LAB 2

AIDI 2004

Declan Trevor Kintu
100944330 PROFESSOR MOHAMMAD AL-TAWALBEH

Table of Contents

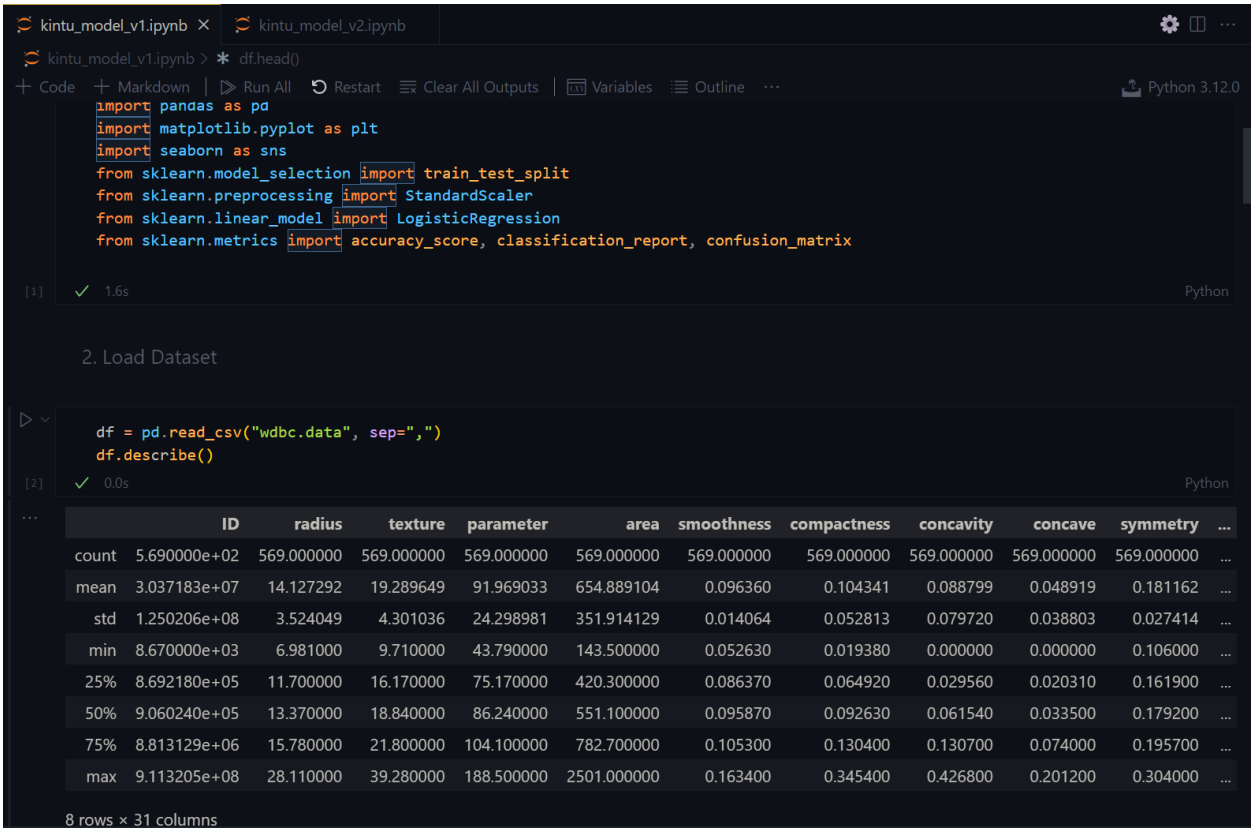
Step 9.

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).

Step 2 and Step 3.

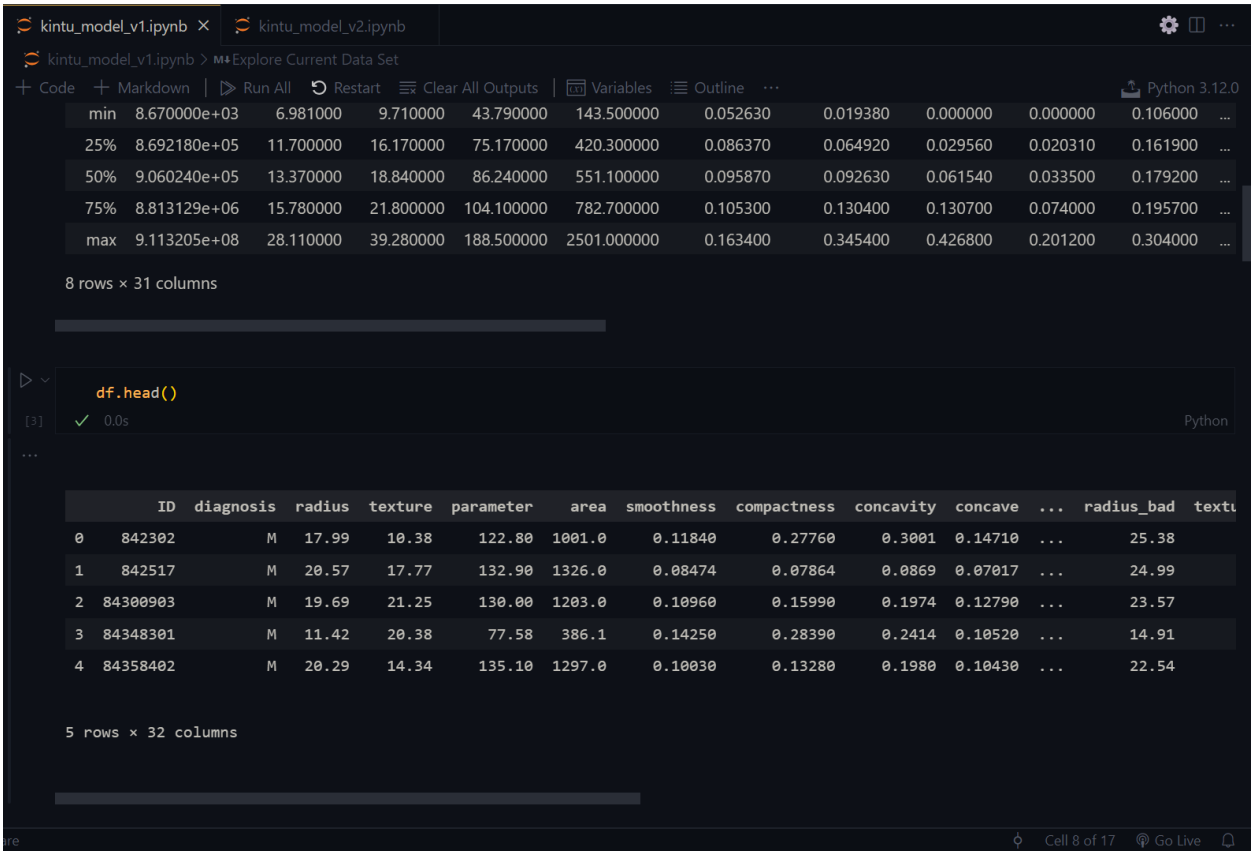
First Model:

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.

