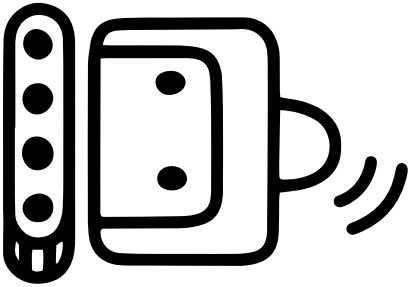
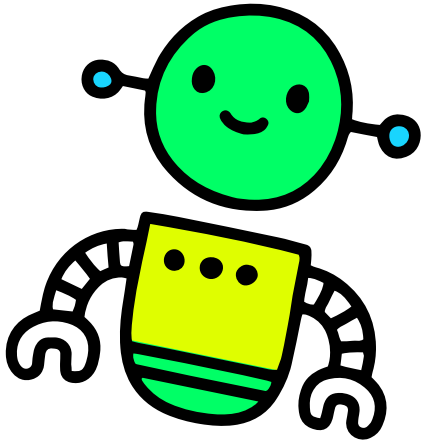
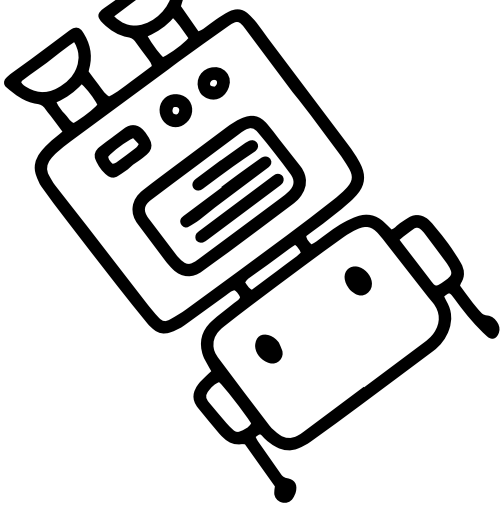
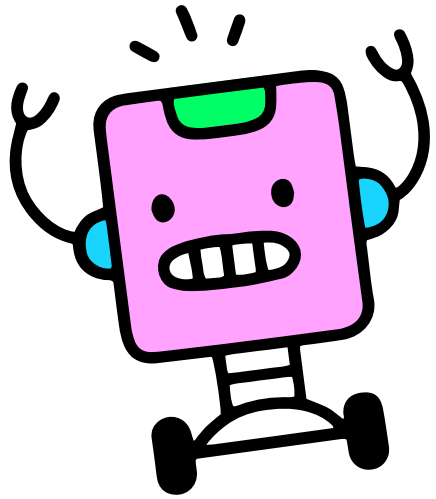
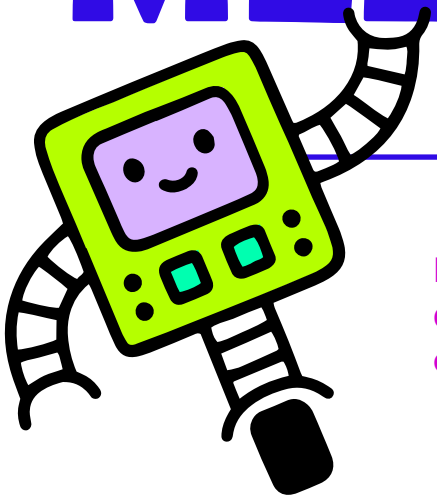






Machine learning MLE01





Fill in these silly robot doodles with the colors of your choice.

Mix and match bright colors for a drawing that really stands out.

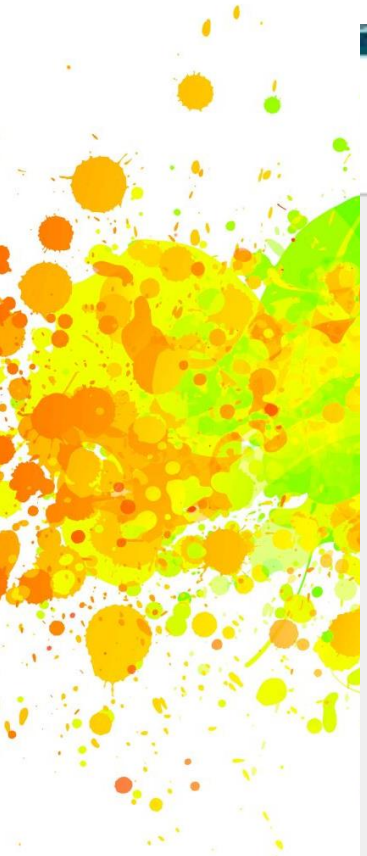
Instructions:
Print. Color. Repeat.

jupyter Class_01_Classsification Last Checkpoint 7 minutes ago

File Edit View Run Kernel Settings Help

Code

[ ]:
```
import numpy as np # Linear Algebra
import pandas as pd # data handling
import matplotlib.pyplot as plt # for visualization
import seaborn as sns
%matplotlib inline

from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer # Handles missing values
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```



jupyter Class_01_Classsification Last Checkpoint 10 minutes ago

File Edit View Run Kernel Settings Help

Code

[ ]:

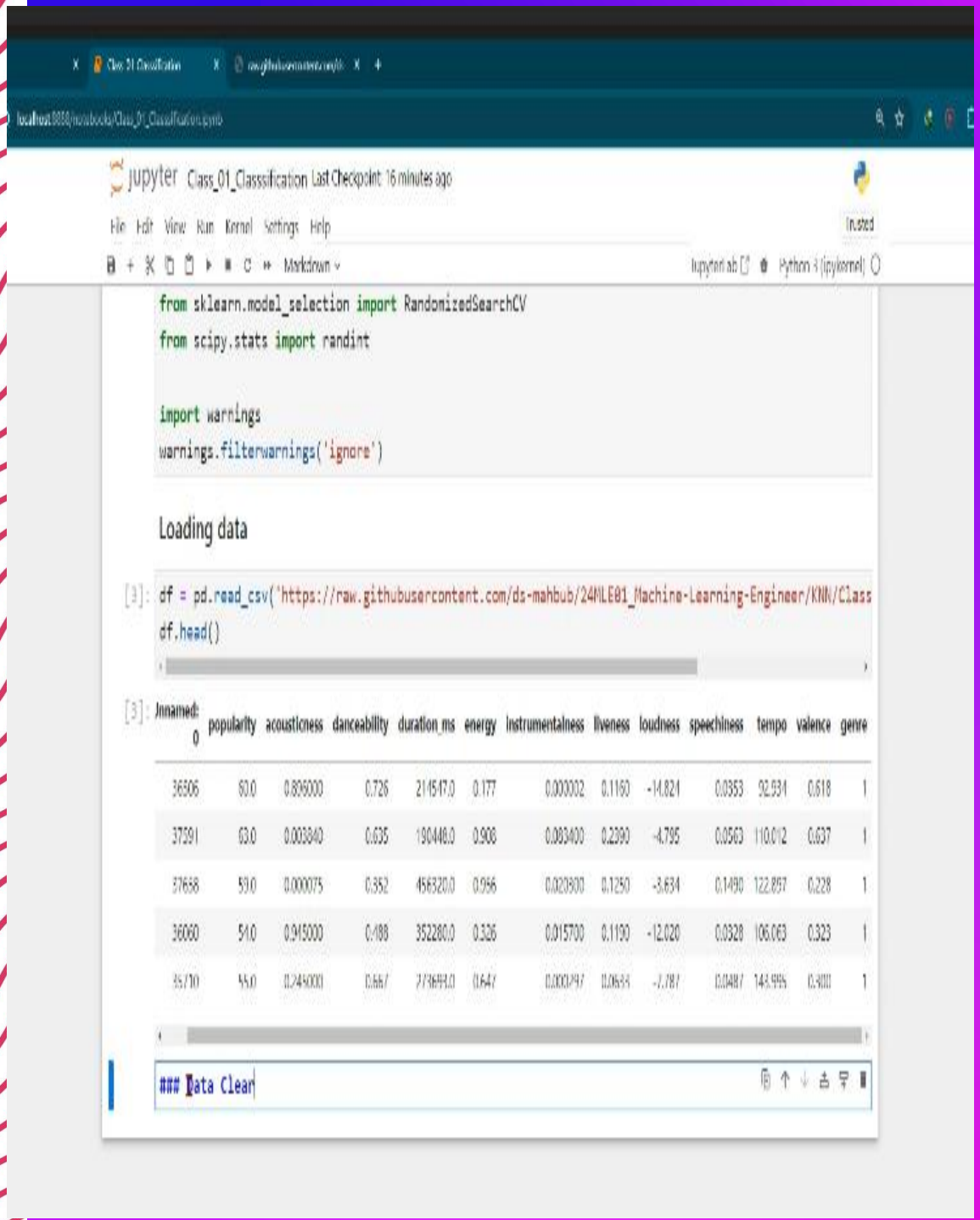
```
import numpy as np # Linear Algebra
import pandas as pd # data handling
import matplotlib.pyplot as plt # for visualization
import seaborn as sns
%matplotlib inline

from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer # Handles missing valu
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix


from sklearn.ensemble import ExtraTreesClassifier # feature sel
from sklearn.model_selection import GridSearchCV # hyperparamet
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
```

File Edit View Run Kernel Settings Help

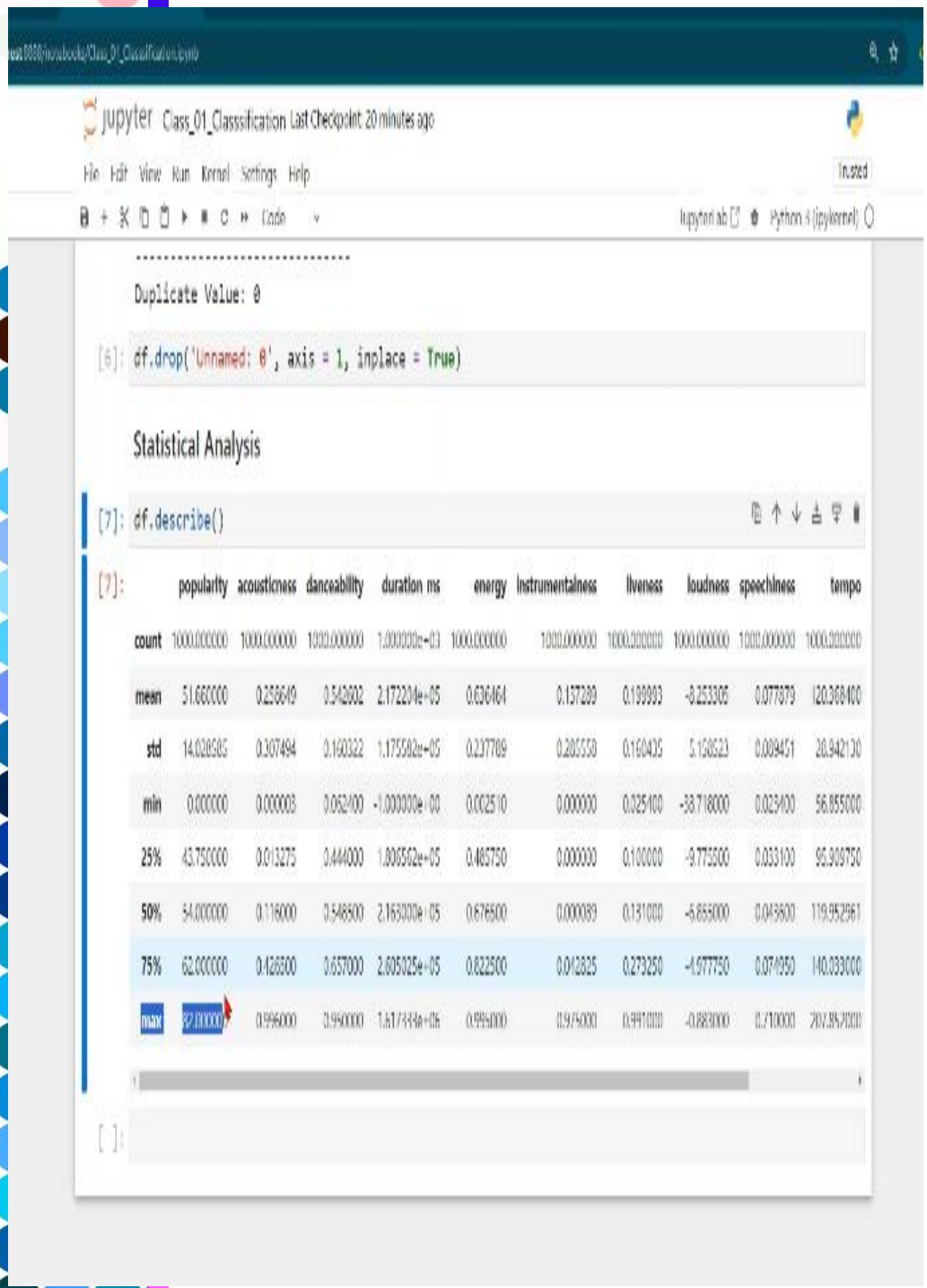
Code

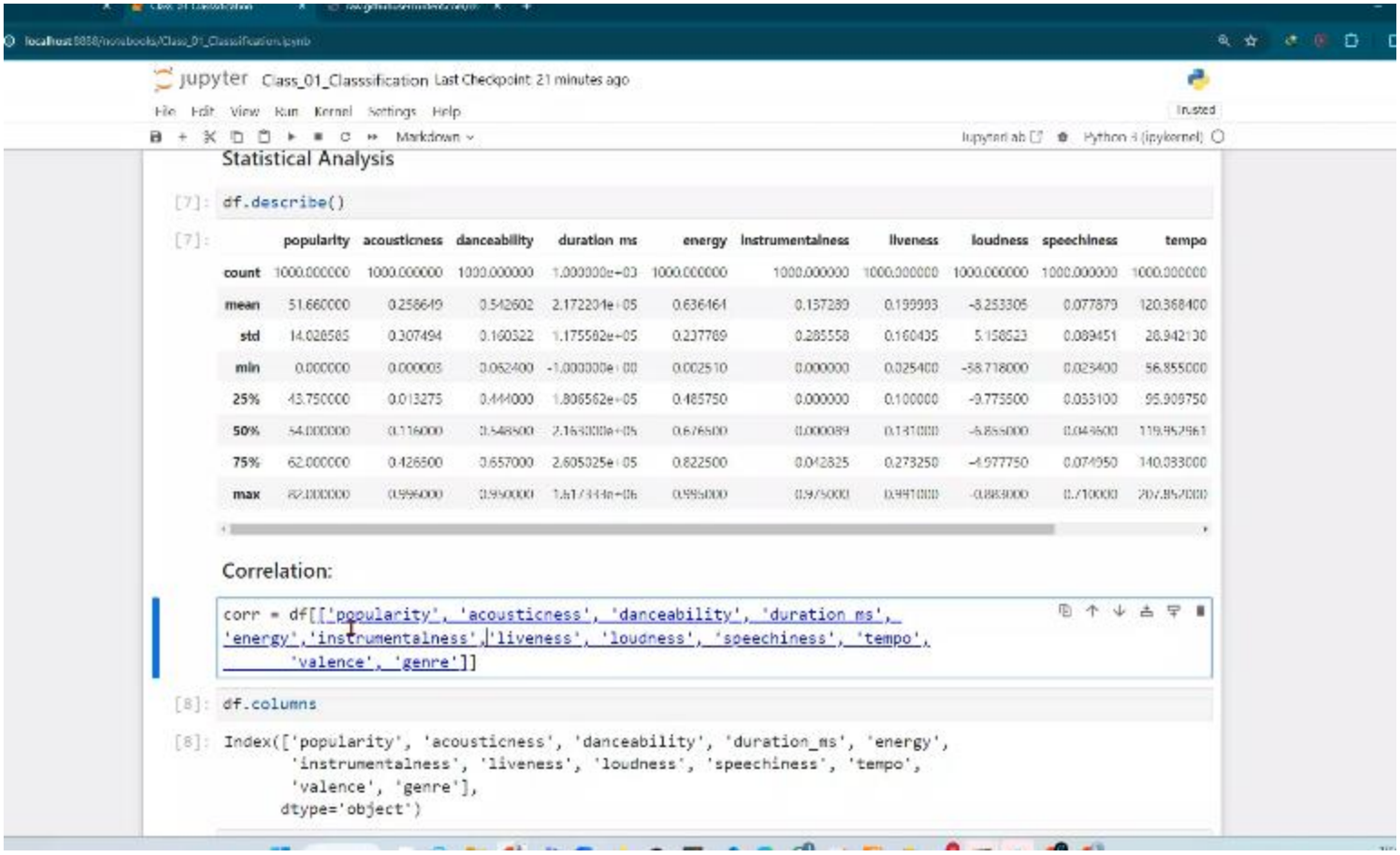
[2]:
```
import numpy as np # Linear Algebra
import pandas as pd # data handling
import matplotlib.pyplot as plt # for visualization
import seaborn as sns
%matplotlib inline

