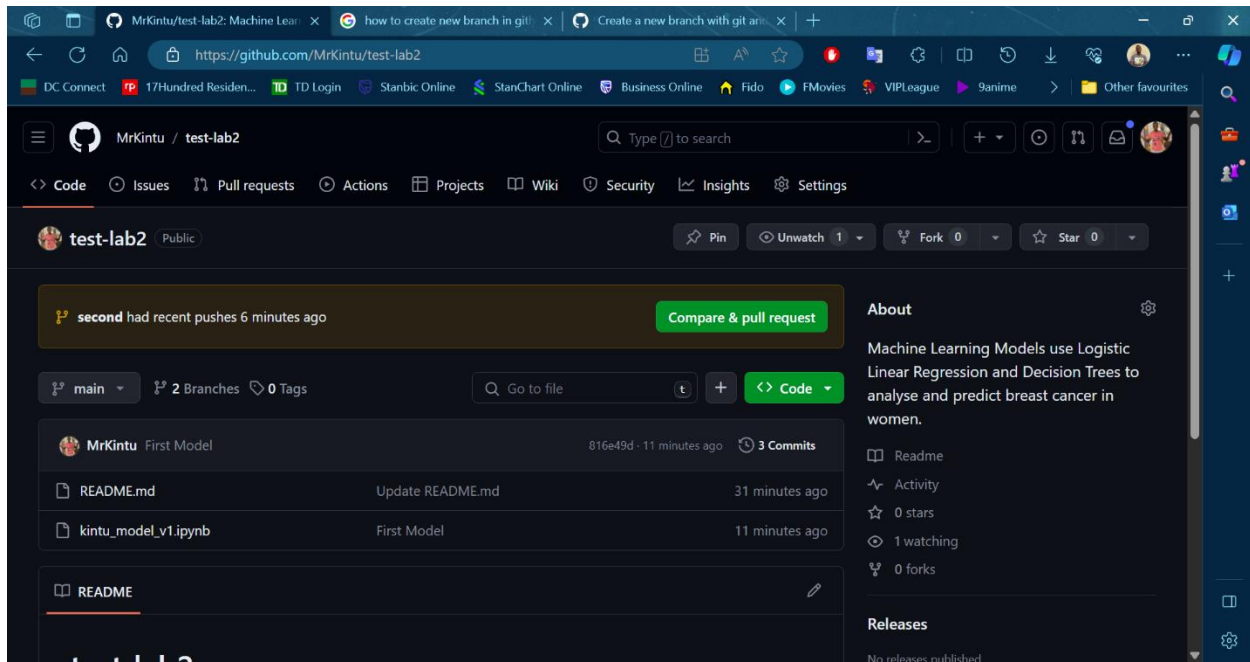
To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git checkout second
Already on 'second'
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git push origin/second
fatal: 'origin/second' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git push origin second
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 373.17 KiB | 12.04 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'second' on GitHub by visiting:
remote:   https://github.com/MrKintu/test-lab2/pull/new/second
remote:
To https://github.com/MrKintu/test-lab2.git
 * [new branch]      second -> second
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2>
```

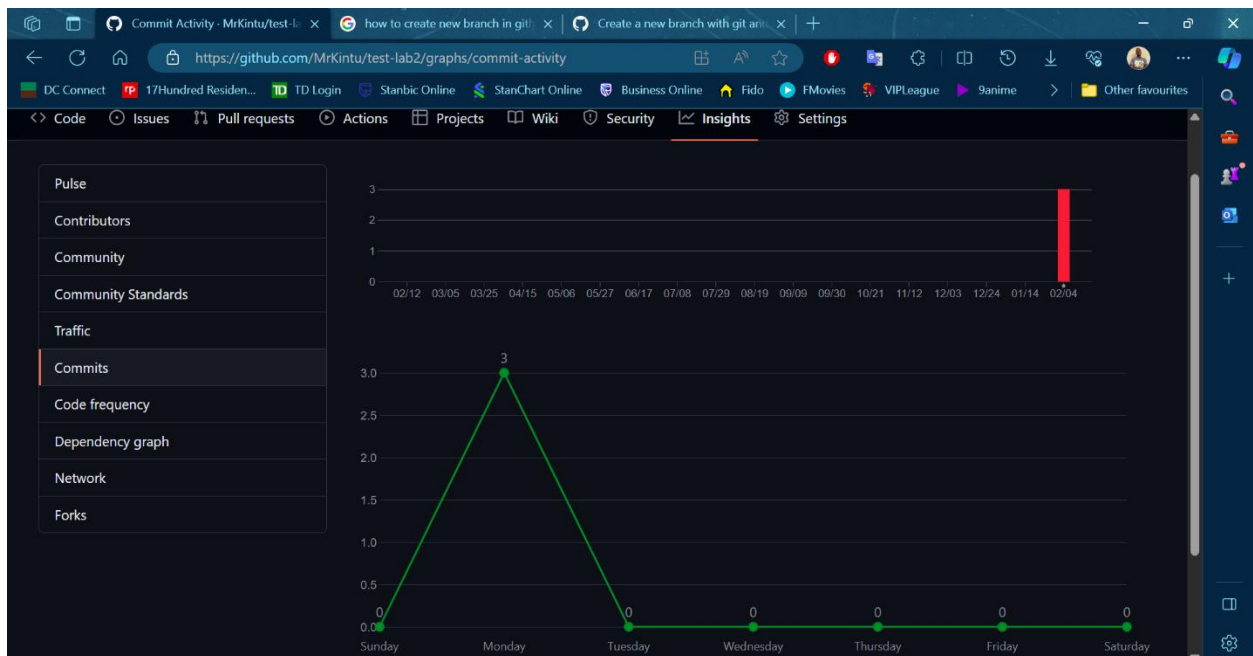
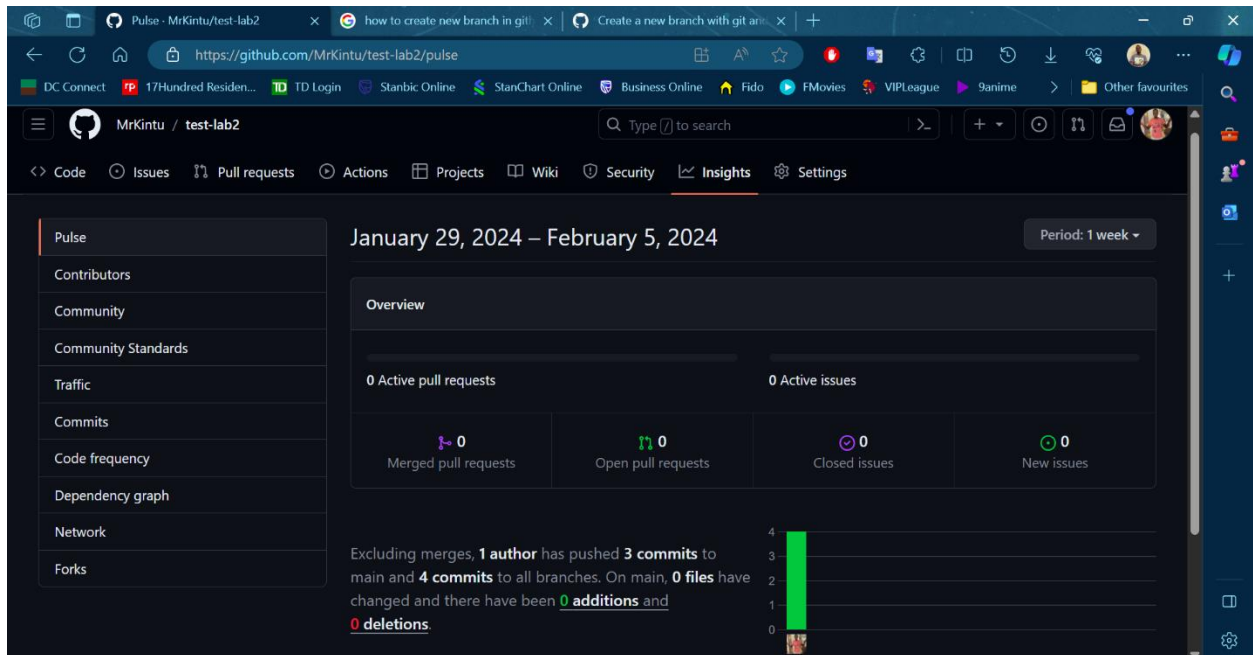
Step 6.

This is what the GitHub repository looks like after the commits.



Step 7.

This is the repository activity log.



Step 8.

This is the README file attached to this project.