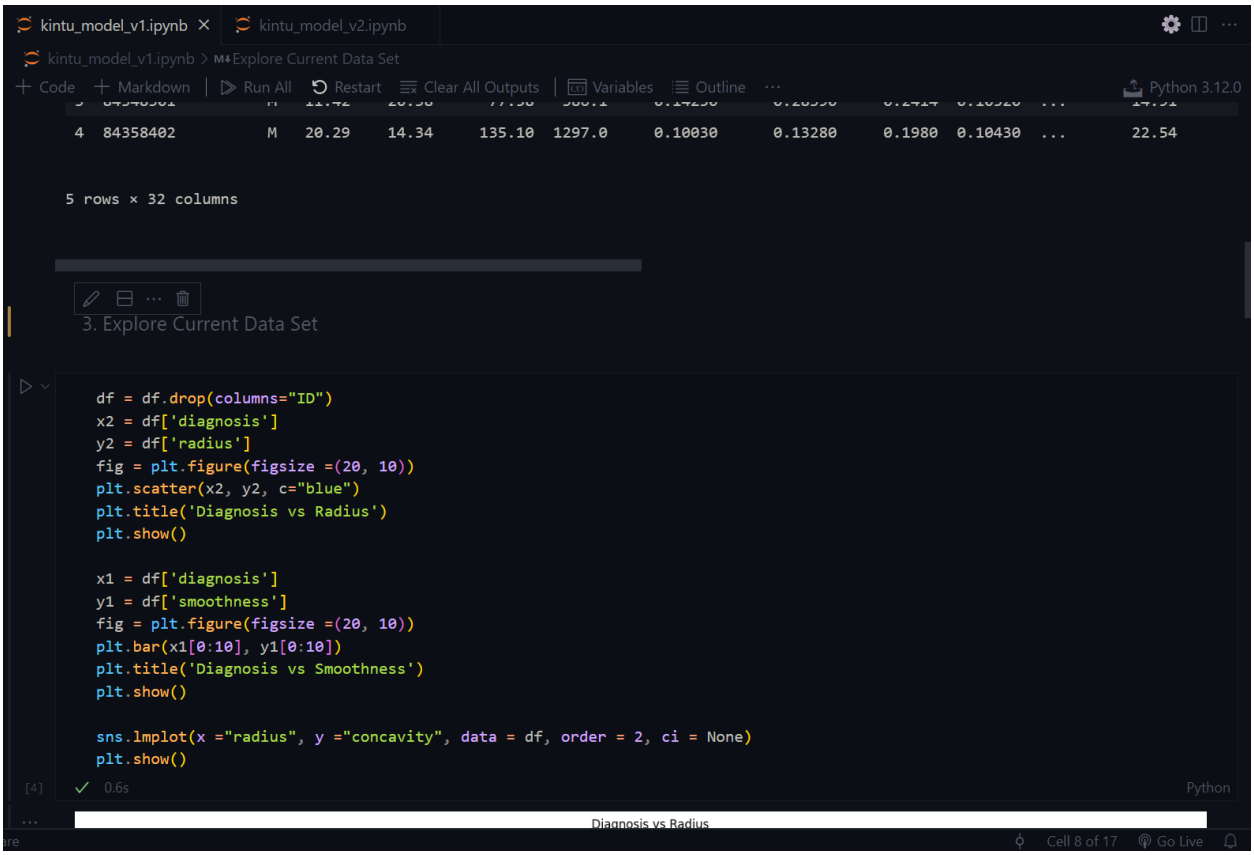


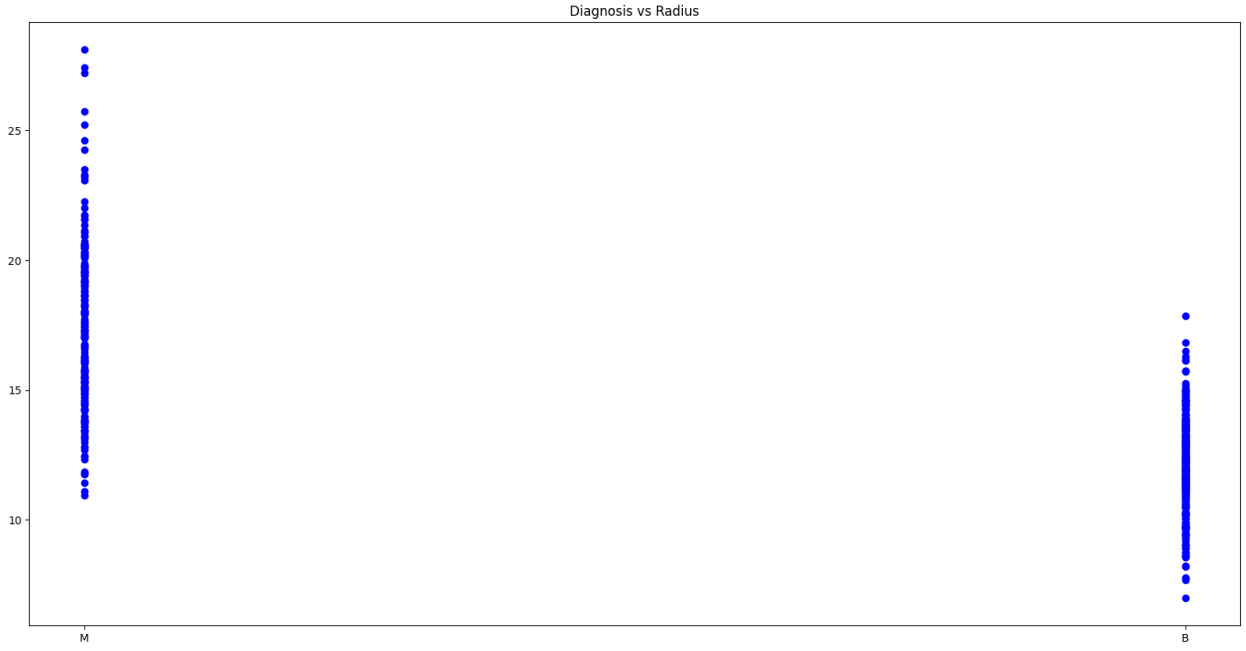
	ID	radius	texture	parameter	area	smoothness	compactness	concavity	concave	symmetry	...
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	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	...

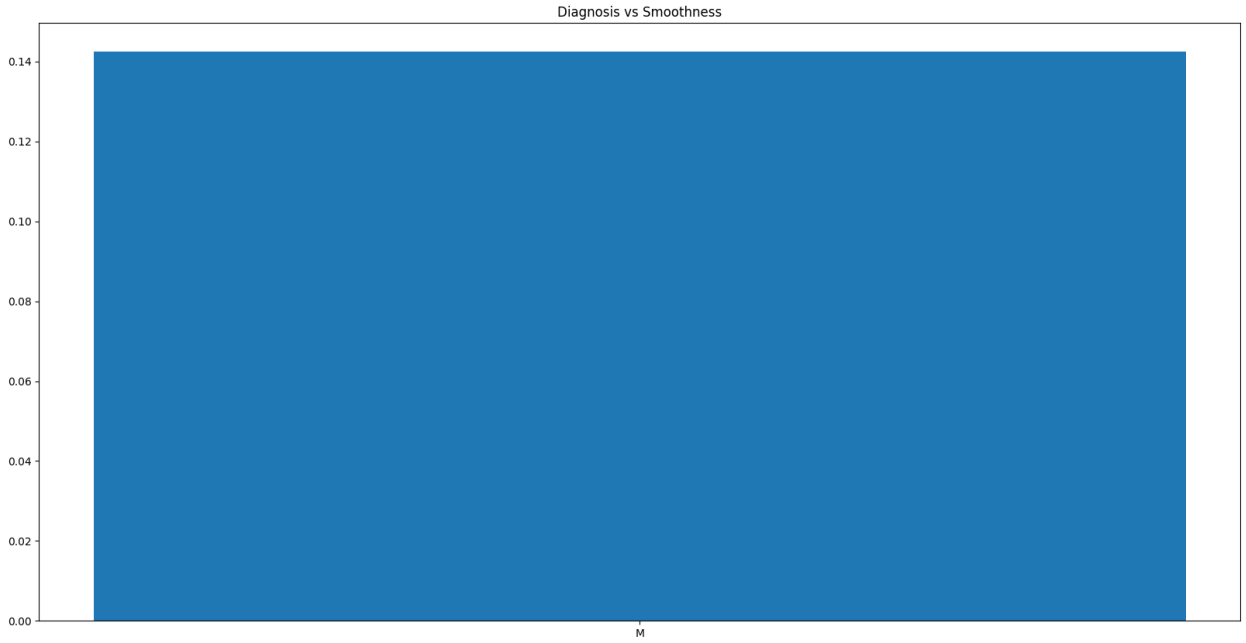
8 rows × 31 columns

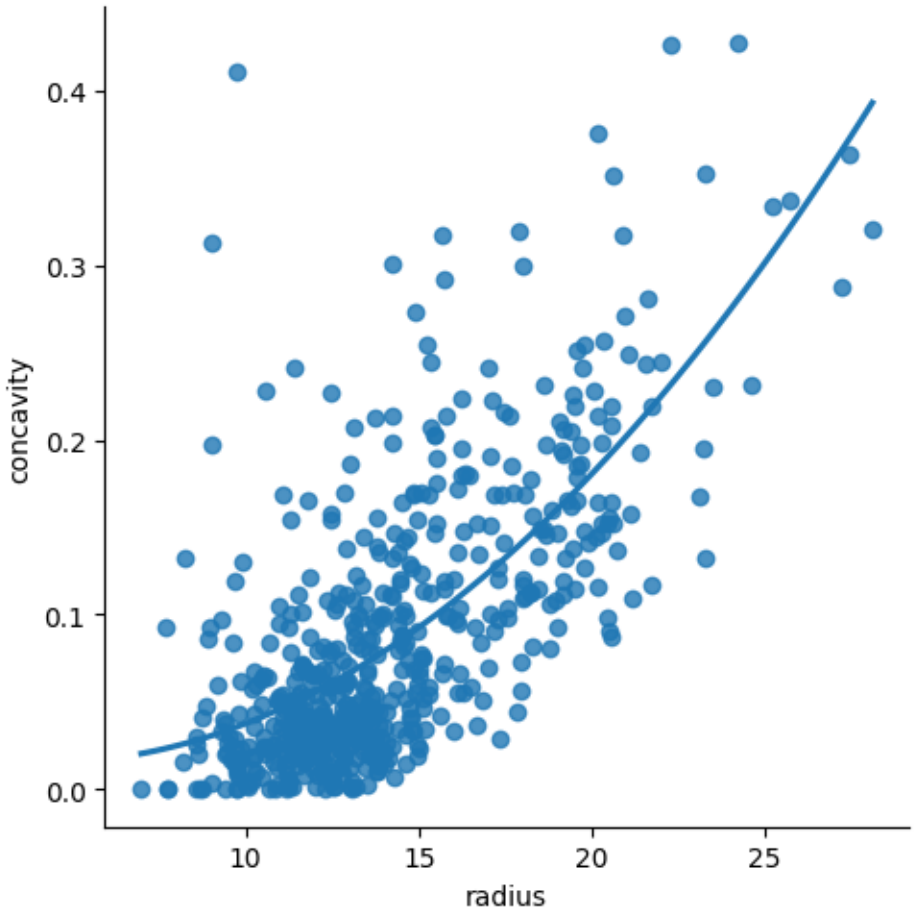

kintu_model_v1.ipynb
kintu_model_v2.ipynb
kintu_model_v1.ipynb > Explore Current Data Set
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
Python 3.12.0
min 8.670000e+03 6.981000 9.710000 43.790000 143.500000 0.052630 0.019380 0.000000 0.000000 0.106000 ...
25% 8.692180e+05 11.700000 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
Cell 8 of 17
Go Live