from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer # Handles missing values
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix


from sklearn.ensemble import ExtraTreesClassifier # feature selection
from sklearn.model_selection import GridSearchCV # hyperparameter tuning
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

import warnings
warnings.filterwarnings('ignore')
```

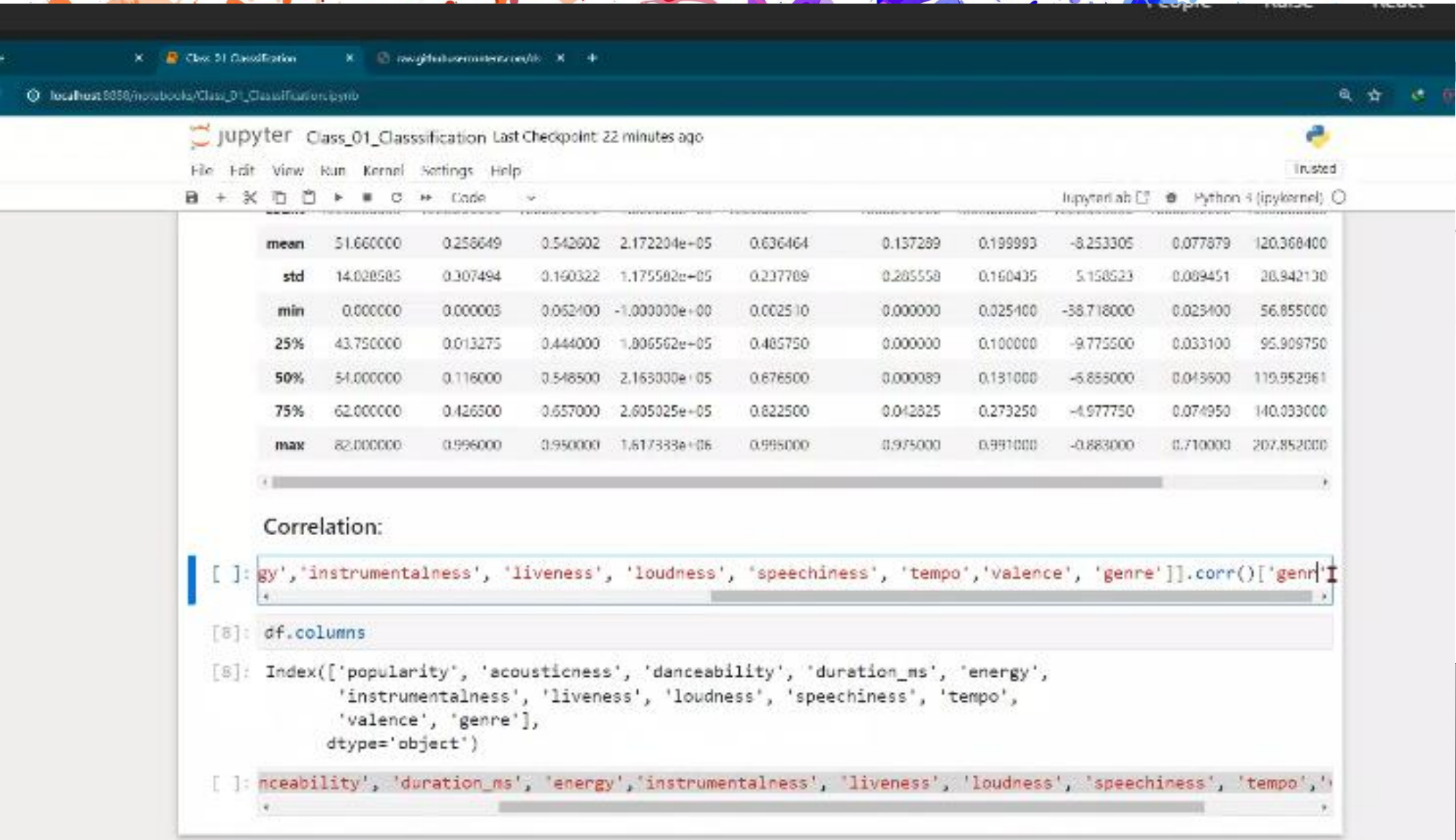
[ ]:


jupyter Class_01_Classsification Last Checkpoint: 16 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Markdown
Python 3 (ipykernel)
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
import warnings
warnings.filterwarnings('ignore')
Loading data
[3]: df = pd.read_csv('https://raw.githubusercontent.com/ds-mahbub/24MLE01_Machine-Learning-Engineer/KNN/Class
df.head()
[3]: Unnamed: 0 popularity acousticness danceability duration_ms energy instrumentalness liveness loudness speechiness tempo valence genre
36506 60.0 0.896000 0.726 214547.0 0.177 0.000002 0.1160 -14.824 0.0353 92.934 0.618 1
37391 63.0 0.003840 0.635 190448.0 0.908 0.083400 0.2390 -4.795 0.0563 110.012 0.637 1
37658 53.0 0.000075 0.352 456320.0 0.956 0.020800 0.1250 -3.634 0.1490 122.857 0.228 1
36060 54.0 0.945000 0.188 352280.0 0.326 0.015700 0.1190 -12.020 0.0328 106.063 0.323 1
35770 55.0 0.244000 0.667 273693.0 0.647 0.000297 0.0684 -2.787 0.0487 143.995 0.801 1
Data Clean


jupyter Class_01_Classification Last Checkpoint: 16 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Python 3 (ipykernel)
Data Cleaning
[5]: print(df.isna().sum())
print('.....')
print(f'Duplicate Value: {df.duplicated().sum()}')
Unnamed: 0 0
popularity 0
acousticness 0
danceability 0
duration_ms 0
energy 0
instrumentalness 0
liveness 0
loudness 0
speechiness 0
tempo 0
valence 0
genre 0
dtype: int64
.....
Duplicate Value: 0
[ ]: df.drop('Unnamed: 0', axis = 1, inplace = True)


jupyter Class_01_Classification Last Checkpoint: 20 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Python 3 (ipykernel)
Duplicate Value: 0
[6]: df.drop('Unnamed: 0', axis = 1, inplace = True)
Statistical Analysis
[7]: df.describe()
[7]:
popularity acousticness danceability duration_ms energy instrumentalness liveness loudness speechiness tempo
count 1000.000000 1000.000000 1000.000000 1.000000e+03 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000
mean 51.660000 0.258649 0.542602 2.172204e+05 0.636464 0.137289 0.199993 -8.253305 0.077879 120.368400
std 14.028505 0.307494 0.160322 1.175502e+05 0.237709 0.280559 0.160435 5.158523 0.089451 28.942130
min 0.000000 0.000003 0.062400 -1.000000e+00 0.002510 0.000000 0.025100 -38.718000 0.023400 56.855000
25% 43.750000 0.013275 0.444000 1.806562e+05 0.485750 0.000000 0.100000 -9.775500 0.033100 95.909750
50% 54.000000 0.116000 0.548500 2.163000e+05 0.676500 0.000089 0.131000 -6.655000 0.045600 119.952561
75% 62.000000 0.426500 0.657000 2.605025e+05 0.822500 0.012825 0.273250 -4.977750 0.074950 140.053000
max 87.000000 0.996000 0.990000 1.617333e+06 0.994000 0.976000 0.997000 -0.883000 0.470000 207.857000


jupyter Class_01_Classsification Last Checkpoint: 21 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Markdown
JupyterLab
Python 3 (ipykernel)
Statistical Analysis
[7]: df.describe()
[7]:
popularity acousticness danceability duration_ms energy instrumentalness liveness loudness speechiness tempo
count 1000.000000 1000.000000 1000.000000 1.000000e+03 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000
mean 51.660000 0.258649 0.542602 2.172201e+05 0.636464 0.137280 0.199993 -8.253305 0.077879 120.368400
std 14.028585 0.307494 0.160522 1.175582e+05 0.237789 0.285558 0.160435 5.158523 0.089451 28.942130
min 0.000000 0.000003 0.062400 -1.000000e+00 0.002510 0.000000 0.025400 -58.718000 0.023400 56.855000
25% 43.750000 0.013275 0.444000 1.806562e+05 0.485750 0.000000 0.100000 -9.775500 0.033100 95.909750
50% 54.000000 0.116000 0.548500 2.163000e+05 0.676500 0.000089 0.131000 -6.855000 0.044600 119.952961
75% 62.000000 0.426500 0.657000 2.605025e+05 0.822500 0.042825 0.273250 -4.977750 0.074050 140.033000
max 82.000000 0.996000 0.990000 1.617334e+06 0.995000 0.969000 0.991000 -0.883000 0.710000 207.852000
Correlation:
corr = df[['popularity', 'acousticness', 'danceability', 'duration_ms',
'energy','instrumentalness','liveness', 'loudness', 'speechiness', 'tempo',
'valence', 'genre']]
[8]: df.columns
[8]: Index(['popularity', 'acousticness', 'danceability', 'duration_ms', 'energy',
'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',
'valence', 'genre'],
dtype='object')


jupyter Class_01_Classsification Last Checkpoint: 22 minutes ago
File Edit View Run Kernel Settings Help
Trusted
JupyterLab
Python 3 (ipykernel)
mean 51.660000 0.258649 0.542602 2.172204e+05 0.636464 0.137289 0.199993 -8.253305 0.077879 120.368400
std 14.028585 0.307494 0.160322 1.175582e+05 0.237709 0.285559 0.160435 5.158523 0.089451 28.942130
min 0.000000 0.000003 0.052400 -1.000000e+00 0.002510 0.000000 0.025400 -38.718000 0.023400 56.855000
25% 43.750000 0.013275 0.444000 1.806562e+05 0.485750 0.000000 0.100000 -9.775500 0.033100 95.909750
50% 54.000000 0.116000 0.548500 2.163000e+05 0.676500 0.000089 0.131000 -6.855000 0.043600 119.952961
75% 62.000000 0.426500 0.657000 2.605025e+05 0.822500 0.042825 0.273250 -4.977750 0.074950 140.033000
max 82.000000 0.996000 0.950000 1.617333e+06 0.996000 0.975000 0.991000 -0.883000 0.710000 207.852000
Correlation:
[ ]: gy','instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo','valence', 'genre']].corr()['genr
[6]: df.columns
[6]: Index(['popularity', 'acousticness', 'danceability', 'duration_ms', 'energy',
'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',
'valence', 'genre'],
dtype='object')
[ ]: nceability', 'duration_ms', 'energy','instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo','

jupyter Class_01_Classsification Last Checkpoint: 23 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

75%	62.000000	0.476000	0.665000	2.606075e+05	0.822500	0.042825	0.773750	-4.977750	0.049450	140.044000
max	82.000000	0.996000	0.950000	1.617333e+06	0.995000	0.975000	0.991000	-0.883000	0.710000	207.852000

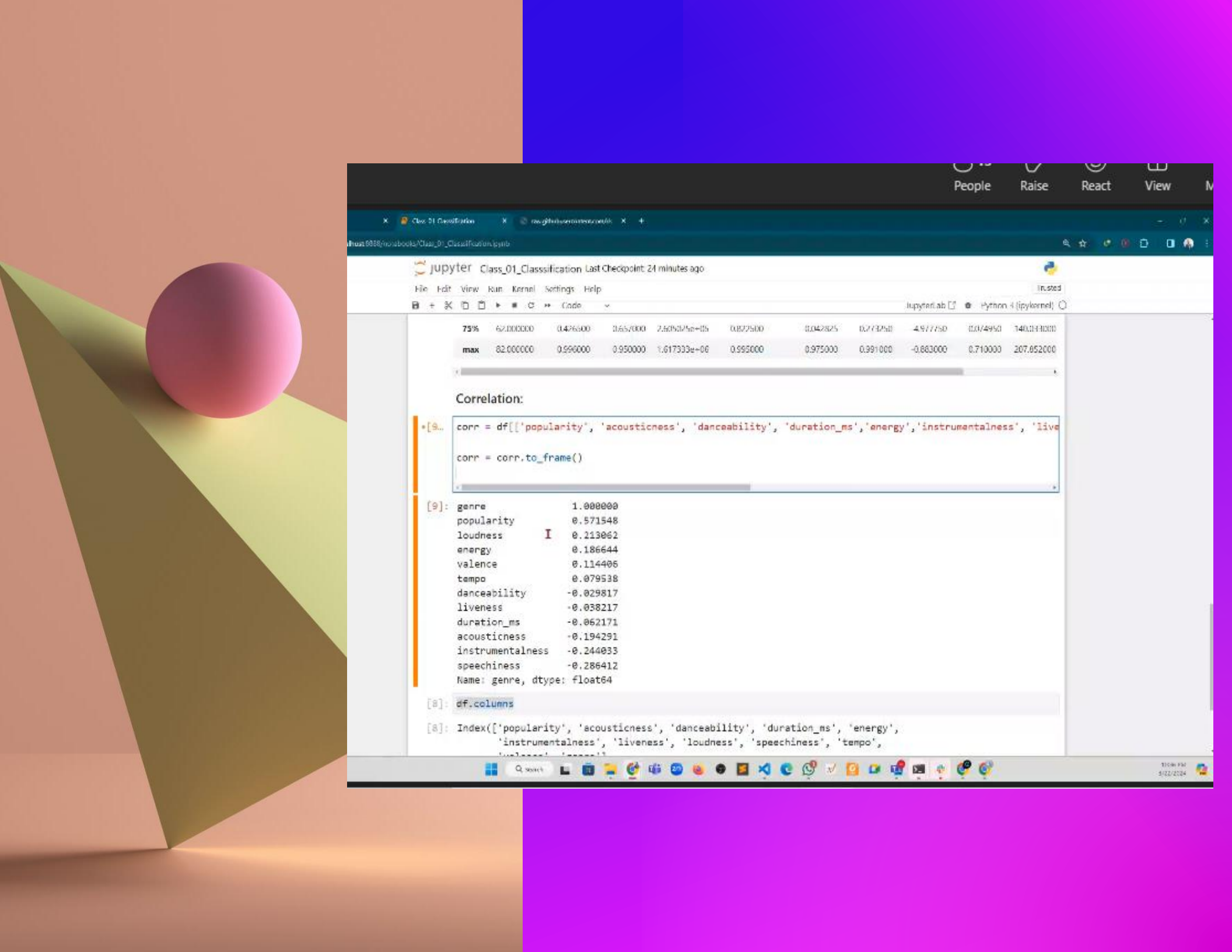
Correlation:

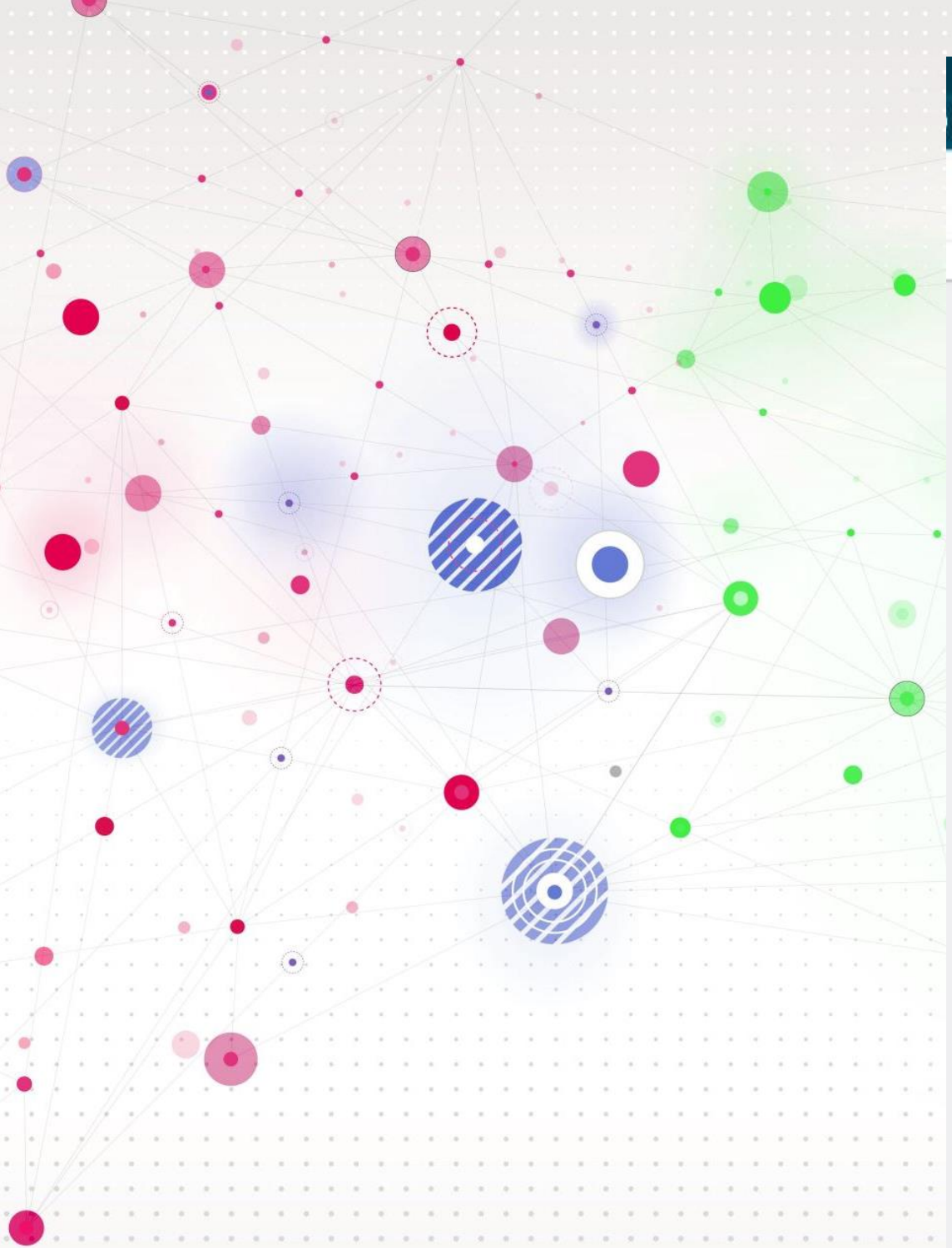
[9]:
```
corr = df[['popularity', 'acousticness', 'danceability', 'duration_ms','energy','instrumentalness', 'live
corr
```

[9]:
```
genre              1.000000
popularity         0.571548
loudness           0.213062
energy             0.186644
valence            0.114406
tempo              0.079538
danceability      -0.029817
liveness          -0.038217
duration_ms       -0.062171
acousticness      -0.194291
instrumentalness  -0.244033
speechiness       -0.286412
Name: genre, dtype: float64
```

[8]:
```
df.columns
```

[8]:
```
Index(['popularity', 'acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',
       'valence', 'genre'],
      dtype='object')
```


People
Raise
React
View
jupyter Class_01_Classsification Last Checkpoint: 24 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
jupyterlab
Python 3 (ipykernel)
75% 62.000000 0.426000 0.665000 2.469937e+05 0.872500 0.042825 0.273250 -4.977750 0.074950 140.018000
max 82.000000 0.996000 0.950000 1.617333e+06 0.995000 0.975000 0.991000 -0.883000 0.710000 207.052000
Correlation:
[9]: corr = df[['popularity', 'acousticness', 'danceability', 'duration_ms','energy','instrumentalness', 'live
corr = corr.to_frame()
[9]: genre 1.000000
popularity 0.571548
loudness 0.213062
energy 0.186644
valence 0.114406
tempo 0.079538
danceability -0.029817
liveness -0.038217
duration_ms -0.062171
acousticness -0.194291
instrumentalness -0.244033
speechiness -0.286412
Name: genre, dtype: float64
[8]: df.columns
[8]: Index(['popularity', 'acousticness', 'danceability', 'duration_ms', 'energy',
'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',



Class_01_Classification Last Checkpoint: 25 minutes ago

File Edit View Run Kernel Settings Help

Code

[11]:
```
corr = df[['popularity', 'acousticness', 'danceability', 'duration_ms','energy','instrumentalness', 'live

corr = corr.to_frame()
corr.style.background_gradient(cmap = 'RdYlBu')
```

[11]:

	genre
genre	1.000000
popularity	0.571549
loudness	0.271067
energy	0.196644
valence	0.114406
tempo	0.079530
danceability	0.029817
liveness	-0.038217
duration_ms	-0.062171
acousticness	-0.194291
instrumentalness	-0.244033
speechiness	-0.286417

[8]:
```
df.columns
```

[8]:
```
Index(['popularity', 'acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',
       'valence', 'genre'],
      dtype='object')
```

jupyter Class_01_Classsification Last Checkpoint: 29 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab

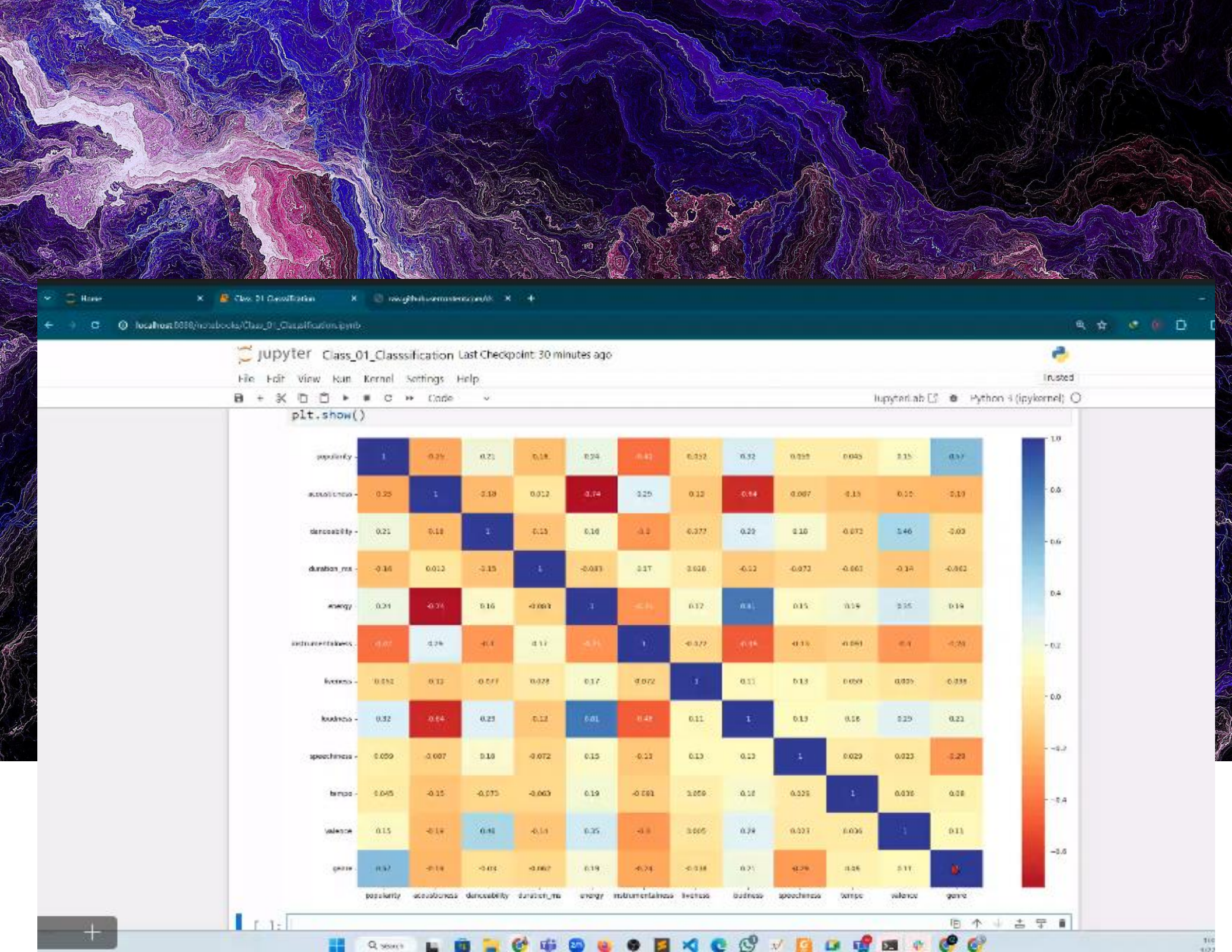
Python 3 (ipykernel)

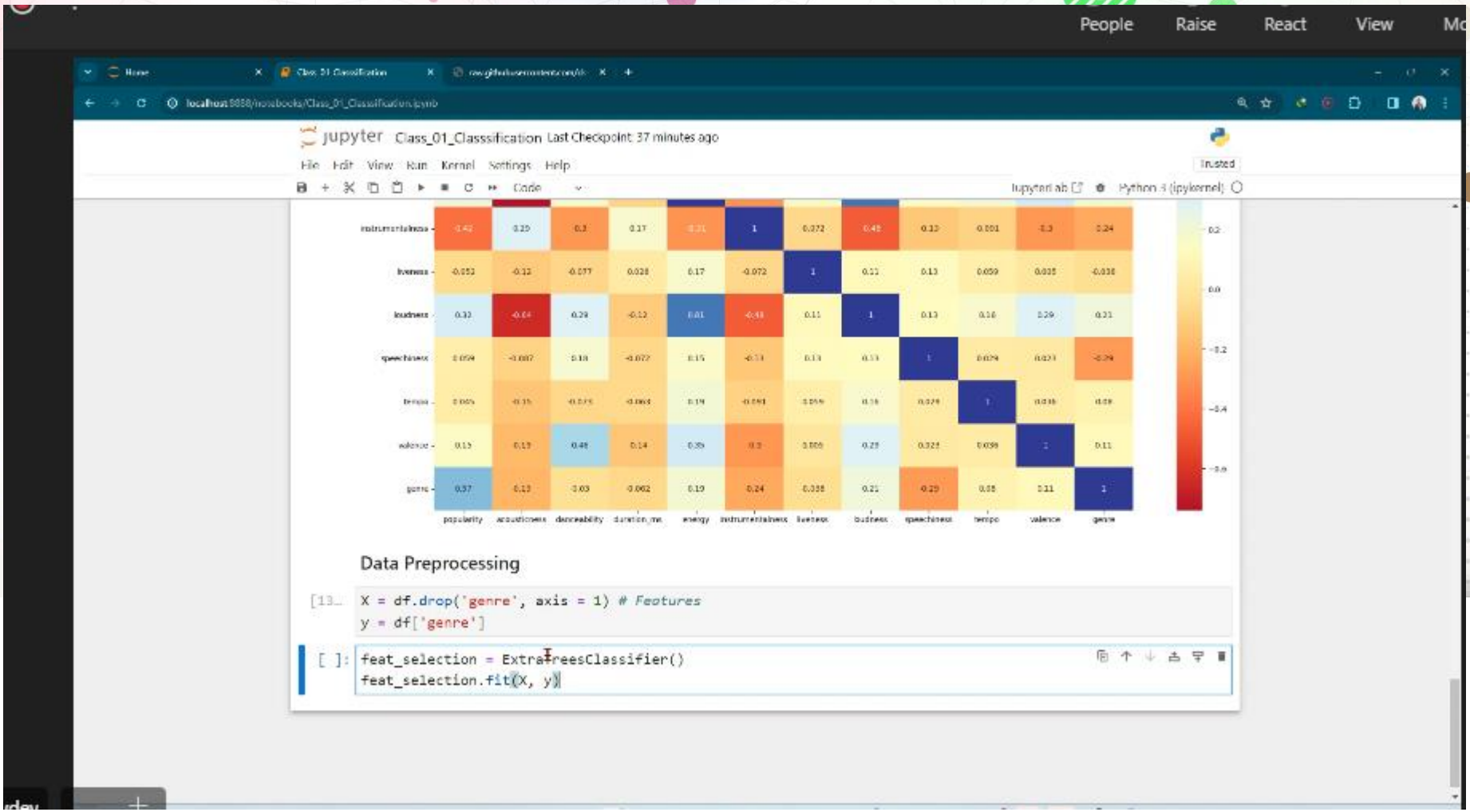
popularity	0.571548
loudness	0.213062
energy	0.186641
valence	0.114406
tempo	0.079538
danceability	-0.029817
liveness	-0.038217
duration_ms	-0.062171
acousticness	-0.194241
instrumentalness	-0.244033
speechiness	0.286412

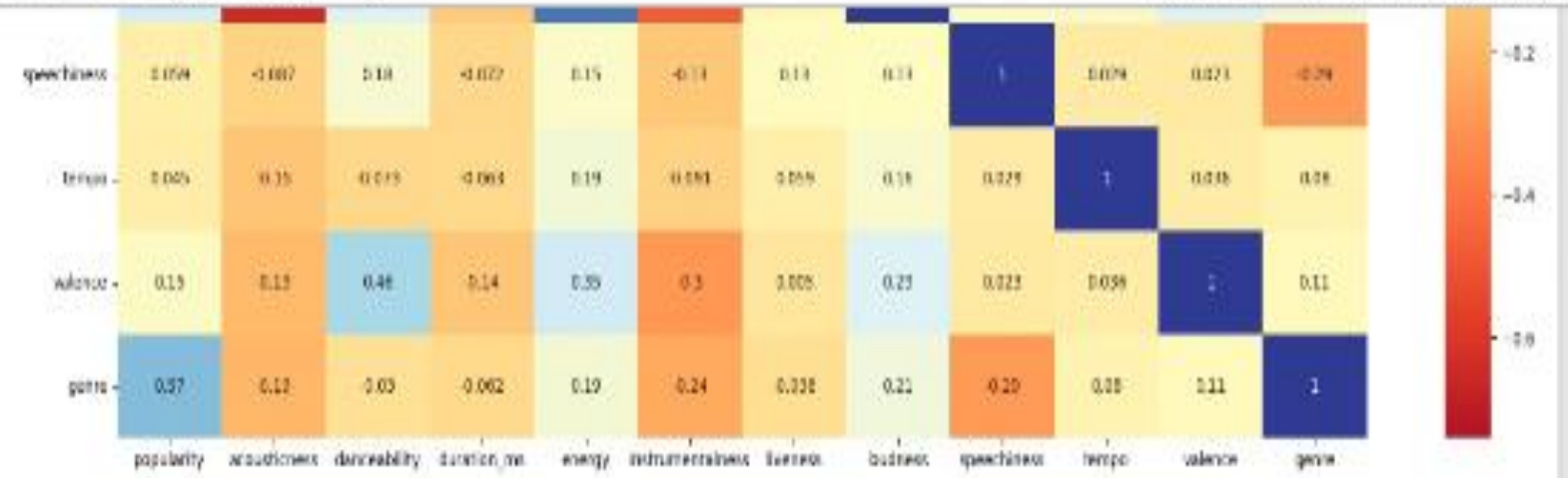
[8]:
```
df.columns
```

[8]:
```
Index(['popularity', 'acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',
       'valence', 'genre'],
      dtype='object')
```

[ ]:
```
plt.figure(figsize = (20, 12))
sns.heatmap(df.corr(), annot = True, cma)
```


jupyter Class_01_Classsification Last Checkpoint: 30 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
plt.show()
popularity
acousticness
danceability
duration_ms
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
genre


People
Raise
React
View
jupyter Class_01_Classification Last Checkpoint: 37 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
popularity
acousticness
danceability
duration_ms
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence
genre
Data Preprocessing
[13...
X = df.drop('genre', axis = 1) # Features
y = df['genre']
[ ]:
feat_selection = ExtraTreesClassifier()
feat_selection.fit(X, y)



Data Preprocessing

[13]
```
X = df.drop('genre', axis = 1) # Features
y = df['genre']
```

[14]
```
feat_selection = ExtraTreesClassifier()
feat_selection.fit(X, y)
```

[14]
```
ExtraTreesClassifier()
```

[ ]
```
feat_importances = pd.Series(feat_selection.feature_importances_, index = X.columns)
```

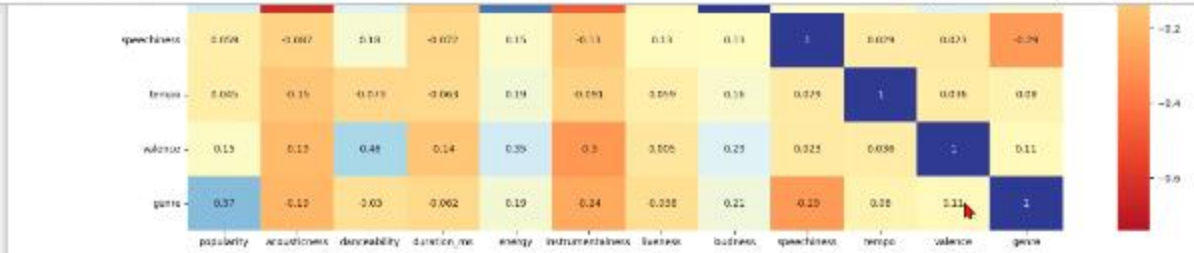
jupyter Class_01_Classsification Last Checkpoint: 39 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)



Data Preprocessing

[13...]
```
X = df.drop('genre', axis = 1) # Features
y = df['genre']
```

[14...]
```
feat_selection = ExtraTreesClassifier()
feat_selection.fit(X, y)
```

[14...]
```
ExtraTreesClassifier()
```

[ ]:
```
feat_importances = pd.Series(feat_selection.feature_importances_, index = X.columns)
feat_importances.nlargest(len(df.columns)).plot(kind = 'b')
```

jupyter Class_01_Classsification Last Checkpoint: 41 minutes ago

File Edit View Run Kernel Settings Help

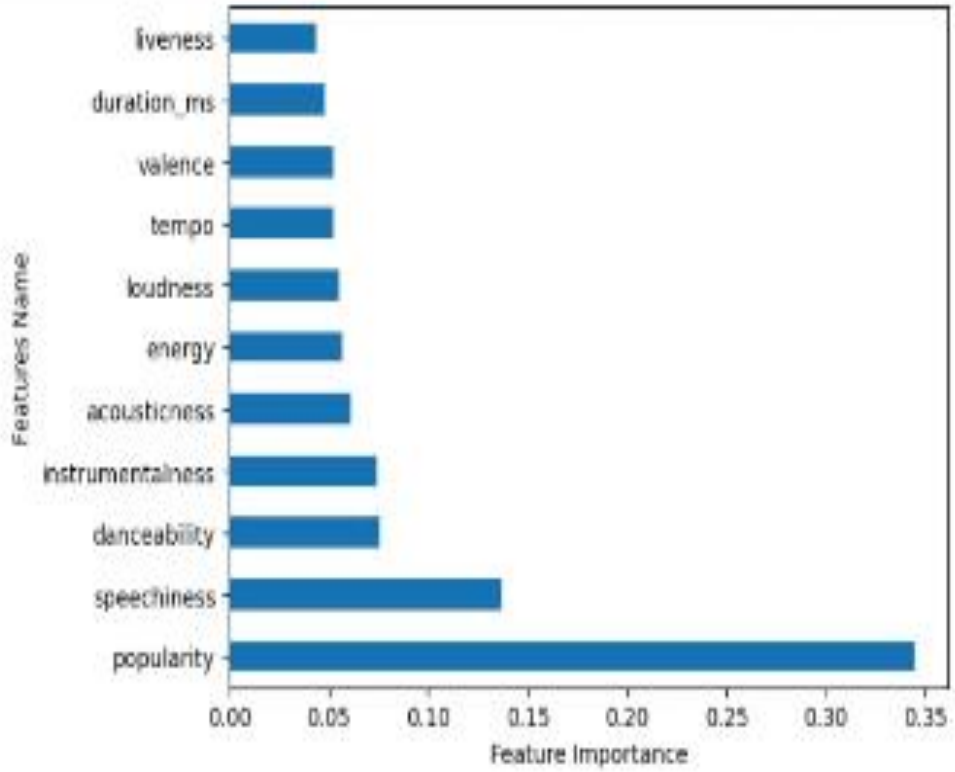
Trusted

Code

JupyterLab

Python 3 (ipykernel)