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

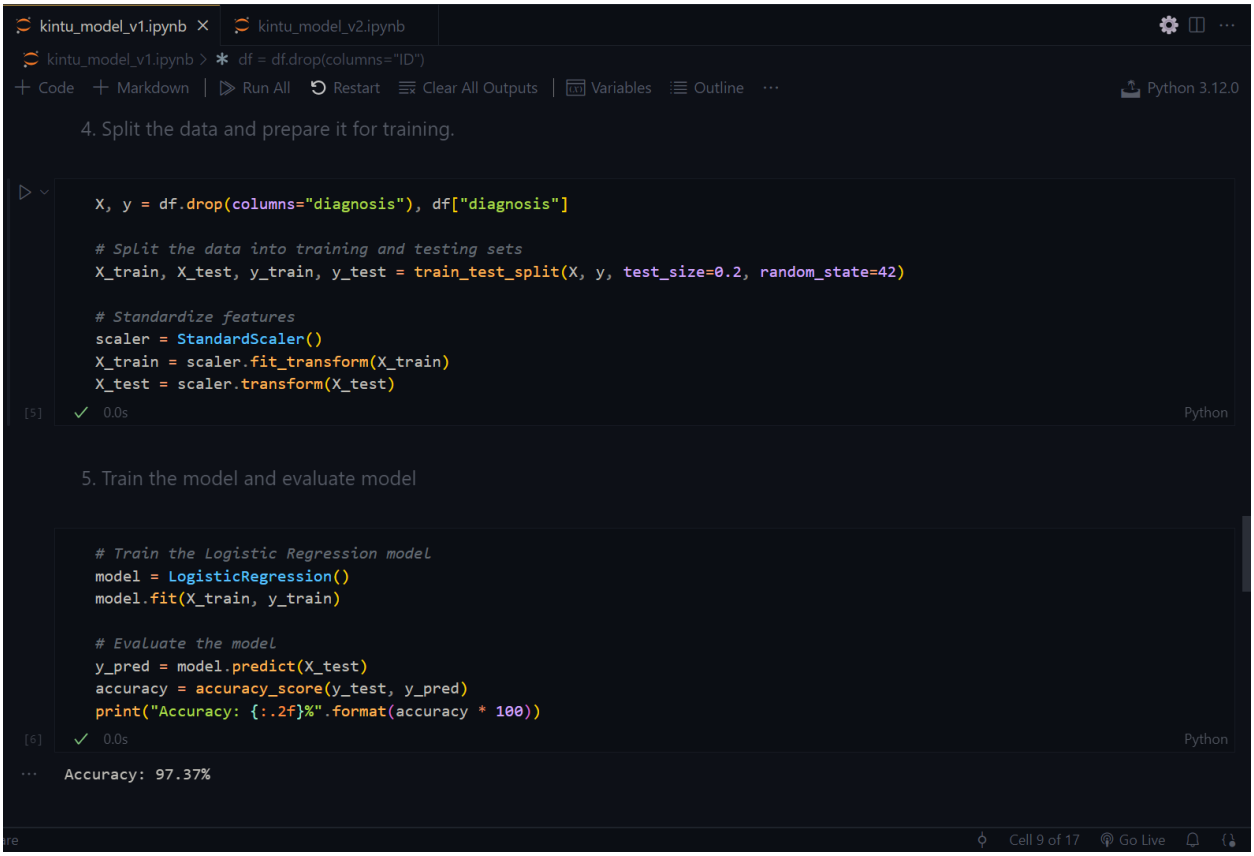



Diagnosis vs Radius
25
20
15
10
M
B

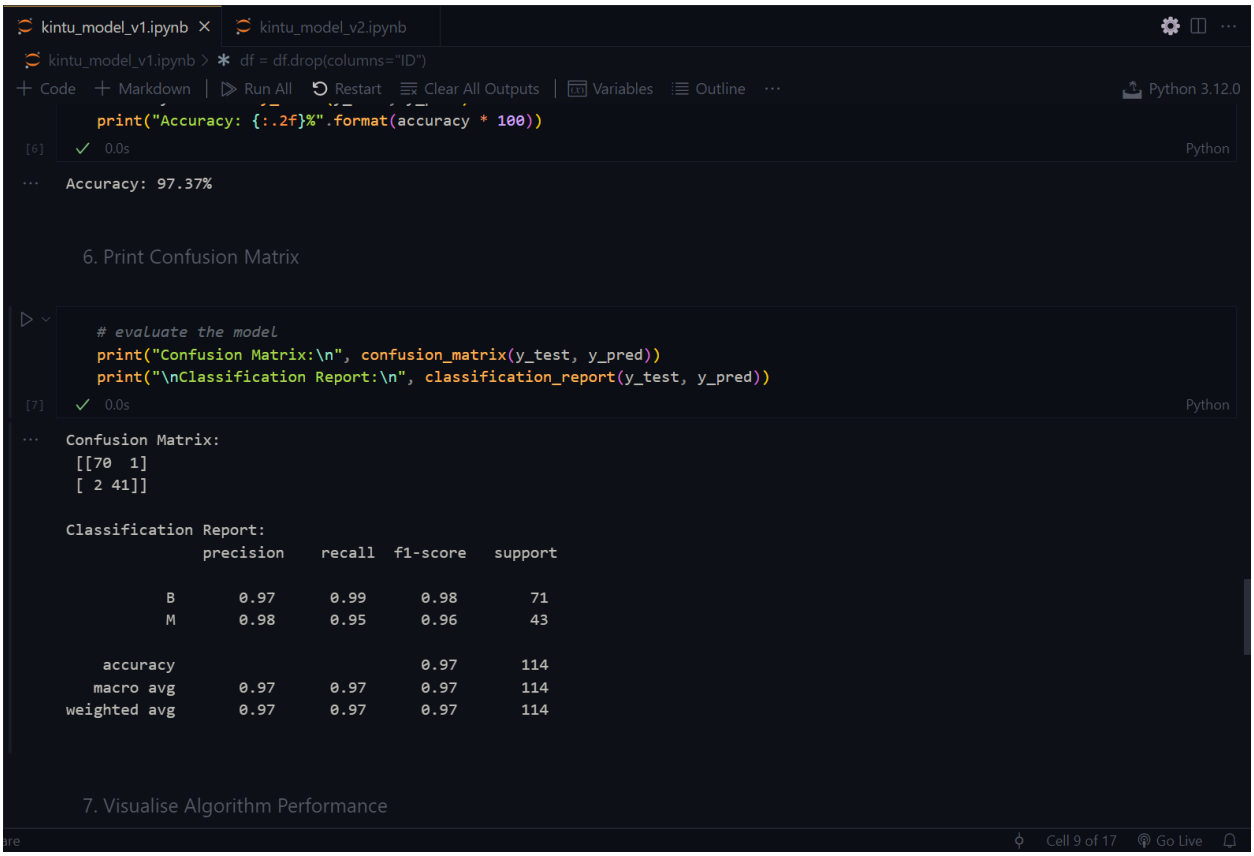

Diagnosis vs Smoothness
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
M


0.4
0.3
0.2
0.1
0.0
concavity
10
15
20
25
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.



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



kintu_model_v1.ipynb | 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 Python