```
feat_importances.nlargest(len(df.columns)).plot(kind = 'barh')
plt.xlabel('Feature Importance')
plt.ylabel('Features Name')
plt.show()
```



[ ]:

People Raise React View

jupyter Class_01_Classsification Last Checkpoint: 42 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

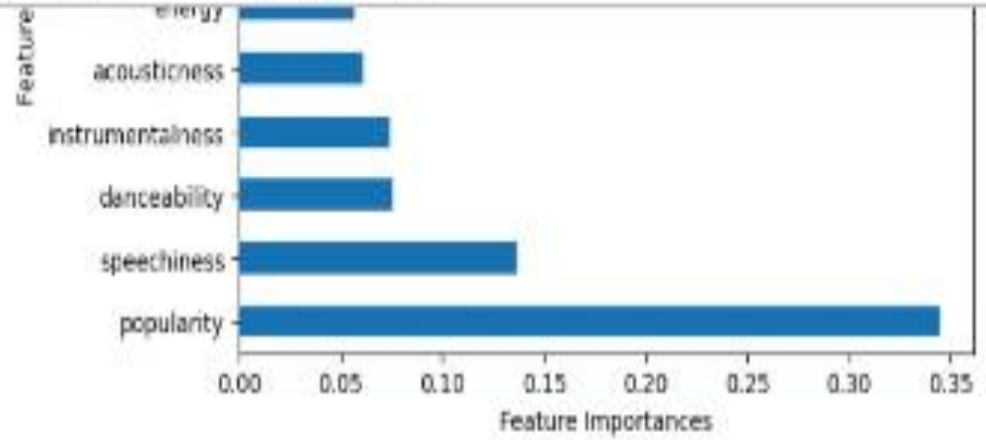
Data Preprocessing

```
[13]: X = df.drop('genre', axis = 1) # Features
      y = df['genre']
```

```
[14]: feat_selection = ExtraTreesClassifier()
      feat_selection.fit(X, y)
```