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

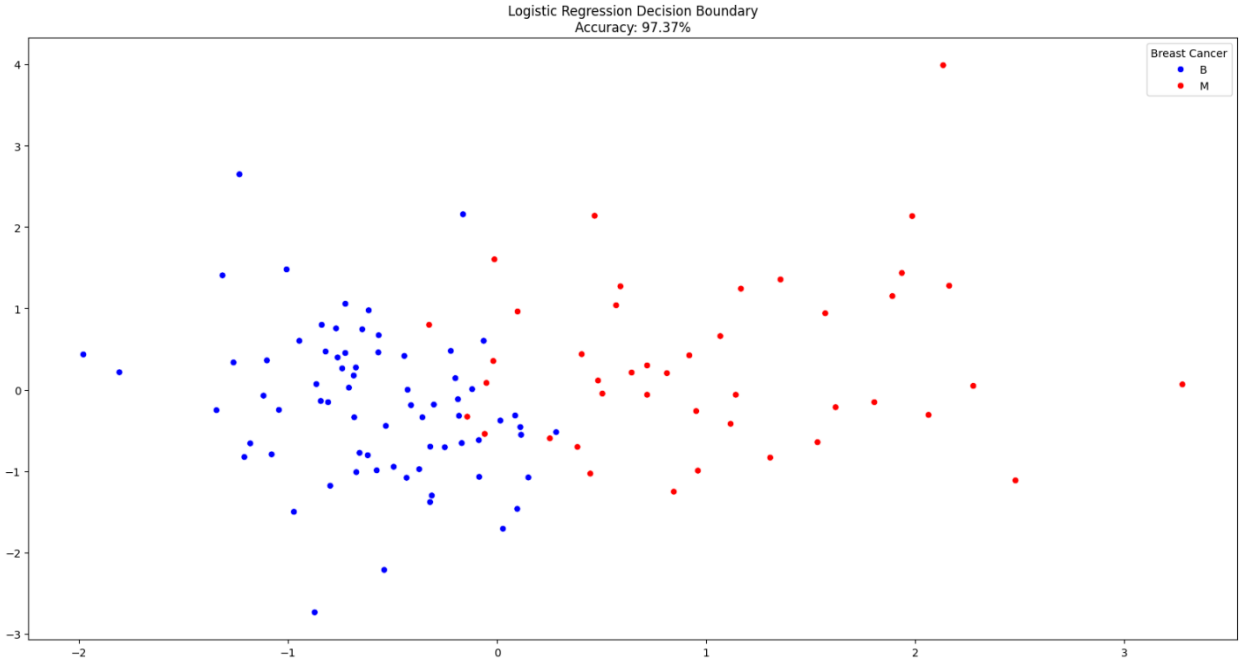
+ Code + Markdown

7. Visualise Algorithm Performance

```python
# 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()
```

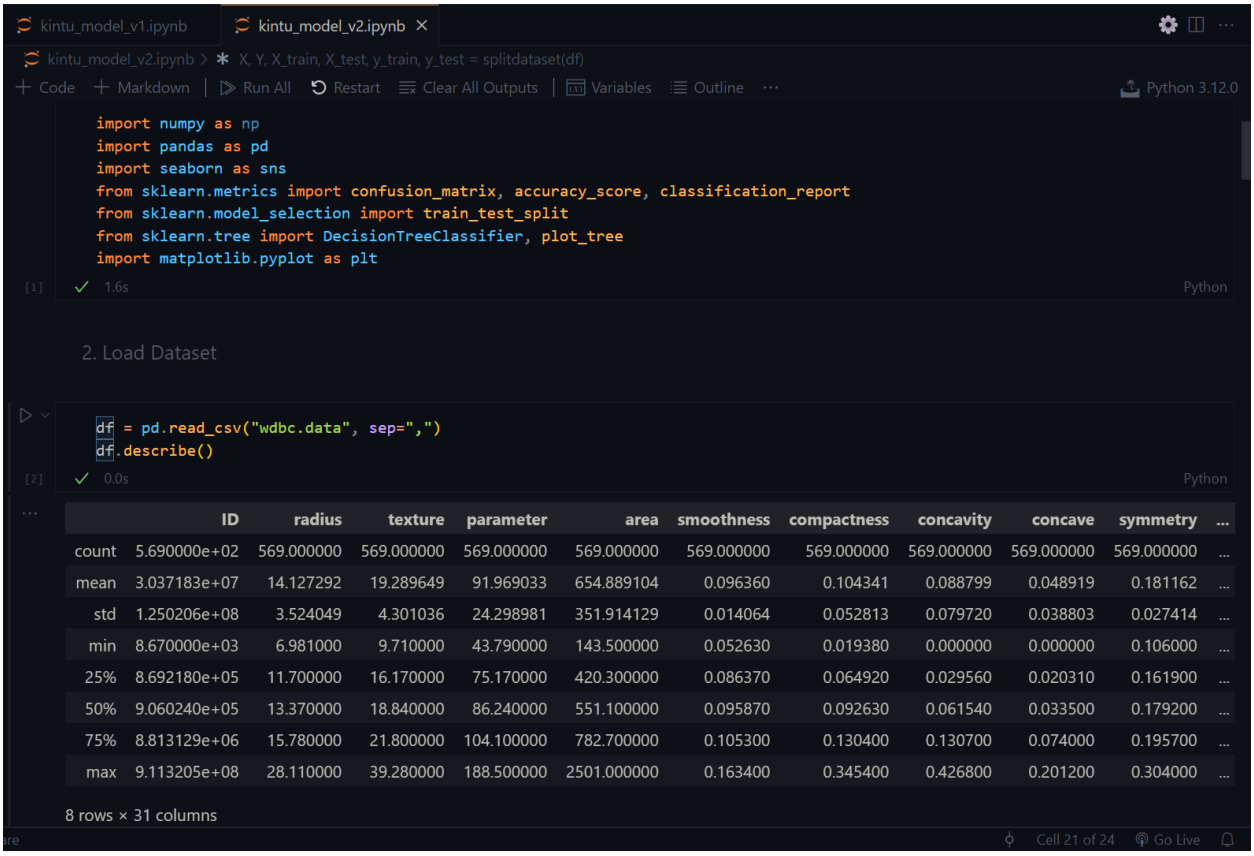
[8] ✓ 0.2s Python

re Cell 9 of 17 Go Live

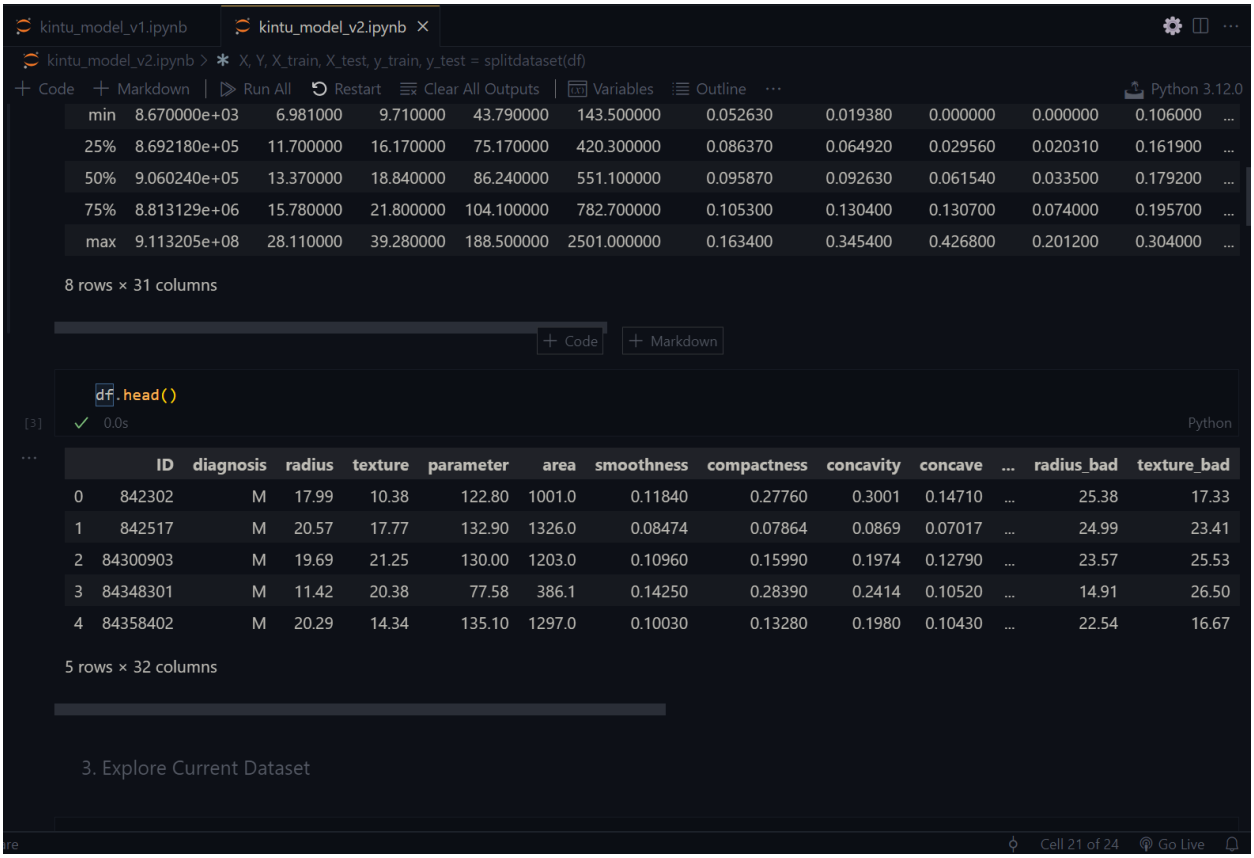


Second Model:

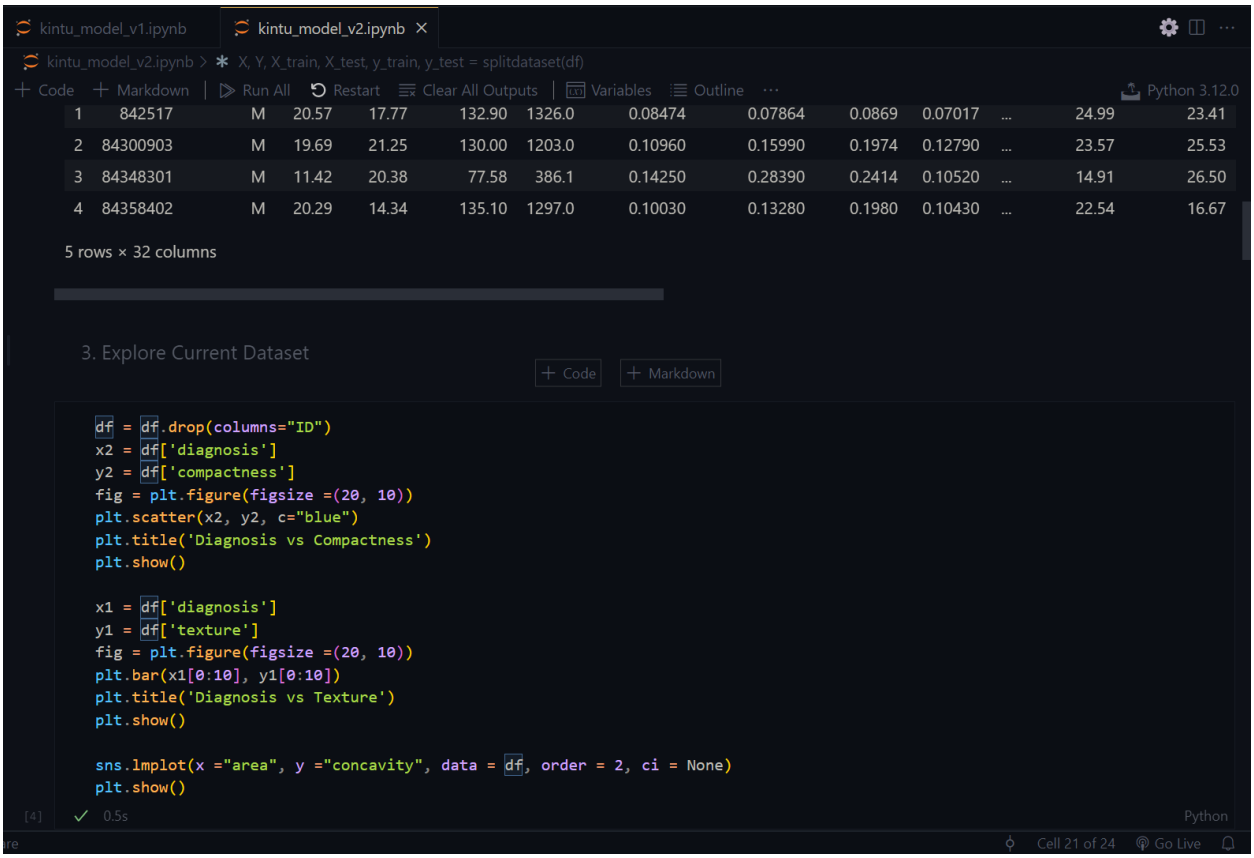
This machine learning algorithm is implemented using Decision Tree Classification. The first step is to load the necessary libraries and display the current dataset.

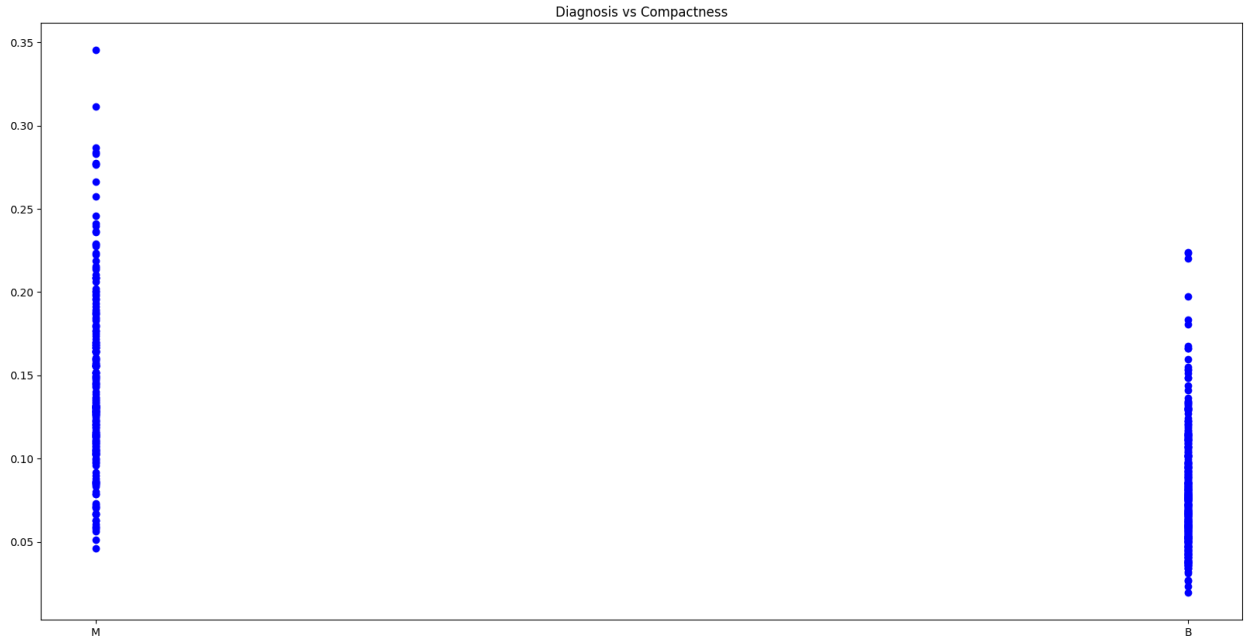


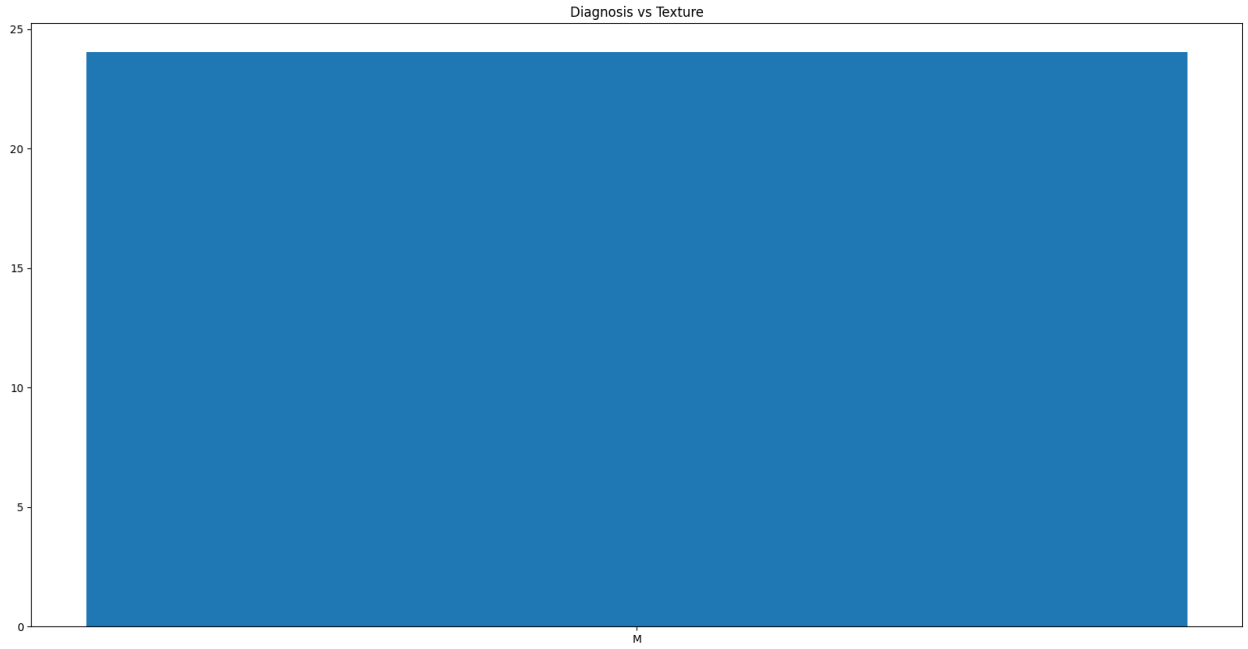
	ID	radius	texture	parameter	area	smoothness	compactness	concavity	concave	symmetry	...
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	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	...

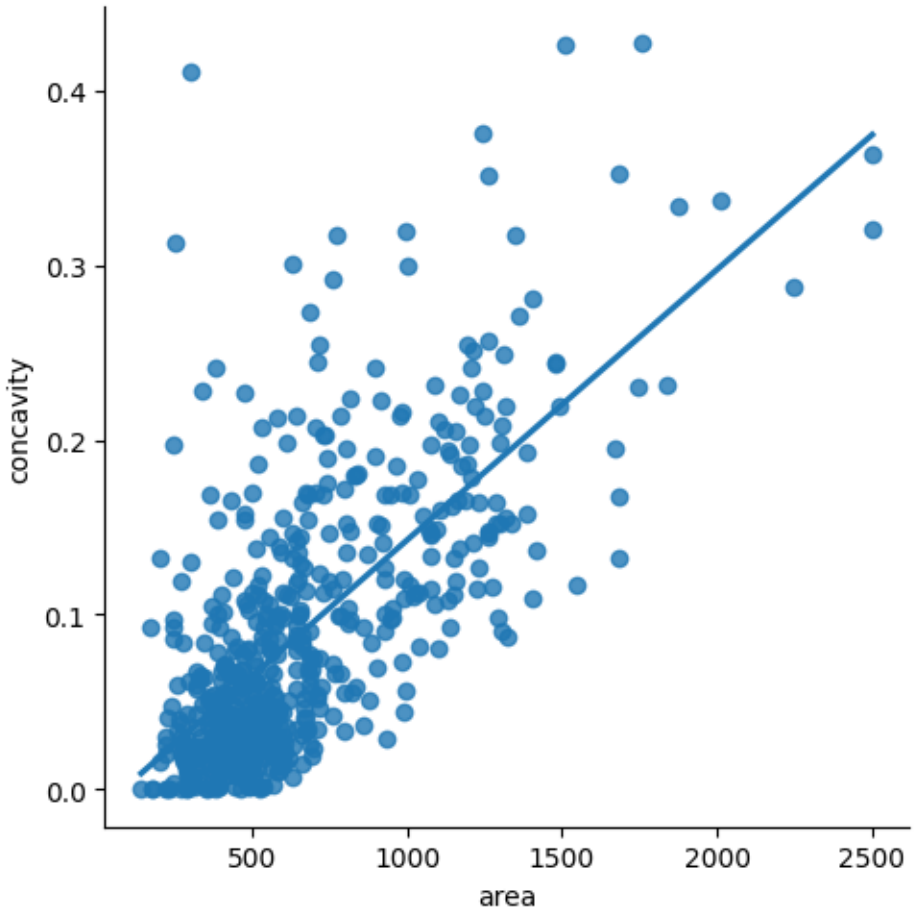

kintu_model_v1.ipynb
kintu_model_v2.ipynb
kintu_model_v2.ipynb > X, Y, X_train, X_test, y_train, y_test = splitdataset(df)
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
Python 3.12.0
min 8.670000e+03 6.981000 9.710000 43.790000 143.500000 0.052630 0.019380 0.000000 0.000000 0.106000 ...
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
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 × 32 columns
3. Explore Current Dataset
Cell 21 of 24
Go Live