```
[14]: ExtraTreesClassifier
      ExtraTreesClassifier()
```

```
[17]: feat_importances = pd.Series(feat_selection.feature_importances_, index = X.columns)
      feat_importances.nlargest(len(df.columns)).plot(kind = 'barh')
      plt.xlabel('Feature Importances')
      plt.ylabel('Features Name')
      plt.show()
```



[18..
```
feat_importances.sort_values(ascending = False)
```

[18..
```
popularity          0.344300
speechiness         0.136679
danceability        0.075298
instrumentalness    0.074093
acousticness        0.061256
energy              0.056974
loudness            0.054959
tempo               0.052449
valence             0.052240
duration_ms         0.048249
liveness            0.043503
dtype: float64
```

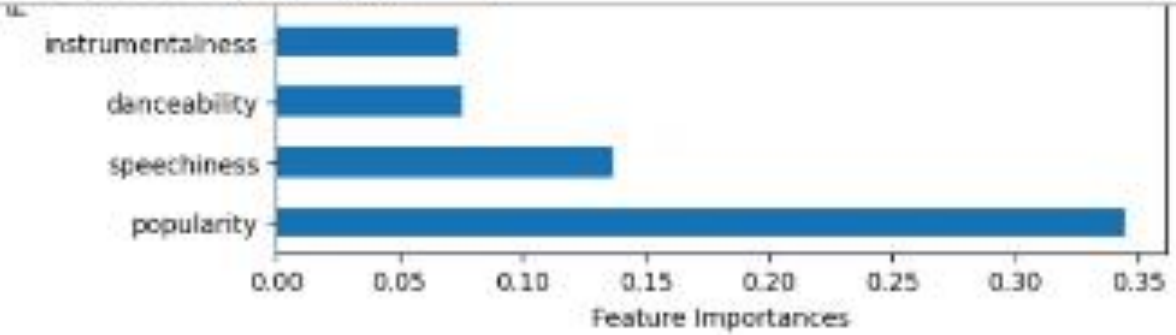
[ ]:

jupyter Class_01_Classsification Last Checkpoint: 44 minutes ago

File Edit View Run Kernel Settings Help

Code

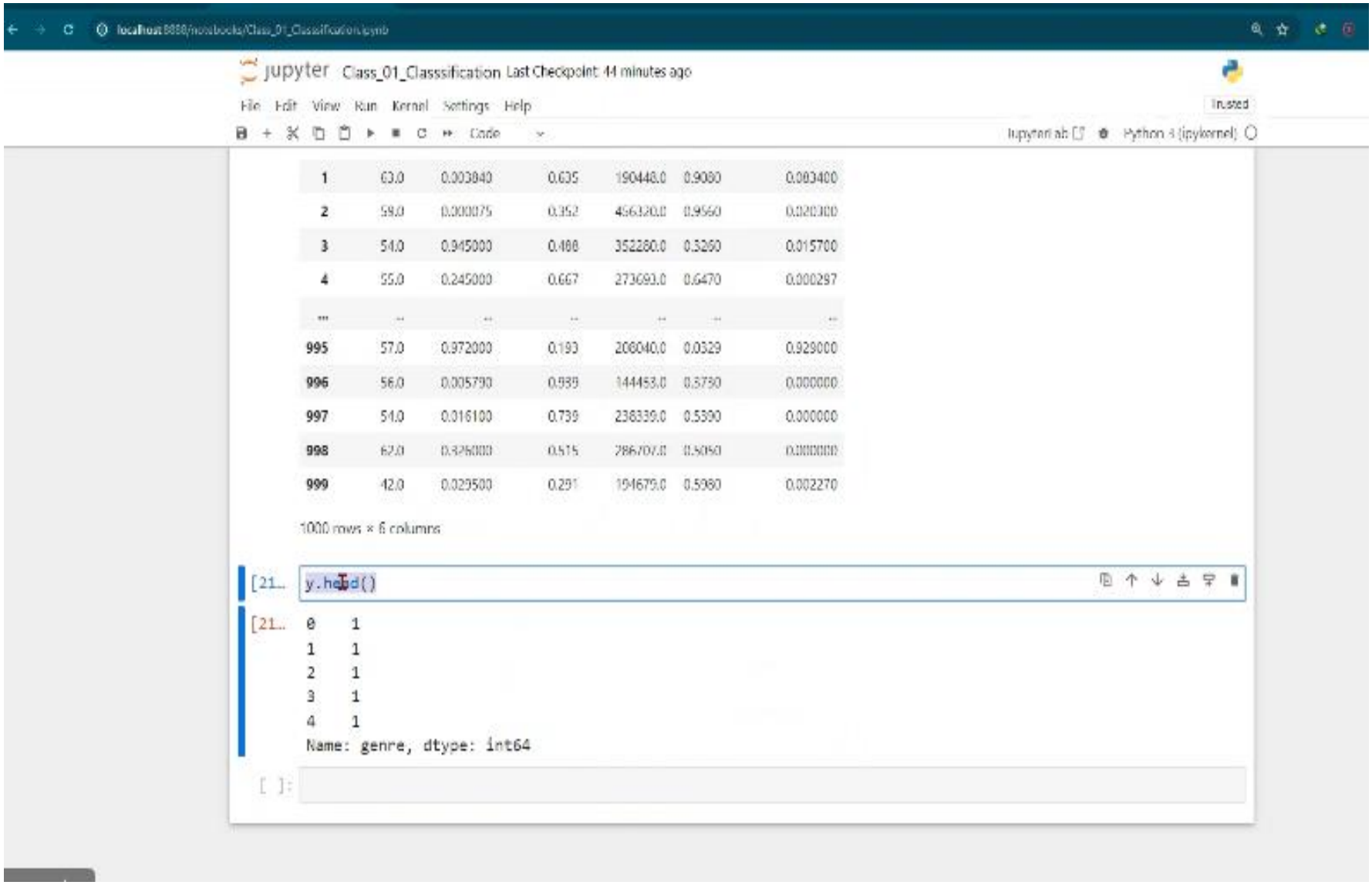
JupyterLab Python



[18...
```
feat_importances.sort_values(ascending = False)
```

[18...
```
popularity          0.344300
speechiness         0.136679
danceability        0.075298
instrumentalness    0.074093
acousticness        0.061256
energy              0.056974
loudness            0.054959
tempo               0.052449
valence             0.052240
duration_ms         0.048249
liveness            0.043503
dtype: float64
```

[ ]:
```
X = df[feat_importances[:6].index]
```


jupyter Class_01_Classsification Last Checkpoint: 44 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
1 63.0 0.003840 0.635 190448.0 0.9080 0.003400
2 58.0 0.000075 0.352 466320.0 0.9560 0.020300
3 54.0 0.945000 0.488 352260.0 0.5260 0.015700
4 55.0 0.245000 0.667 273693.0 0.6470 0.000297
... ...
995 57.0 0.972000 0.193 206040.0 0.0329 0.929000
996 56.0 0.005790 0.939 144453.0 0.5750 0.000000
997 54.0 0.016100 0.739 238339.0 0.5390 0.000000
998 62.0 0.326000 0.515 286707.0 0.5050 0.000000
999 42.0 0.029500 0.291 194679.0 0.5380 0.002270
1000 rows × 6 columns
[21...
y.head()
[21...
0 1
1 1
2 1
3 1
4 1
Name: genre, dtype: int64
[ ]:

jupyter Class_01_Classsification Last Checkpoint: 46 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

jupyterlab Python 3 (ipykernel)

[20...

X

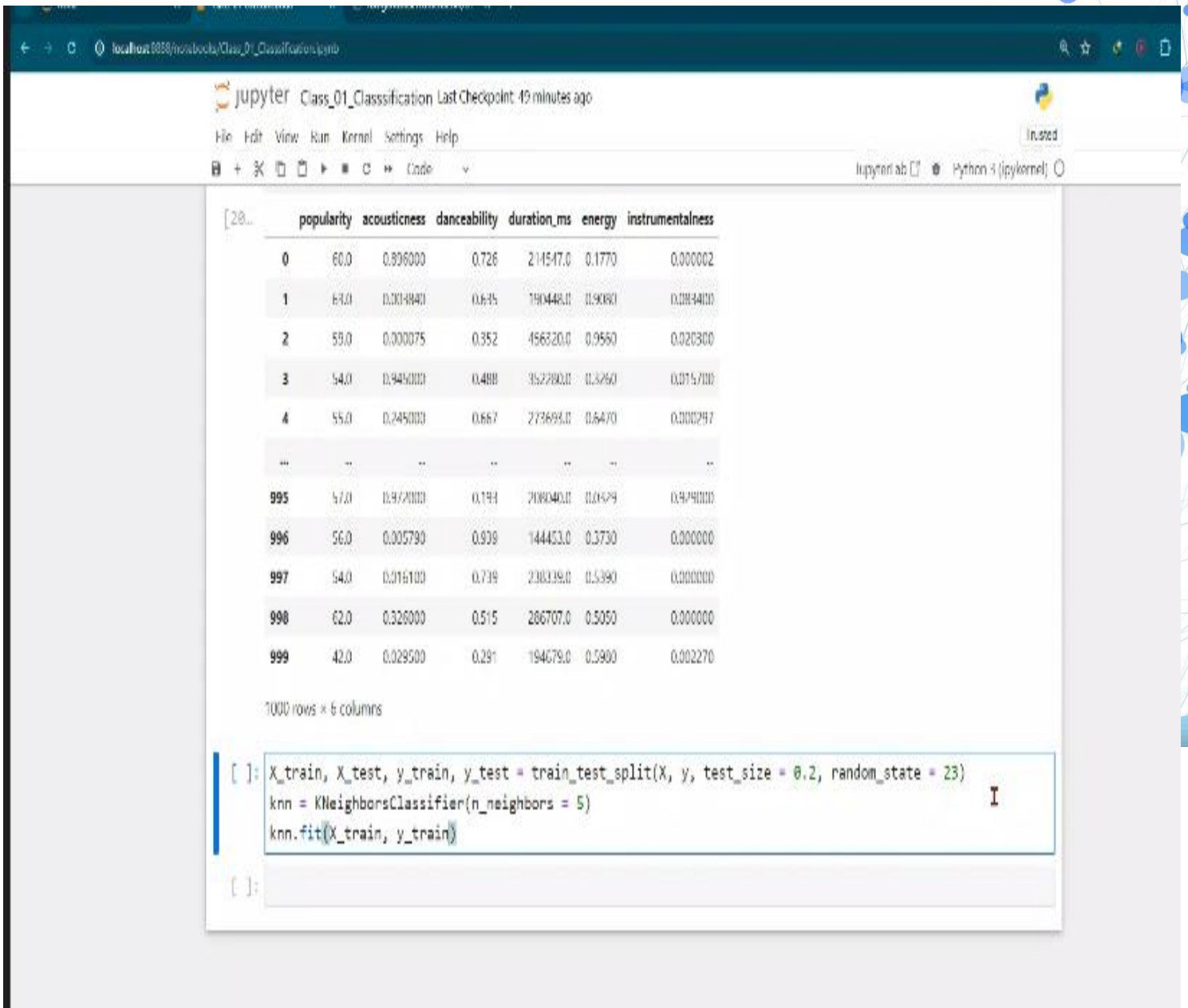
[20...

	popularity	acousticness	danceability	duration_ms	energy	instrumentalness
0	60.0	0.896000	0.726	214547.0	0.1770	0.000002
1	63.0	0.003840	0.635	190448.0	0.9000	0.083400
2	59.0	0.000175	0.352	456640.0	0.9560	0.010400
3	54.0	0.945000	0.466	352280.0	0.3260	0.015700
4	55.0	0.245000	0.667	273693.0	0.6470	0.000297
...	...	...	...	...	...	...
995	57.0	0.972000	0.193	200040.0	0.0329	0.929000
996	56.0	0.005790	0.939	144453.0	0.3750	0.000000
997	54.0	0.016100	0.739	236339.0	0.5390	0.000000
998	62.0	0.326000	0.515	286707.0	0.5050	0.000000
999	42.0	0.029500	0.291	194679.0	0.5980	0.002270

1000 rows × 6 columns

[ ]:
```
train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 23, stratify = y)
```

[ ]:



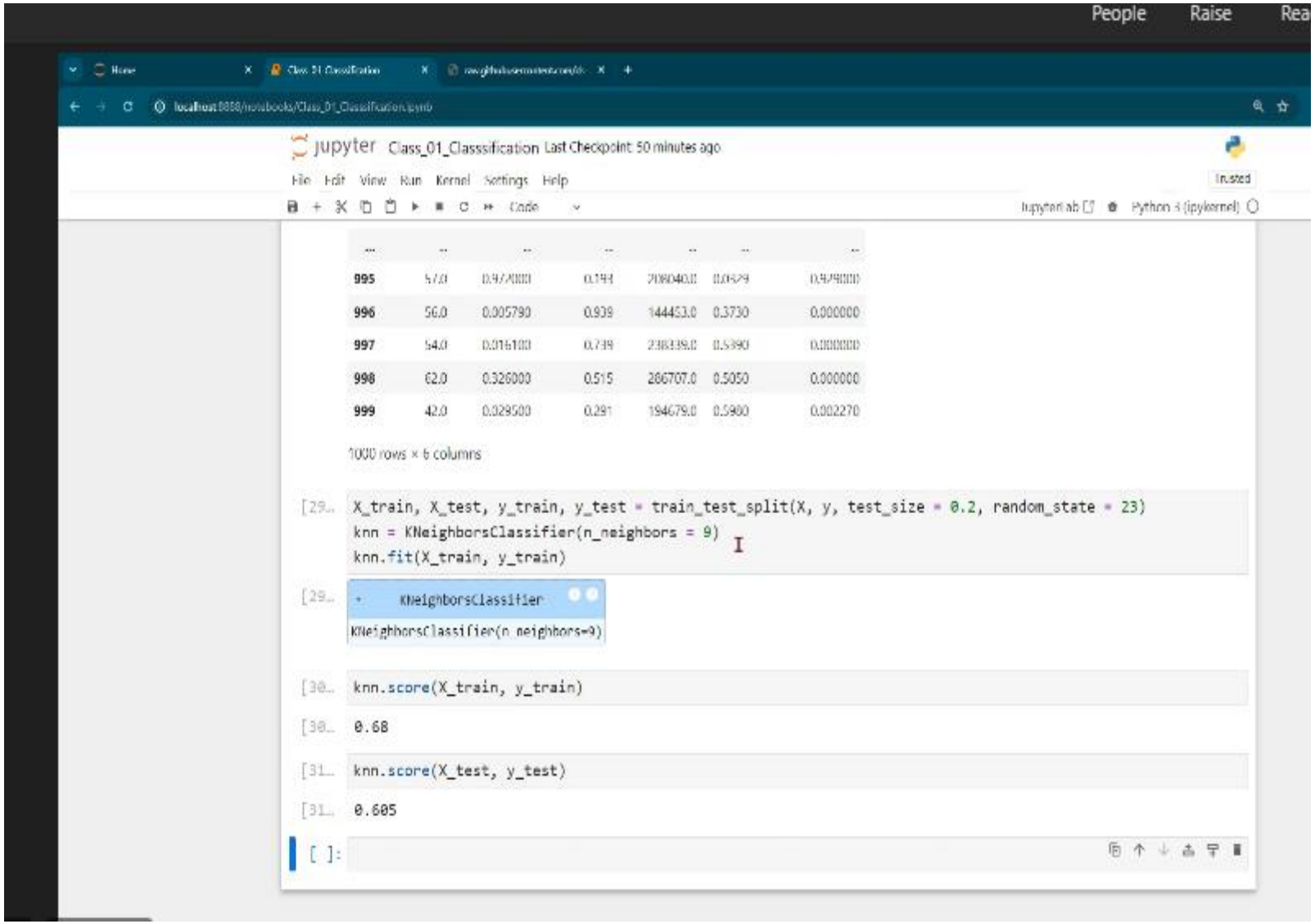
[20...

	popularity	acousticness	danceability	duration_ms	energy	instrumentalness
0	60.0	0.836000	0.726	214547.0	0.1770	0.000002
1	63.0	0.003840	0.635	190448.0	0.9080	0.000400
2	59.0	0.000075	0.352	456320.0	0.9560	0.020300
3	54.0	0.944000	0.488	352280.0	0.3260	0.015700
4	55.0	0.245000	0.667	273693.0	0.6470	0.000297
...	...	...	...	...	...	...
995	57.0	0.977000	0.194	208040.0	0.0429	0.949000
996	56.0	0.005790	0.939	144453.0	0.3730	0.000000
997	54.0	0.016100	0.739	238339.0	0.5390	0.000000
998	62.0	0.326000	0.515	266707.0	0.5050	0.000000
999	42.0	0.029500	0.291	194679.0	0.5900	0.002270

1000 rows × 6 columns

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 23)
     knn = KNeighborsClassifier(n_neighbors = 5)
     knn.fit(X_train, y_train)
```

[ ]:


People
Raise
jupyter Class_01_Classsification Last Checkpoint: 50 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Python 3 (ipykernel)
995 57.0 0.972000 0.793 208040.0 0.0529 0.979000
996 56.0 0.005790 0.939 144453.0 0.3730 0.000000
997 54.0 0.016100 0.739 238339.0 0.5390 0.000000
998 62.0 0.326000 0.515 286707.0 0.5050 0.000000
999 42.0 0.029500 0.291 194679.0 0.5900 0.002270
1000 rows × 6 columns
[29] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 23)
knn = KNeighborsClassifier(n_neighbors = 9)
knn.fit(X_train, y_train)
[29] KNeighborsClassifier
KNeighborsClassifier(n_neighbors=9)
[30] knn.score(X_train, y_train)
[30] 0.68
[31] knn.score(X_test, y_test)
[31] 0.605
[ ]:

jupyter Class_01_Classsification Last Checkpoint: 51 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

...	...	...	...	...	...	...
995	57.0	0.047000	0.194	208040.0	0.0529	0.047000
996	56.0	0.005790	0.939	144453.0	0.3730	0.000000
997	54.0	0.016100	0.739	238339.0	0.5390	0.000000
998	62.0	0.326000	0.515	286707.0	0.5050	0.000000
999	42.0	0.029500	0.297	194679.0	0.5900	0.002270

1000 rows × 6 columns

[32...]
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 23)
knn = KNeighborsClassifier(n_neighbors = 11)
knn.fit(X_train, y_train)
```

[32...]
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=11)
```

[33...]
```
knn.score(X_train, y_train)
```

[33...]
```
0.66125
```

[34...]
```
knn.score(X_test, y_test)
```

[34...]
```
0.585
```

[ ]:

996	56.0	0.005790	0.939	144453.0	0.5790	0.000000
997	54.0	0.016100	0.739	238339.0	0.5390	0.000000
998	62.0	0.326000	0.515	286707.0	0.5050	0.000000
999	42.0	0.029500	0.291	194679.0	0.5980	0.002270

1000 rows × 6 columns

[32...
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando
knn = KNeighborsClassifier(n_neighbors = 11)
knn.fit(X_train, y_train)
```

[32...
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=11)
```

[33...
```
knn.score(X_train, y_train)
```

[33...
```
0.66125
```

[34...
```
knn.score(X_test, y_test)
```

[34...
```
0.585
```

[ ]:
```
# choosing k

neighbors = np.arange(1, 35)
```

1000 rows × 6 columns

```
[32...] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 23)
        knn = KNeighborsClassifier(n_neighbors = 11)
        knn.fit(X_train, y_train)
```

```
[32...] KNeighborsClassifier
        KNeighborsClassifier(n_neighbors=11)
```

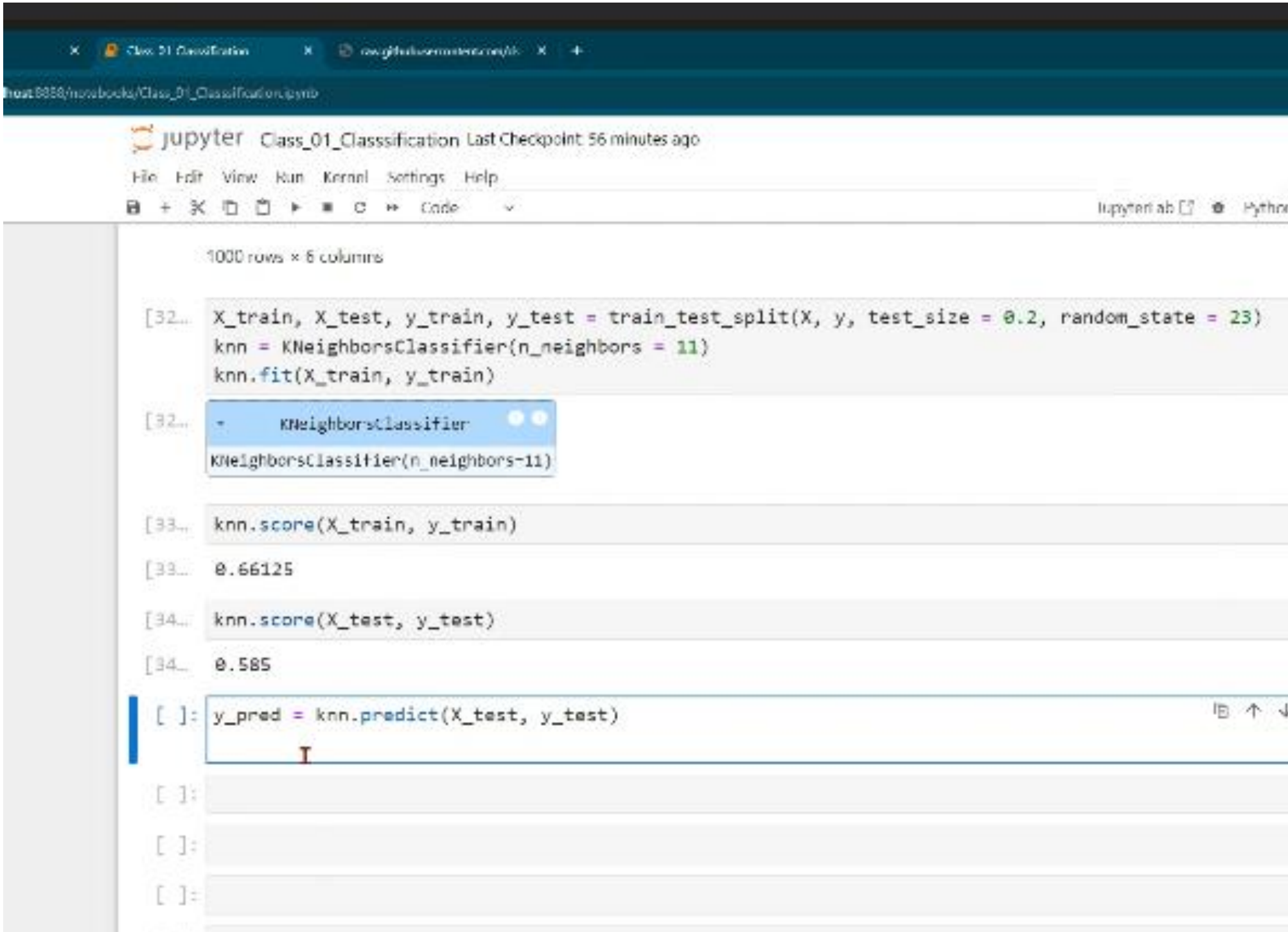
```
[33...] knn.score(X_train, y_train)
```