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



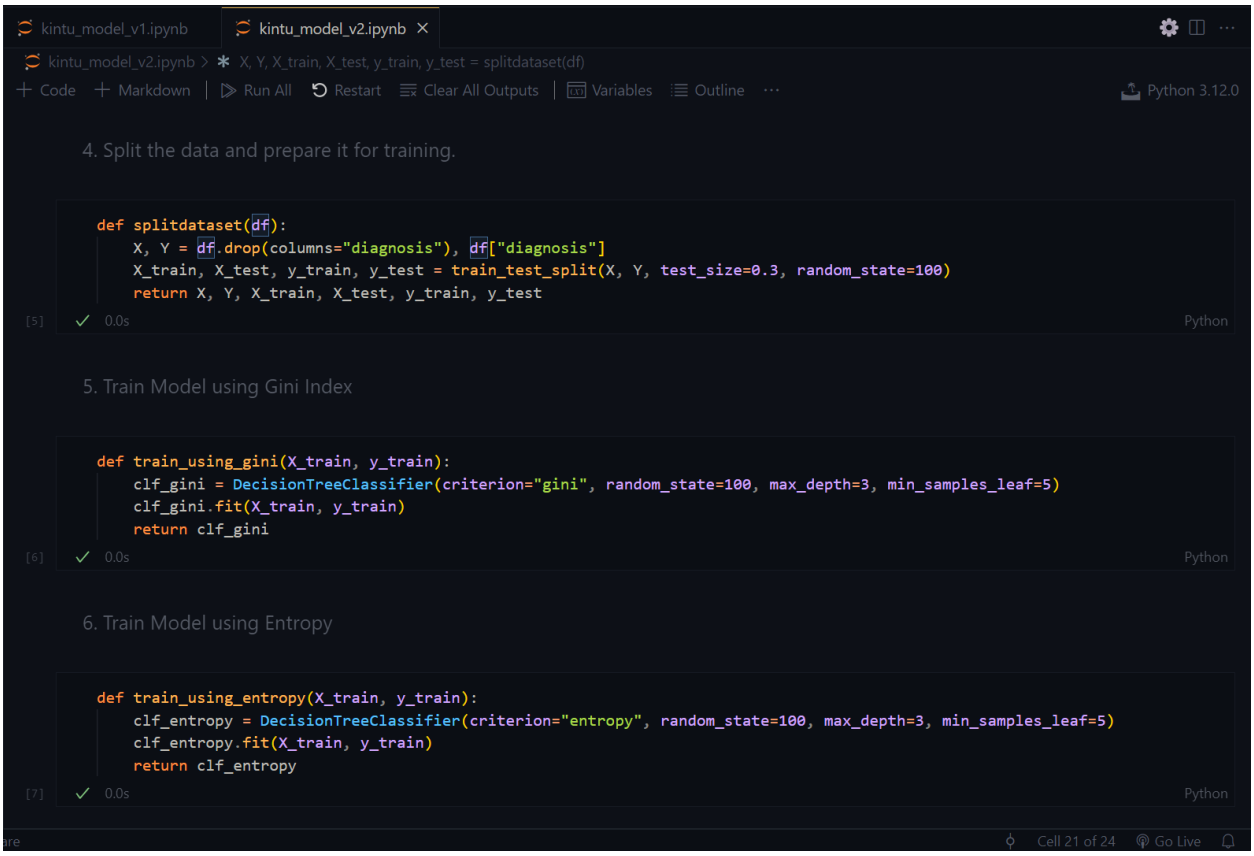

Diagnosis vs Compactness
0.35
0.30
0.25
0.20
0.15
0.10
0.05
M
B


Diagnosis vs Texture
25
20
15
10
5
0
M

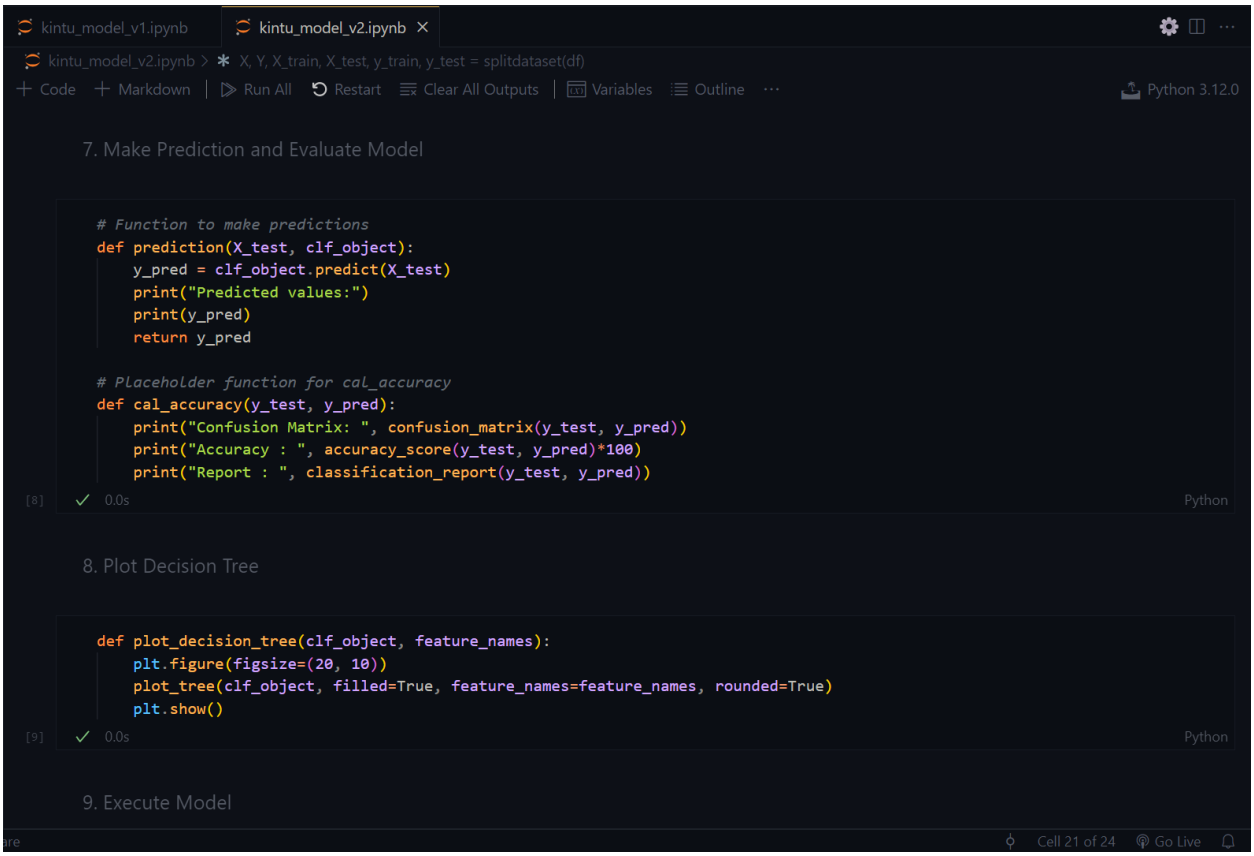

0.4
0.3
0.2
0.1
0.0
concavity
500
1000
1500
2000
2500
area

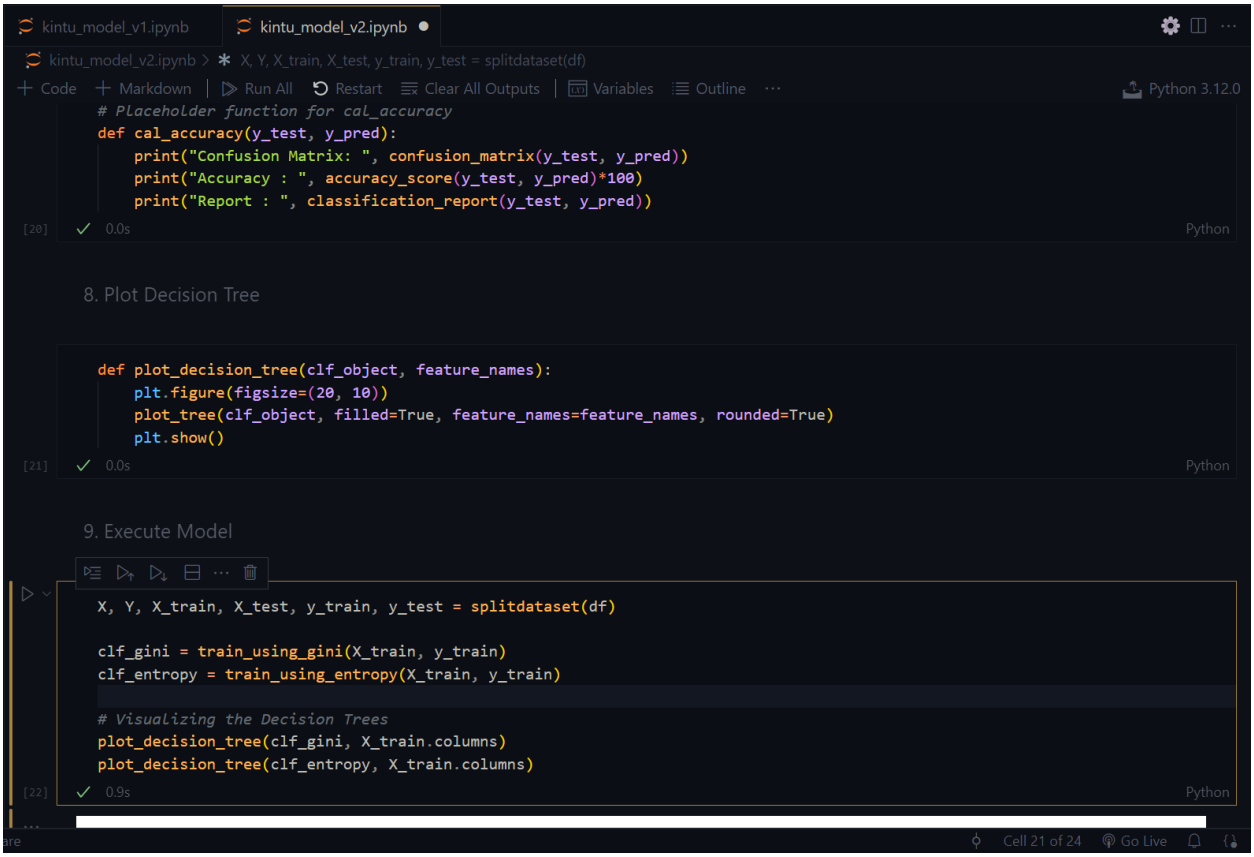
Next, the dataset is split into training and test data. This ML algorithm shall be trained using two variations (Gini Index and Entropy) so that we may be able to compare the performance of the two variations.

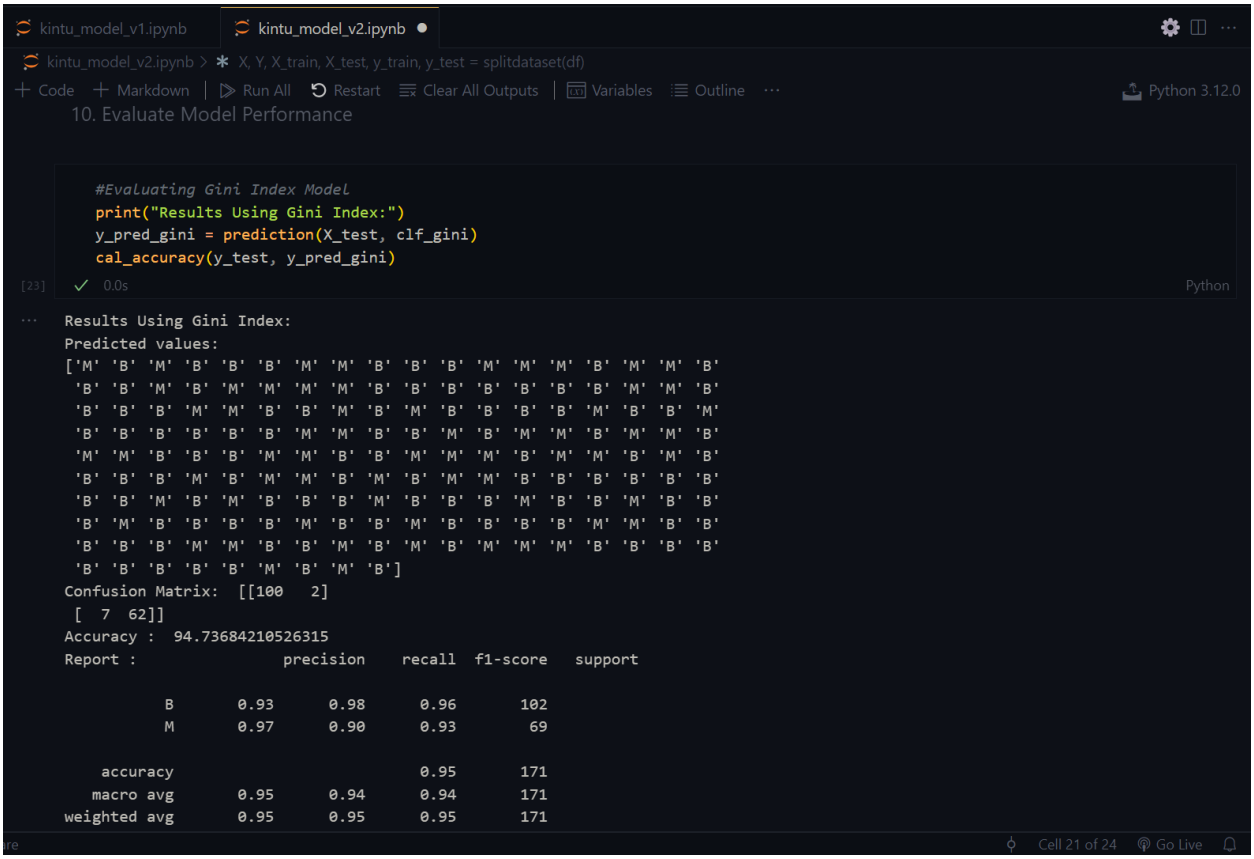
The source code that describes the training of the model using both variations is added.



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




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
10. Evaluate Model Performance
#Evaluating Gini Index Model
print("Results Using Gini Index:")
y_pred_gini = prediction(X_test, clf_gini)
cal_accuracy(y_test, y_pred_gini)
[23]
0.0s
Python
Results Using Gini Index:
Predicted values:
['M' 'B' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'M' 'M' 'M' 'B' 'M' 'M' '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' 'M' 'B' 'B' 'M'
'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'B' 'M' 'M' 'B' 'M' 'M' 'B'
'M' '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' 'M' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B'
'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'B' 'M' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'B'
'B' 'B' 'B' 'B' 'B' 'M' 'B' 'M' 'B']
Confusion Matrix: [[100 2]
[7 62]]
Accuracy : 94.73684210526315
Report : precision recall f1-score support
B 0.93 0.98 0.96 102
M 0.97 0.90 0.93 69
accuracy 0.95 171
macro avg 0.95 0.94 0.94 171
weighted avg 0.95 0.95 0.95 171
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)
```

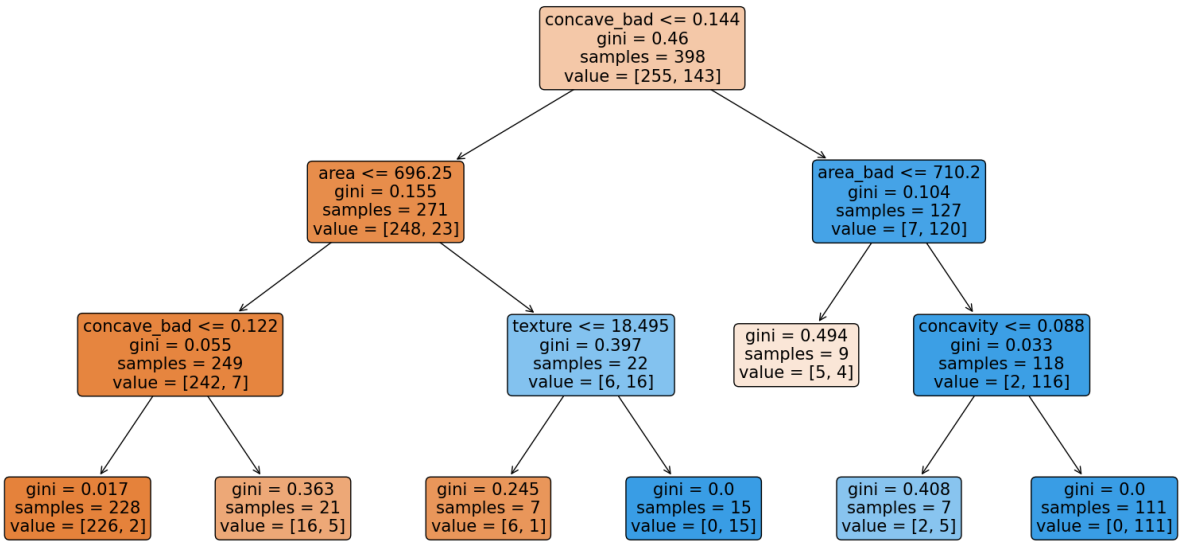
[24] ✓ 0.0s Python

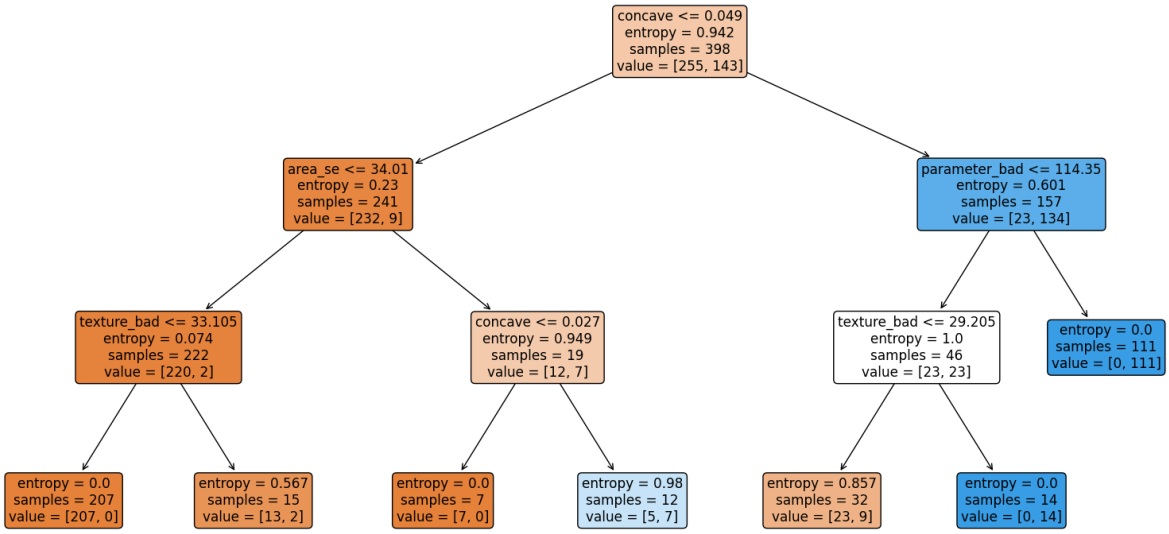
```
Results Using Entropy:
Predicted values:
['M' 'M' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'M' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'B'
 'B' 'B' 'M' 'B' 'M' 'M' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B'
 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M'
 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'B' 'M' 'M' 'B' 'M' 'M' 'B'
 '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       171