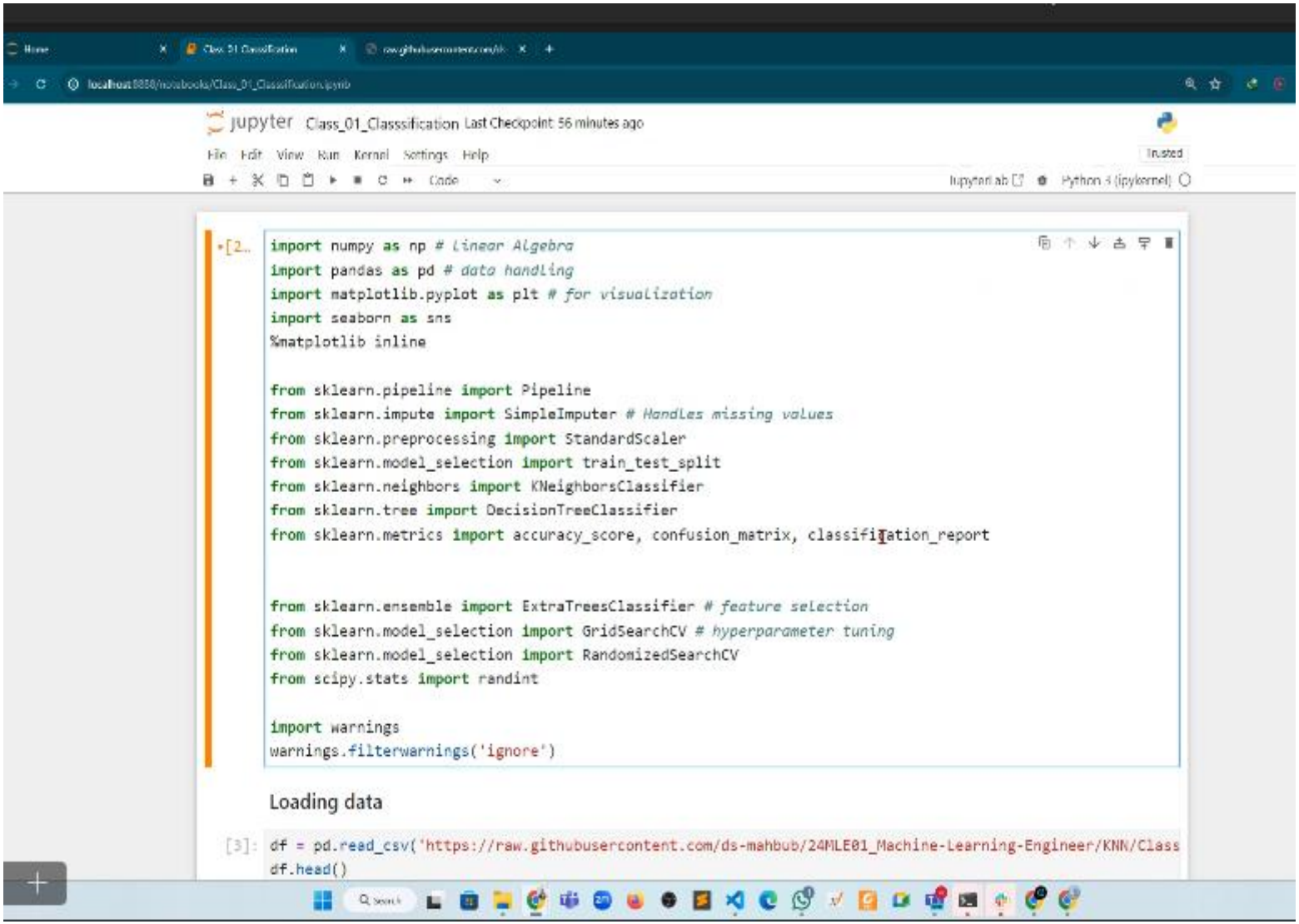
```
[33...] 0.66125
```

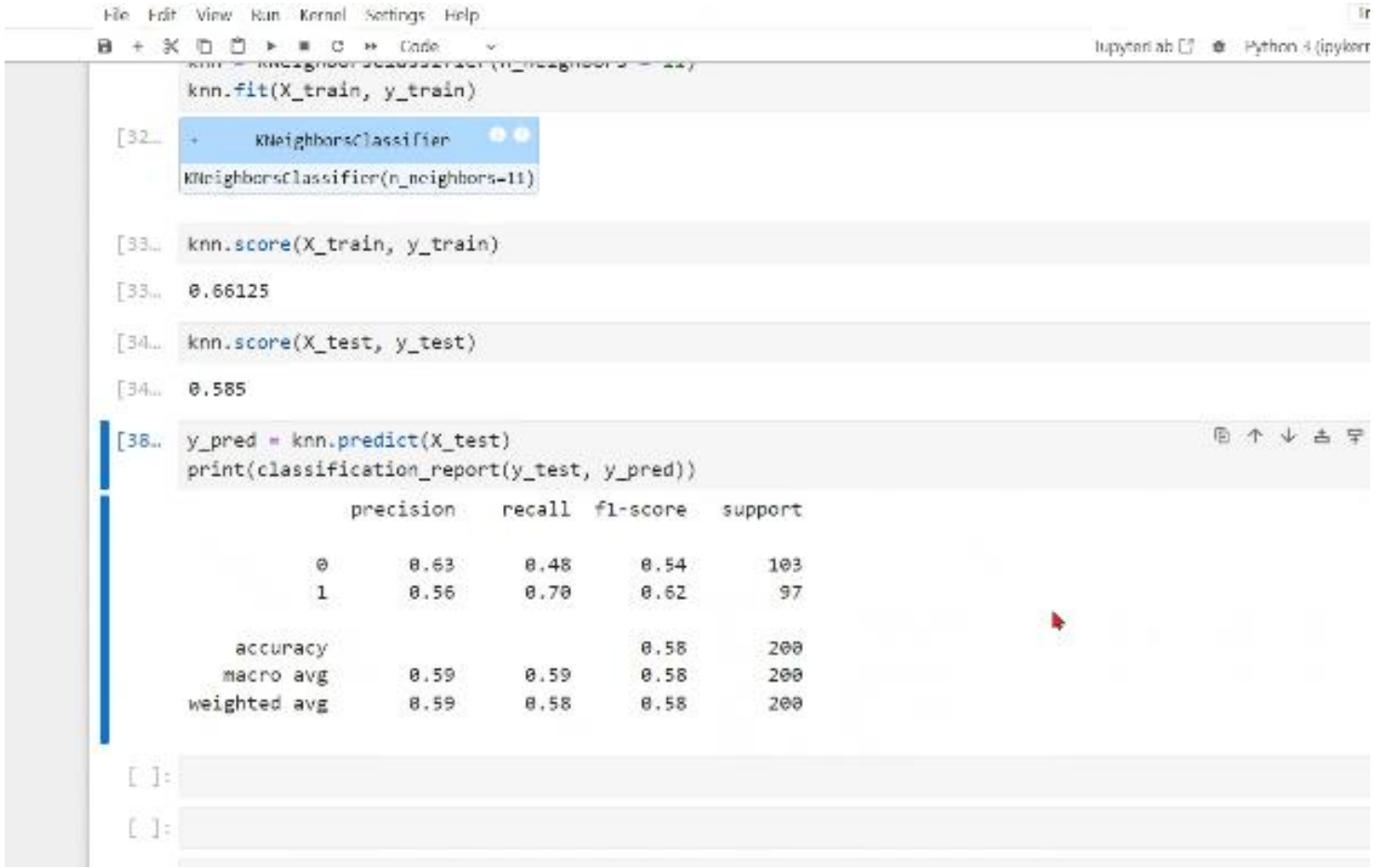
```
[34...] knn.score(X_test, y_test)
```

```
[34...] 0.585
```

```
[ ]: # choosing k

neighbors = np.arange(3, 35)

train_accuracies = {}
test_accuracies = {}
```


jupyter Class_01_Classsification Last Checkpoint 56 minutes ago
File Edit View Run Kernel Settings Help
Code
1000 rows × 6 columns
[32... X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 23)
knn = KNeighborsClassifier(n_neighbors = 11)
knn.fit(X_train, y_train)
[32... KNeighborsClassifier
KNeighborsClassifier(n_neighbors=11)
[33... knn.score(X_train, y_train)
[33... 0.66125
[34... knn.score(X_test, y_test)
[34... 0.585
[ ]: y_pred = knn.predict(X_test, y_test)
[ ]:
[ ]:
[ ]:


jupyter Class_01_Classsification Last Checkpoint: 56 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[2]:
import numpy as np # Linear Algebra
import pandas as pd # data handling
import matplotlib.pyplot as plt # for visualization
import seaborn as sns
%matplotlib inline
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer # Handles missing values
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.ensemble import ExtraTreesClassifier # feature selection
from sklearn.model_selection import GridSearchCV # hyperparameter tuning
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
import warnings
warnings.filterwarnings('ignore')
Loading data
[3]:
df = pd.read_csv('https://raw.githubusercontent.com/ds-mahbub/24MLE01_Machine-Learning-Engineer/KNN/Class
df.head()


File Edit View Run Kernel Settings Help
Code
knn.fit(X_train, y_train)
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=11)
knn.score(X_train, y_train)
0.66125
knn.score(X_test, y_test)
0.585
y_pred = knn.predict(X_test)
print(classification_report(y_test, y_pred))
precision recall f1-score support
0 0.63 0.48 0.54 103
1 0.56 0.70 0.62 97
accuracy 0.58 200
macro avg 0.59 0.59 0.58 200
weighted avg 0.59 0.58 0.58 200

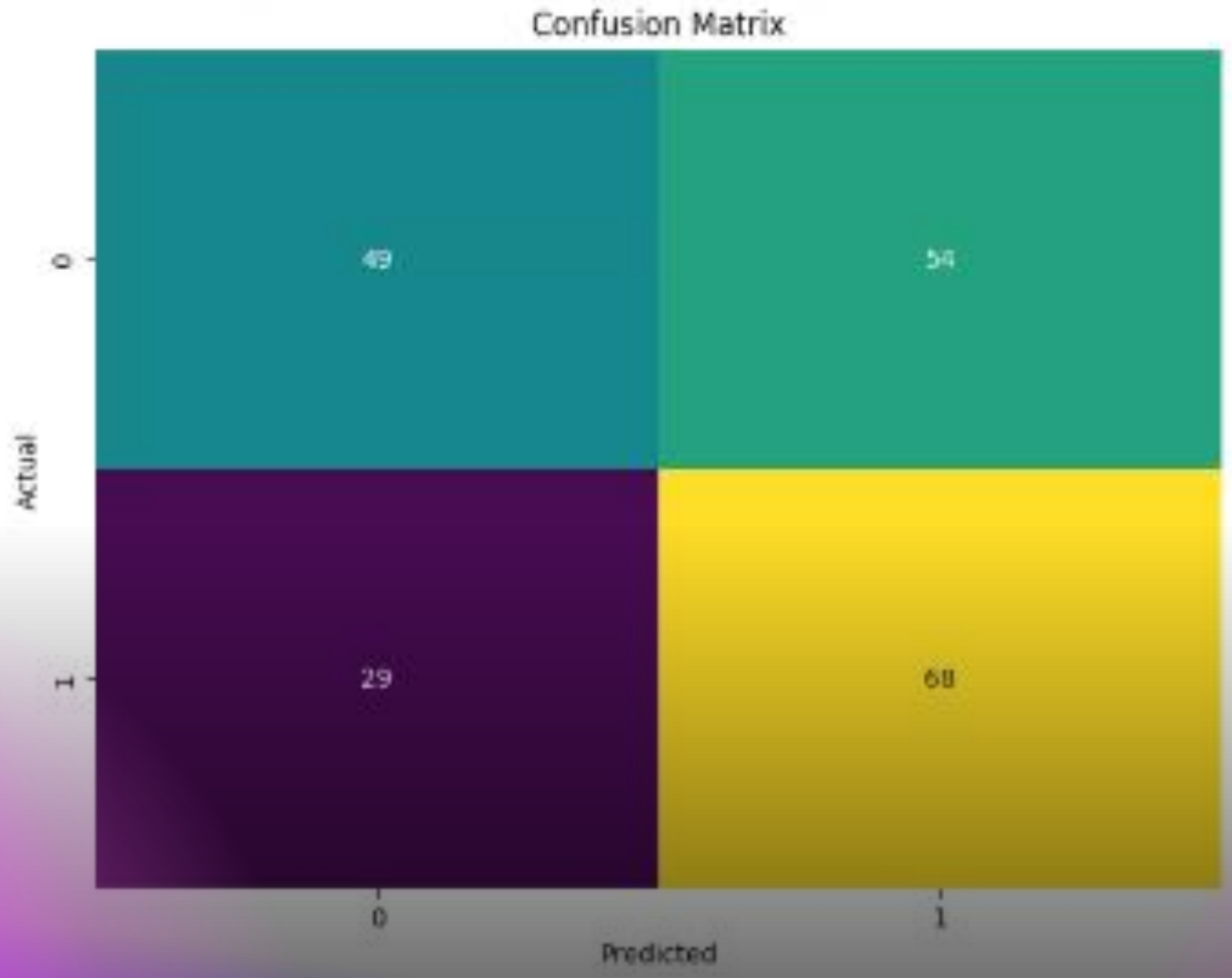
1	0.56	0.70	0.62	97
accuracy			0.58	200
macro avg	0.59	0.59	0.58	200
weighted avg	0.59	0.58	0.58	200

[3]:
```
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize = (8, 6))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = 'viridis', cbar = False)
plt.xlabel('Predicte')
plt.show()
```

File Edit View Run Kernel Settings Help

Code

```
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

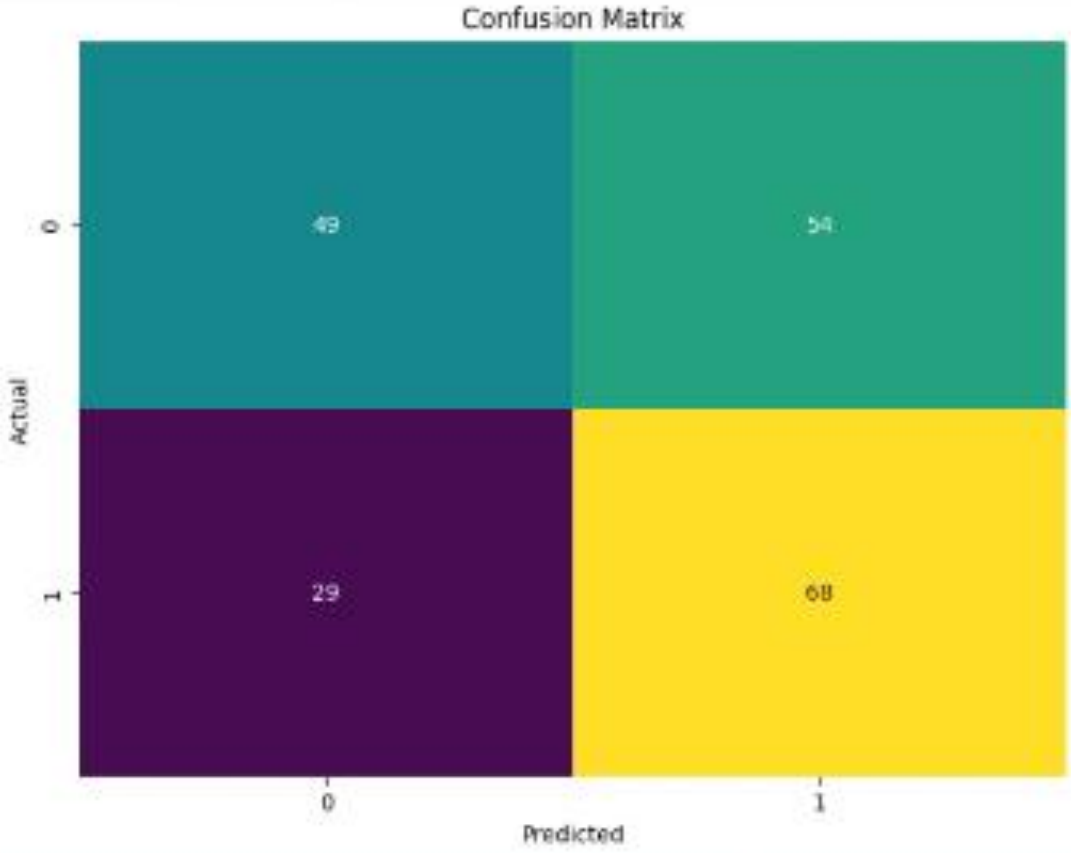


TP: Actual & Predicted both are positive

File Edit View Run Kernel Settings Help

Markdown

```
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

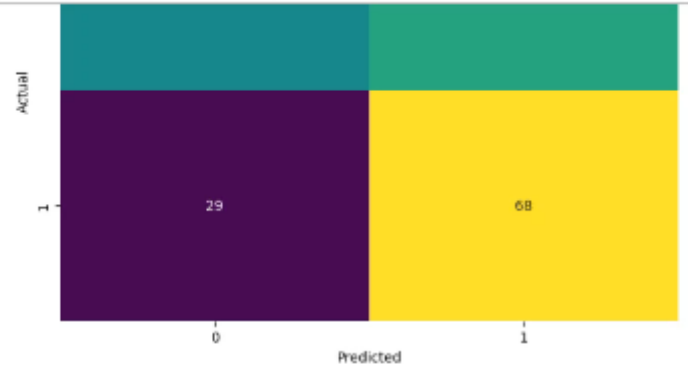


TP: Actual & Predicted both are positive --> 68

F

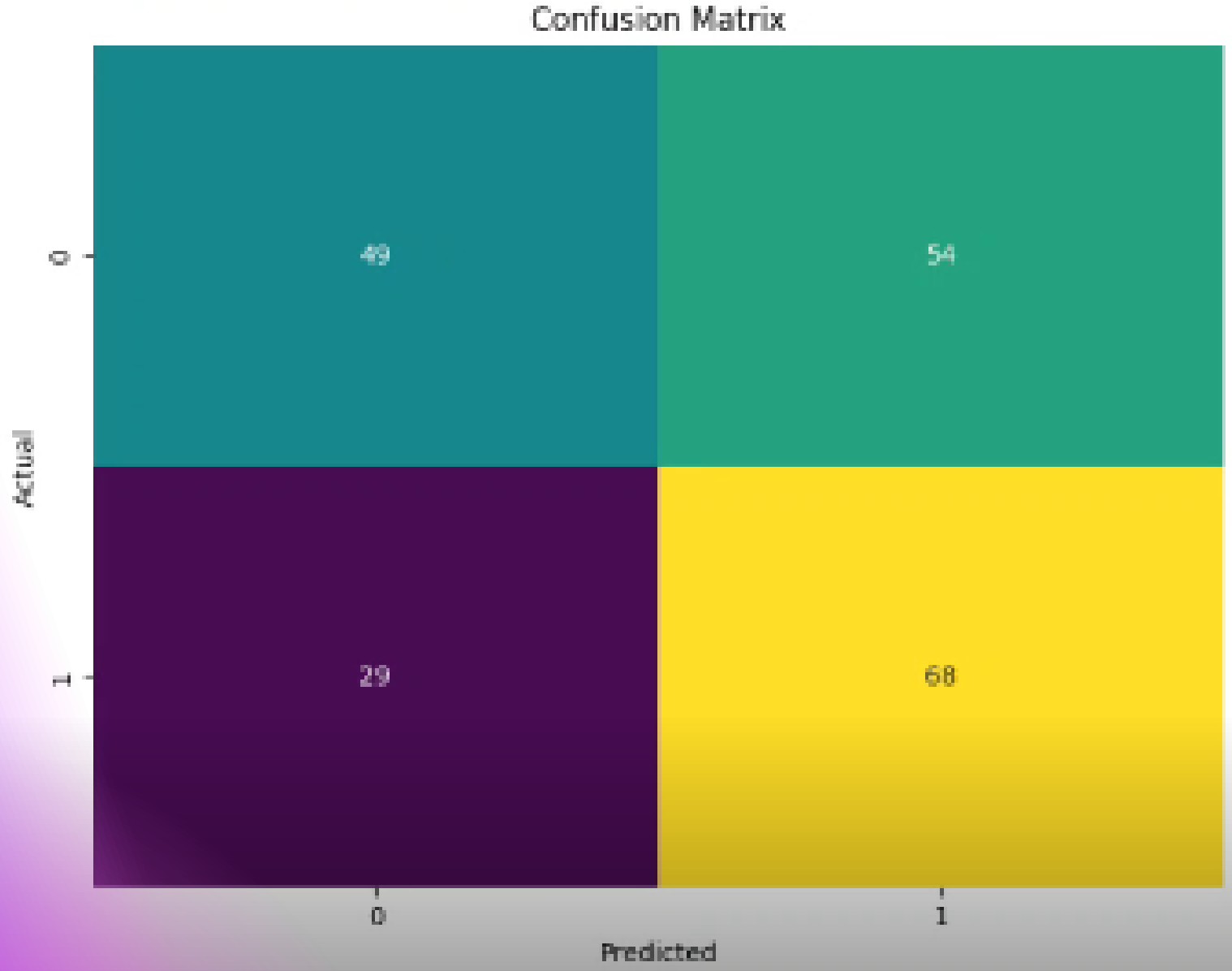
File Edit View Run Kernel Settings Help

Markdown

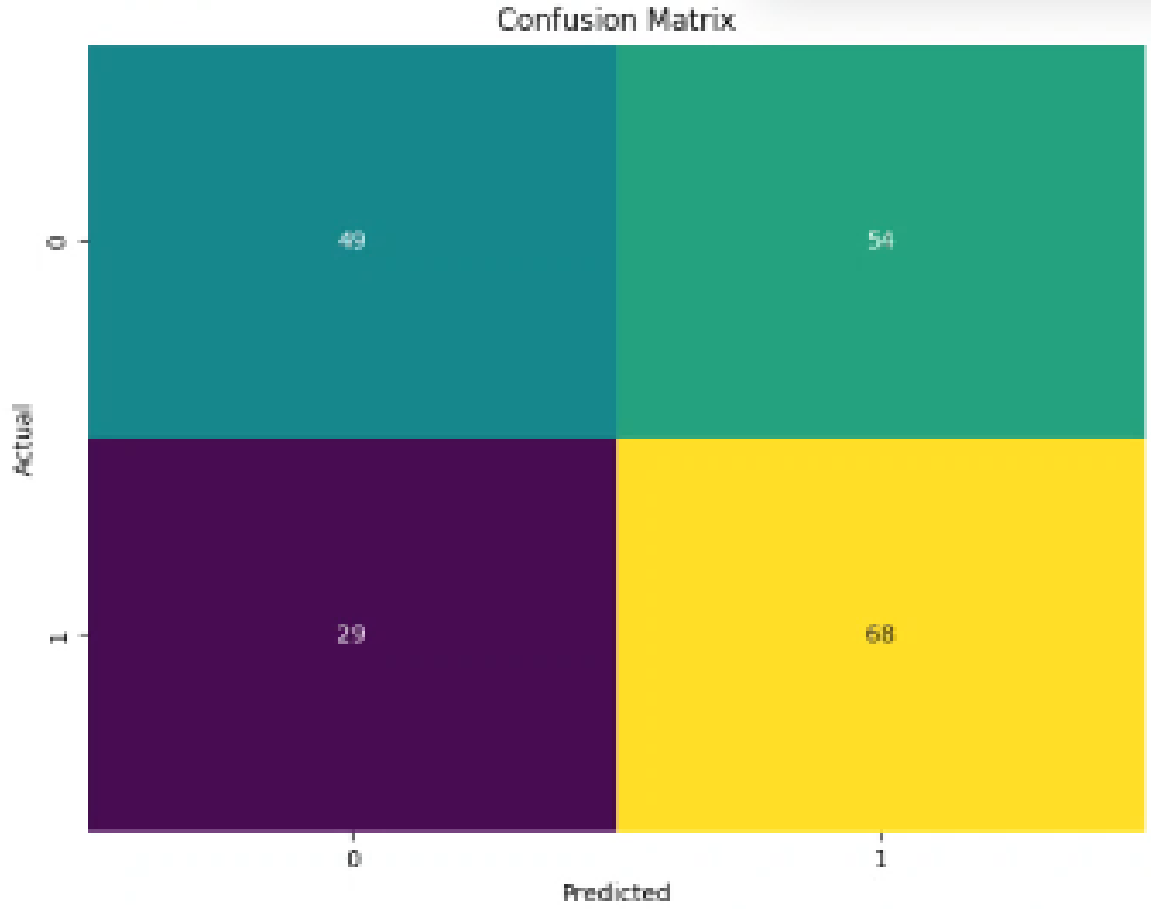


TP: Actual & Predicted both are positive ---> 68
FP: Actual Negative, Model predicted as positive

[ ]:
[ ]:
[ ]:
[ ]:
[ ]:



TP: Actual & Predicted both are positive ---> 68
FP: Actual Negative, Model predicted as positive ---> 54
TN: Actual and Predicted both are negative -----> 49




```
TP: Actual & Predicted both are positive ---> 68
FP: Actual Negative, Model predicted as positive ---> 54
TN: Actual and Predicted both are negative -----> 49
FN: Actual Positive but predicted as Neg
```

- TP: Actual & Predicted both are positive ---> 68
- FP: Actual Negative, Model predicted as positive ---> 54
- TN: Actual and Predicted both are negative -----> 49
- FN: Actual Positive but predicted as Negative ----> 29

[ ]:
```
TP = 68
FP = 54
TN = 49
FN = 29

precision = TP / (TP + FP)
recall = TP / (TP + FN)
f1_score = 2 * (precision * recall) / (precision + recall)
```

[ ]:

- TN: Actual and Predicted both are negative -----> 49
- FN: Actual Positive but predicted as Negative ----> 29

[41]
```
TP = 68
FP = 54
TN = 49
FN = 29

precision = TP / (TP + FP)
recall = TP / (TP + FN)
f1_score = 2 * (precision * recall) / (preci

print(f'Precision: {precision}\nRecall: {reca
```

```
Precision: 0.5573770491803278
Recall: 0.7010309278350515
f1_score: 0.6210045662100456
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```
# choosing k

neighbors = np.arange(3, 35)

train_accuracies = {}
```

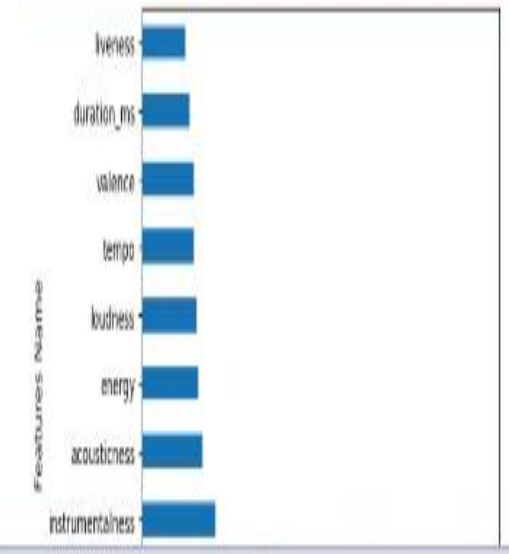
Data Preprocessing

```
[13] X = df.drop('genre', axis = 1) # Features
     y = df['genre']
```

```
[14] feat_selection = ExtraTreesClassifier()
     feat_selection.fit(X, y)
```

```
[14] ExtraTreesClassifier()
```

```
[17] feat_importances = pd.Series(feat_selection.feature_importances_, index = X.columns)
     feat_importances.nlargest(len(df.columns)).plot(kind = 'barh')
     plt.xlabel('Feature Importances')
     plt.ylabel('Features Name')
     plt.show()
```



```
f1_score = 2 * (precision * recall) / (precision + recall)

print(f'Precision: {precision}\nRecall: {recall}\nf1_score: {f1_score}')
```

```
Precision: 0.5573770491803278
Recall: 0.7010309278350515
f1_score: 0.6210045662100456
```