   macro avg       0.93      0.91      0.91       171
weighted avg       0.92      0.92      0.92       171
```

Cell 21 of 24 Go Live

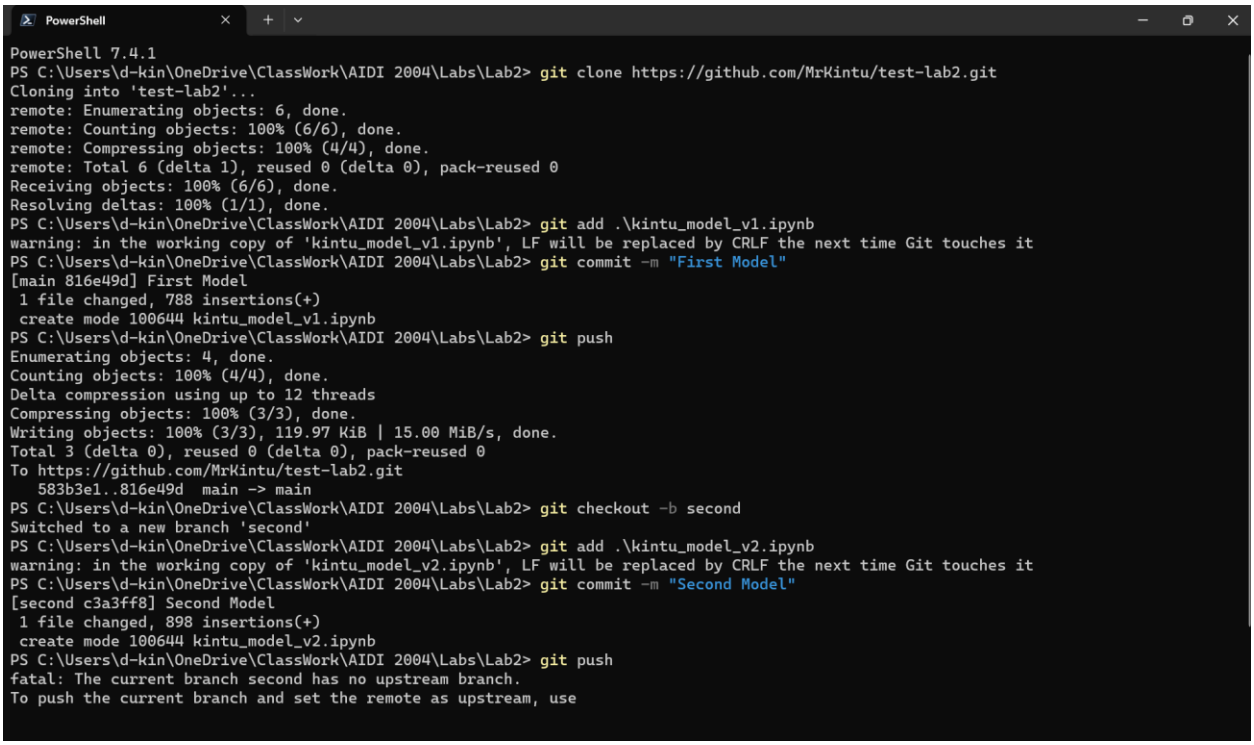


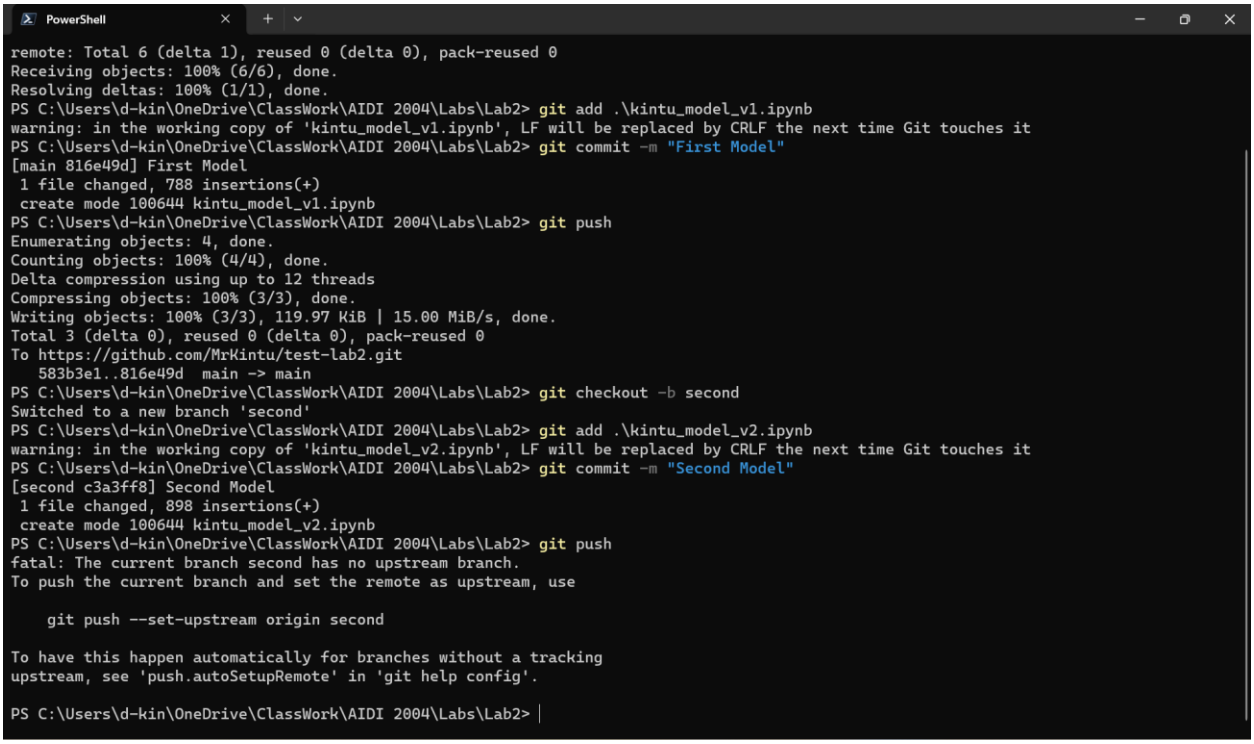

concave <= 0.049
entropy = 0.942
samples = 398
value = [255, 143]
area_se <= 34.01
entropy = 0.23
samples = 241
value = [232, 9]
parameter_bad <= 114.35
entropy = 0.601
samples = 157
value = [23, 134]
texture_bad <= 33.105
entropy = 0.074
samples = 222
value = [220, 2]
concave <= 0.027
entropy = 0.949
samples = 19
value = [12, 7]
texture_bad <= 29.205
entropy = 1.0
samples = 46
value = [23, 23]
entropy = 0.0
samples = 111
value = [0, 111]
entropy = 0.0
samples = 207
value = [207, 0]
entropy = 0.567
samples = 15
value = [13, 2]
entropy = 0.0
samples = 7
value = [7, 0]
entropy = 0.98
samples = 12
value = [5, 7]
entropy = 0.857
samples = 32
value = [23, 9]
entropy = 0.0
samples = 14
value = [0, 14]

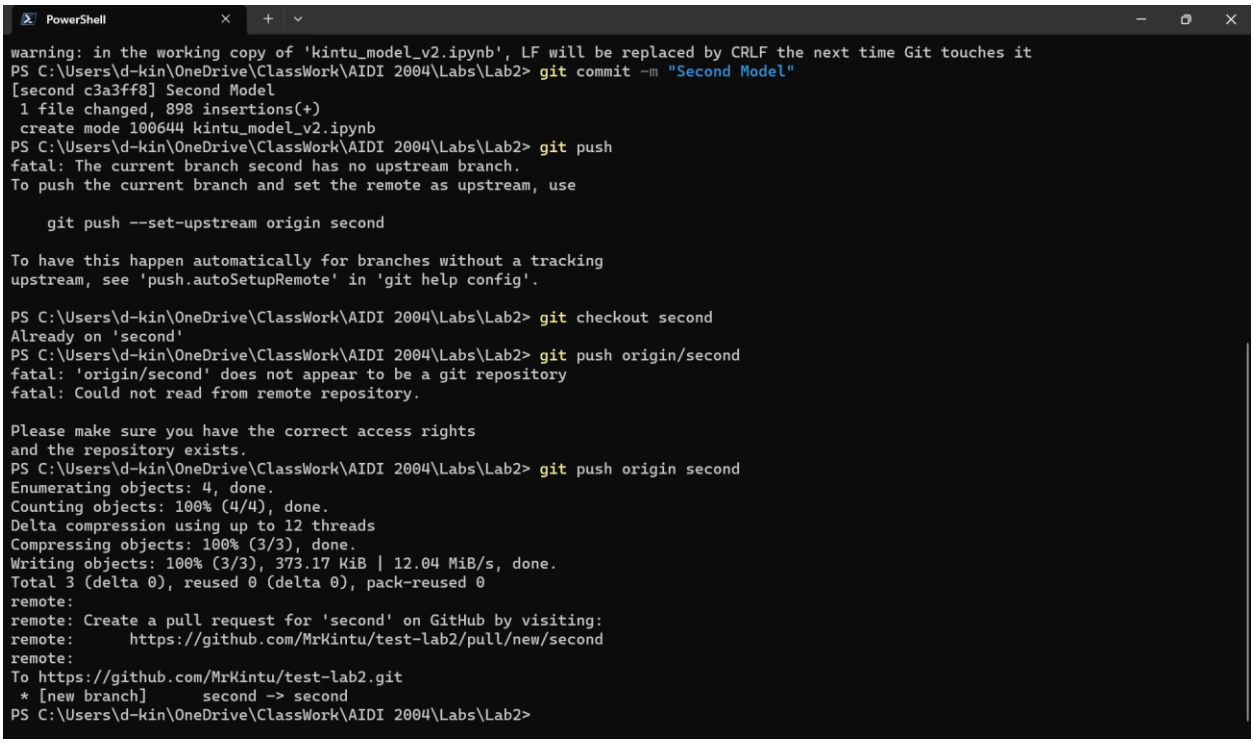
Step 3, Step 4, and Step 5.

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.

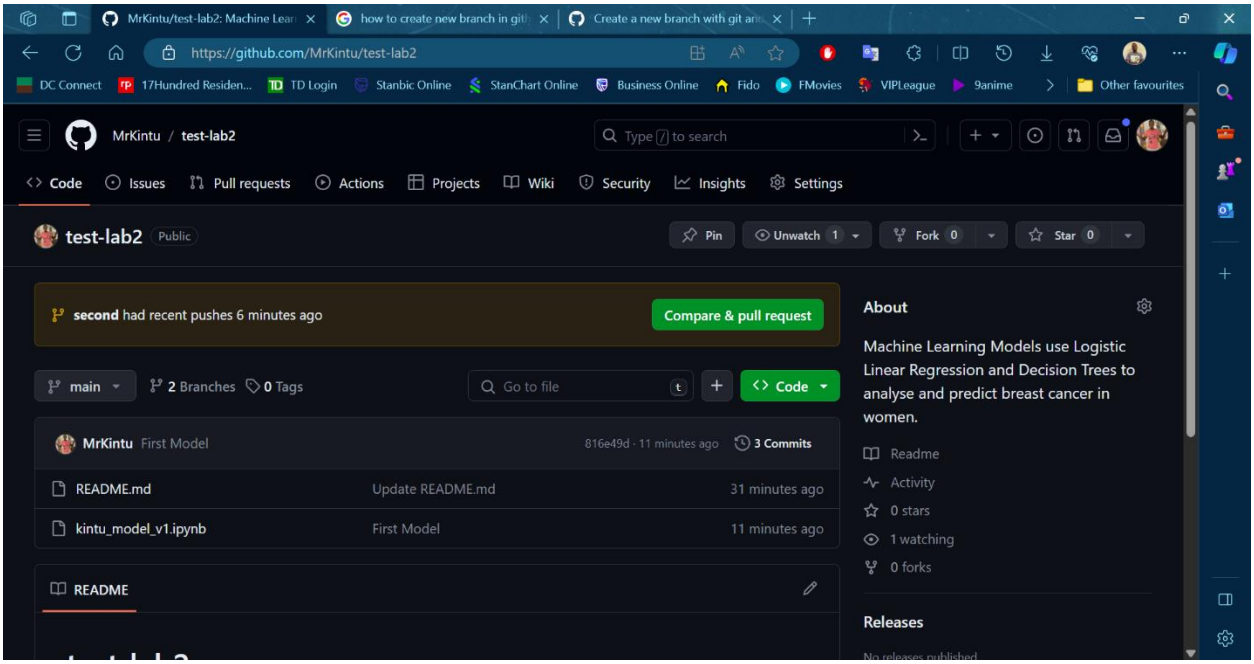


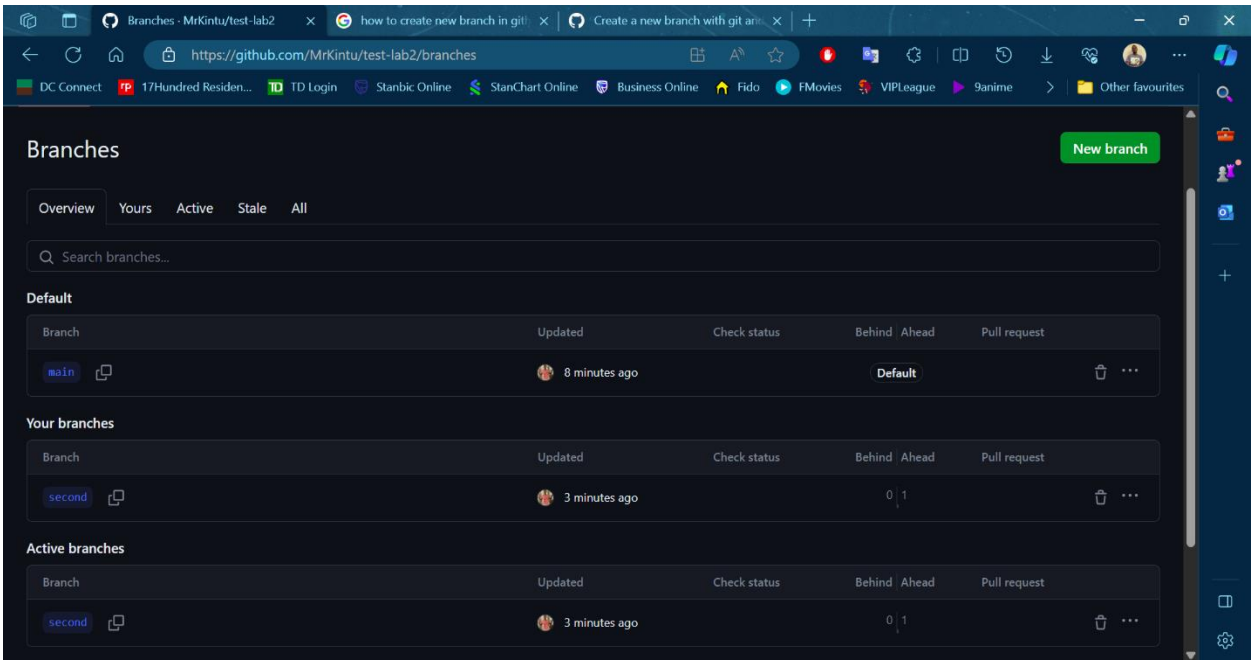

PowerShell
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.
PS C:\Users\d-kin\OneDrive\ClassWork\AIDI 2004\Labs\Lab2> git add .\kintu_model_v1.ipynb
warning: in the working copy of 'kintu_model_v1.ipynb', LF will be 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.

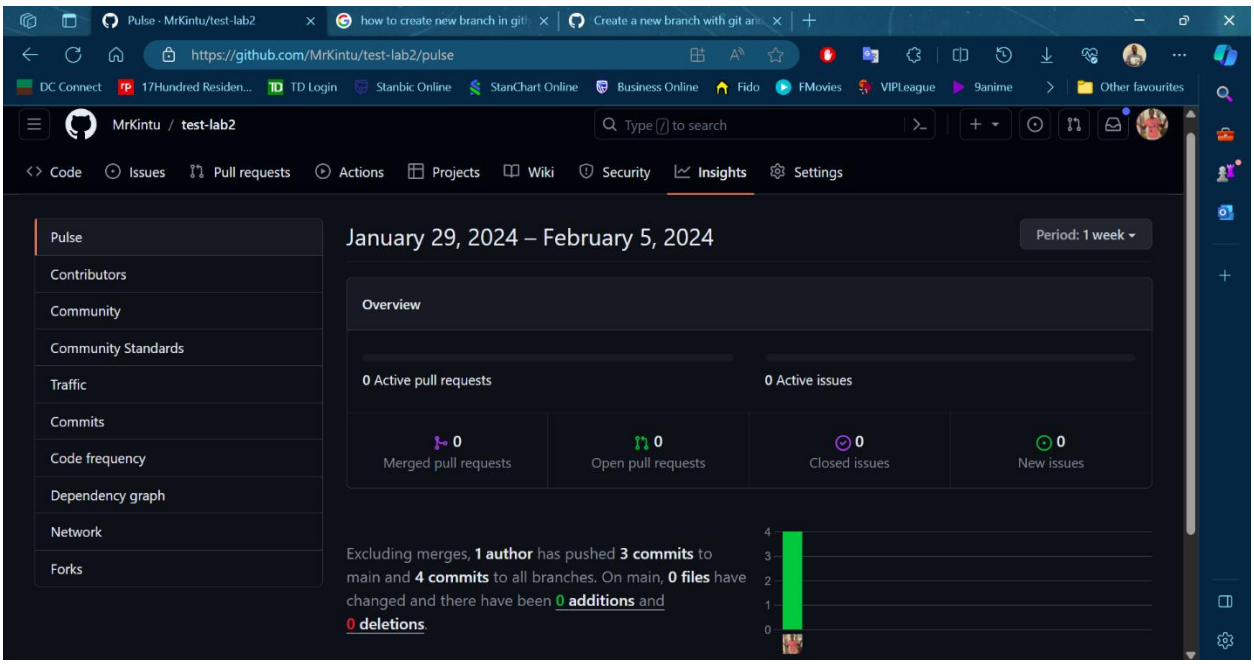
This is what the GitHub repository looks like after the commits.

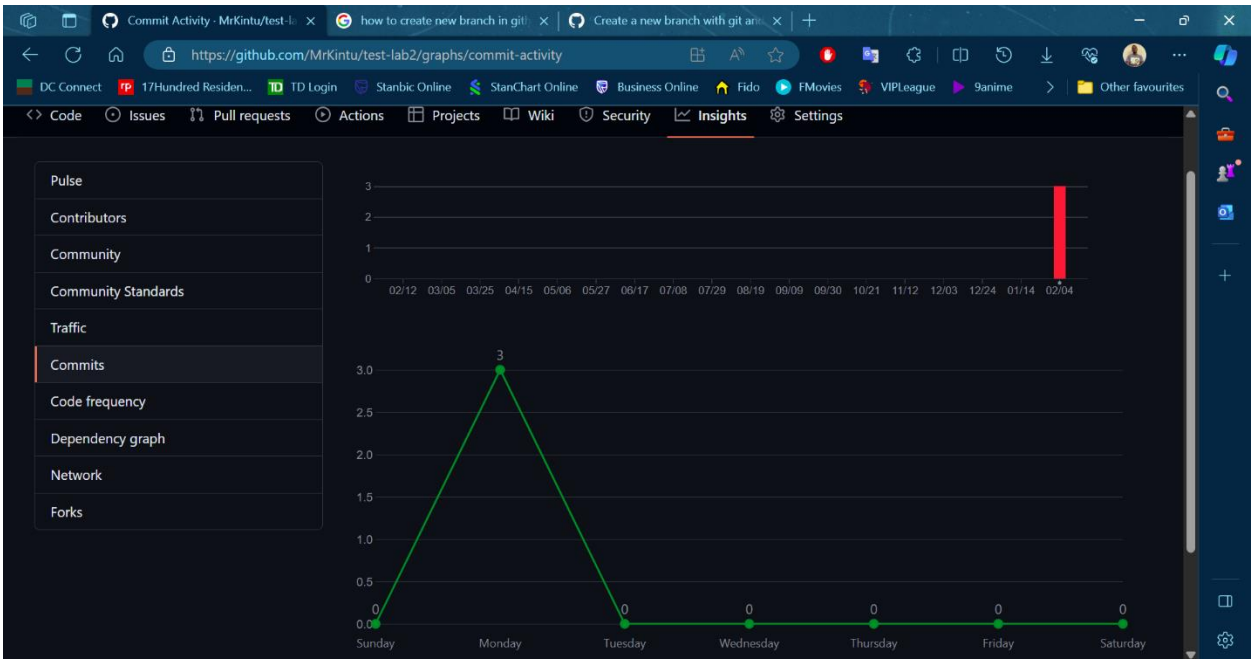




Step 7.

This is the repository activity log.





Step 8.

This is the README file attached to this project.