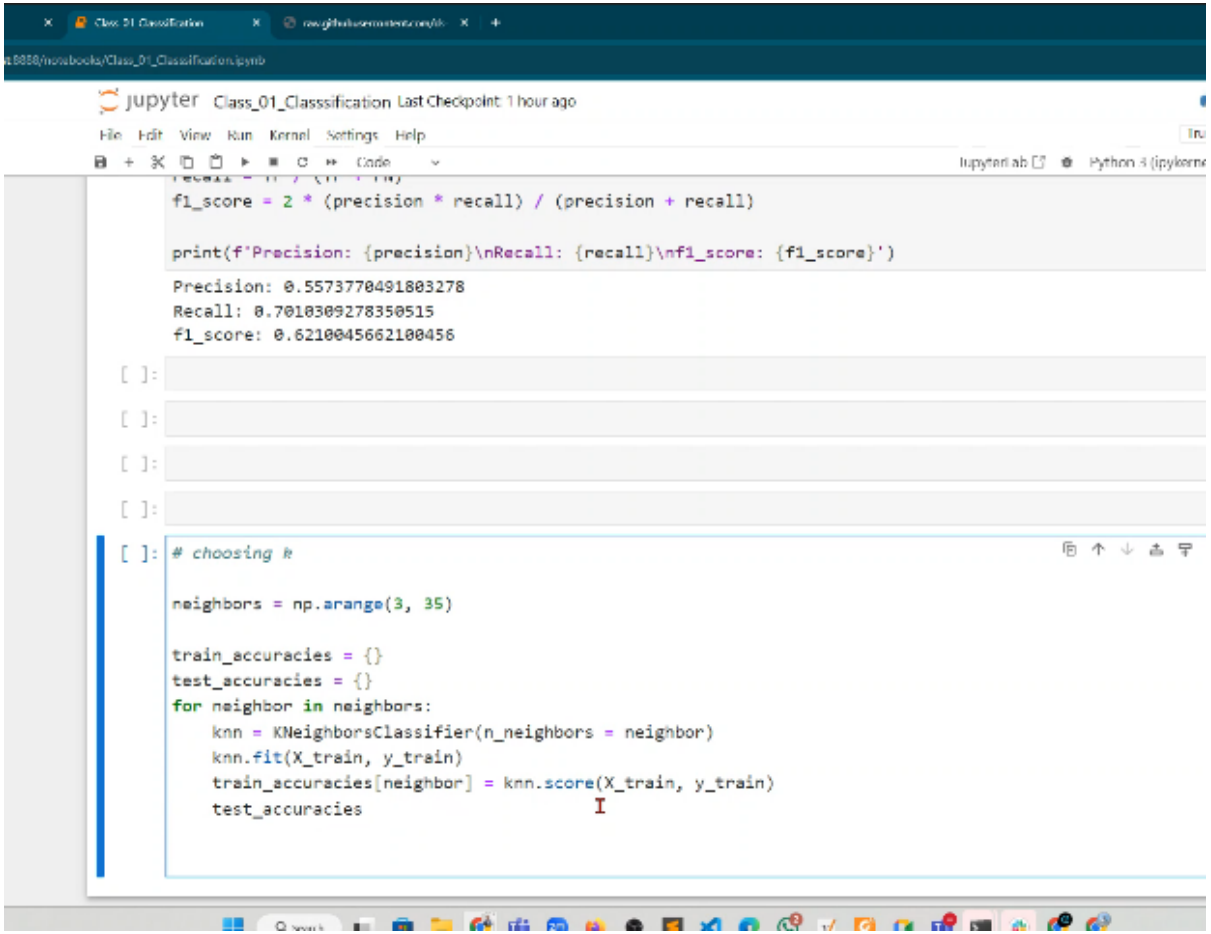
[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```
# choosing k

neighbors = np.arange(3, 35)

train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies
```


jupyter Class_01_Classification Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
f1_score = 2 * (precision * recall) / (precision + recall)
print(f'Precision: {precision}\nRecall: {recall}\nf1_score: {f1_score}')
Precision: 0.5573770491803278
Recall: 0.7010309278350515
f1_score: 0.6210045662100456
# choosing k
neighbors = np.arange(3, 35)
train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
knn = KNeighborsClassifier(n_neighbors = neighbor)
knn.fit(X_train, y_train)
train_accuracies[neighbor] = knn.score(X_train, y_train)
test_accuracies

```
f1_score: 0.6210045662100456
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]: # choosing k

neighbors = np.arange(3, 35)

train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

print(f'Train Accuracies: \n{train_accuracies}\n\nTest Accuracies: \n{test_accuracies}')
```

```
train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

print(f'Train Accuracies: \n{train_accuracies}\n\nTest Accuracies: \n{test_accuracies}')
```

```
Train Accuracies:
{3: 0.75375, 4: 0.71, 5: 0.70625, 6: 0.6775, 7: 0.68125, 8: 0.6575, 9: 0.68, 10: 0.65625, 11: 0.66125, 12: 0.64625, 13: 0.65, 14: 0.63875, 15: 0.63625, 16: 0.62, 17: 0.63375, 18: 0.63125, 19: 0.62875, 20: 0.6275, 21: 0.62375, 22: 0.6275, 23: 0.6275, 24: 0.61875, 25: 0.62875, 26: 0.62375, 27: 0.6275, 28: 0.6275, 29: 0.62375, 30: 0.62625, 31: 0.6275, 32: 0.6325, 33: 0.63125, 34: 0.63125}

Test Accuracies:
{3: 0.555, 4: 0.54, 5: 0.55, 6: 0.565, 7: 0.59, 8: 0.575, 9: 0.605, 10: 0.595, 11: 0.585, 12: 0.58, 13: 0.59, 14: 0.6, 15: 0.56, 16: 0.57, 17: 0.555, 18: 0.545, 19: 0.58, 20: 0.55, 21: 0.575, 22: 0.555, 23: 0.59, 24: 0.595, 25: 0.615, 26: 0.595, 27: 0.61, 28: 0.615, 29: 0.595, 30: 0.59, 31: 0.595, 32: 0.61, 33: 0.62, 34: 0.61}
```

[ ]:

jupyter Class_01_Classsification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Trusted

Code

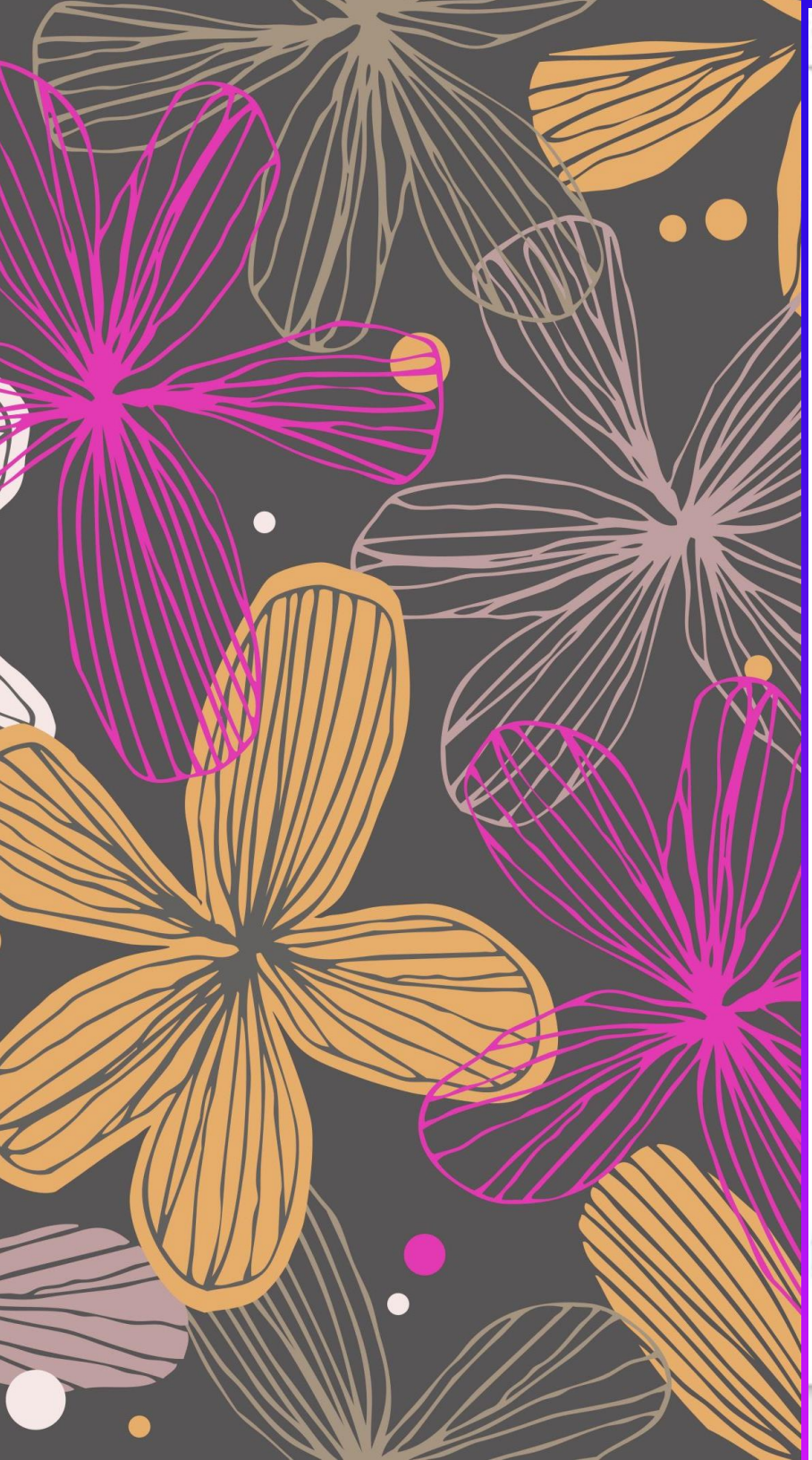
JupyterLab

Python 3 (ipykernel)

```
train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

print(f'Train Accuracies: \n{train_accuracies}\n\nTest Accuracies: \n{test_accuracies}')
```

```
Train Accuracies:
{3: 0.75375, 4: 0.71, 5: 0.70625, 6: 0.6775, 7: 0.68125, 8: 0.6575, 9: 0.68, 10: 0.65625, 11: 0.66125, 1
2: 0.64625, 13: 0.65, 14: 0.63875, 15: 0.63625, 16: 0.62, 17: 0.63375, 18: 0.63125, 19: 0.62875, 20: 0.6
275, 21: 0.62375, 22: 0.6275, 23: 0.6275, 24: 0.61875, 25: 0.62875, 26: 0.62375, 27: 0.6275, 28: 0.6275,
29: 0.62375, 30: 0.62625, 31: 0.6275, 32: 0.6325, 33: 0.63125, 34: 0.63125}

Test Accuracies:
{3: 0.555, 4: 0.54, 5: 0.55, 6: 0.565, 7: 0.59, 8: 0.575, 9: 0.605, 10: 0.595, 11: 0.585, 12: 0.58, 13:
0.59, 14: 0.6, 15: 0.56, 16: 0.57, 17: 0.555, 18: 0.545, 19: 0.58, 20: 0.55, 21: 0.575, 22: 0.555, 23:
0.59, 24: 0.595, 25: 0.615, 26: 0.595, 27: 0.61, 28: 0.615, 29: 0.595, 30: 0.59, 31: 0.595, 32: 0.61, 3
3: 0.62, 34: 0.61}
```

[ ]:



```
train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

print(f'Train Accuracies: \n{train_accuracies}\n\nTest Accuracies: \n{test_accuracies}')
```

```
Train Accuracies:
{3: 0.75375, 4: 0.71, 5: 0.70625, 6: 0.6775, 7: 0.68125, 8: 0.6575, 9: 0.68, 10: 0.65625, 11: 0.661
2: 0.64625, 13: 0.65, 14: 0.63875, 15: 0.63625, 16: 0.62, 17: 0.63375, 18: 0.63125, 19: 0.62875, 20
275, 21: 0.62375, 22: 0.6275, 23: 0.6275, 24: 0.61875, 25: 0.62875, 26: 0.62375, 27: 0.6275, 28: 0.
29: 0.62375, 30: 0.62625, 31: 0.6275, 32: 0.6325, 33: 0.63125, 34: 0.63125}

Test Accuracies:
{3: 0.555, 4: 0.54, 5: 0.55, 6: 0.565, 7: 0.59, 8: 0.575, 9: 0.605, 10: 0.595, 11: 0.585, 12: 0.58,
0.59, 14: 0.6, 15: 0.56, 16: 0.57, 17: 0.555, 18: 0.545, 19: 0.58, 20: 0.55, 21: 0.575, 22: 0.555,
0.59, 24: 0.595, 25: 0.615, 26: 0.595, 27: 0.61, 28: 0.615, 29: 0.595, 30: 0.59, 31: 0.595, 32: 0.6
3: 0.62, 34: 0.61}
```

```
[ ]: plt.title('Varying numbers of neighbors')
     plt.plot(neighbors, train_accuracies.values(), label = 'Training Accuracy')
```

```
train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

print(f'Train Accuracies: \n{train_accuracies}\n\nTest Accuracies: \n{test_accuracies}')
```

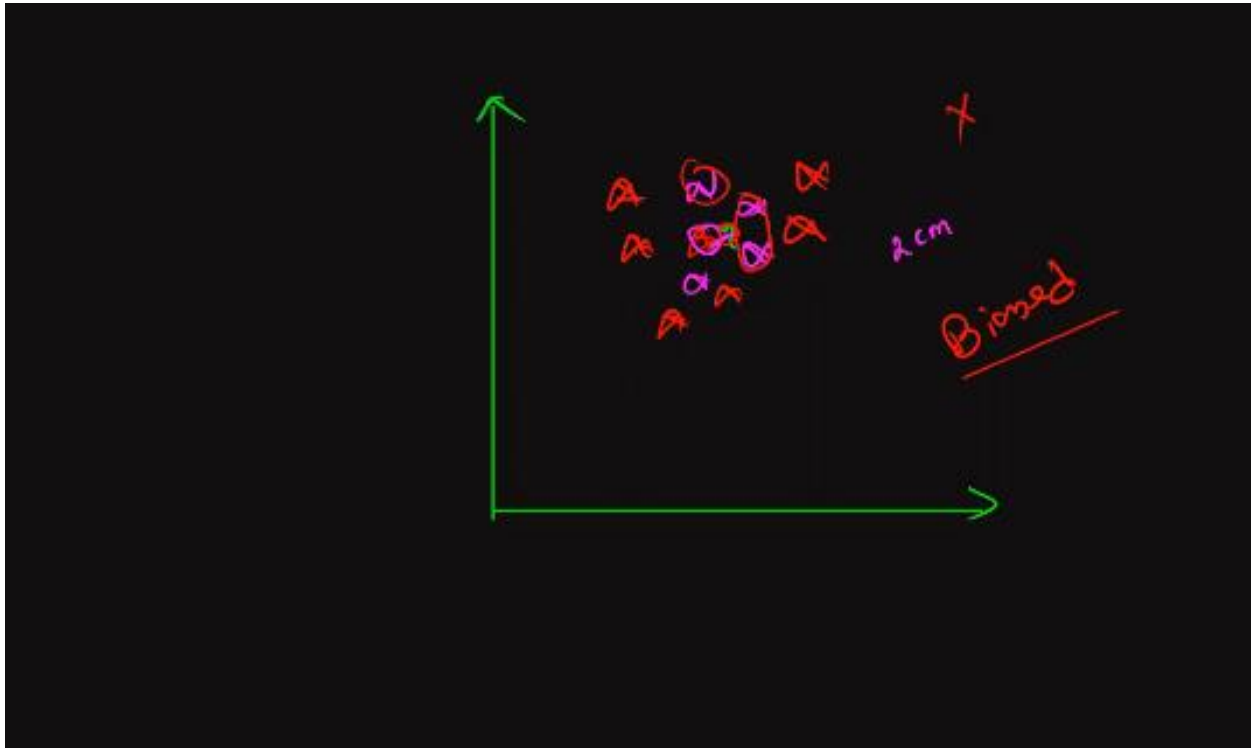
```
Train Accuracies:
{3: 0.75375, 4: 0.71, 5: 0.70625, 6: 0.6775, 7: 0.68125, 8: 0.6575, 9: 0.68, 10: 0.65625, 11: 0.661
2: 0.64625, 13: 0.65, 14: 0.63875, 15: 0.63625, 16: 0.62, 17: 0.63375, 18: 0.63125, 19: 0.62875, 20
275, 21: 0.62375, 22: 0.6275, 23: 0.6275, 24: 0.61875, 25: 0.62875, 26: 0.62375, 27: 0.6275, 28: 0.
29: 0.62375, 30: 0.62625, 31: 0.6275, 32: 0.6325, 33: 0.63125, 34: 0.63125}

Test Accuracies:
{3: 0.555, 4: 0.54, 5: 0.55, 6: 0.565, 7: 0.59, 8: 0.575, 9: 0.605, 10: 0.595, 11: 0.585, 12: 0.58,
0.59, 14: 0.6, 15: 0.56, 16: 0.57, 17: 0.555, 18: 0.545, 19: 0.58, 20: 0.55, 21: 0.575, 22: 0.555,
0.59, 24: 0.595, 25: 0.615, 26: 0.595, 27: 0.61, 28: 0.615, 29: 0.595, 30: 0.59, 31: 0.595, 32: 0.6
3: 0.62, 34: 0.61}
```

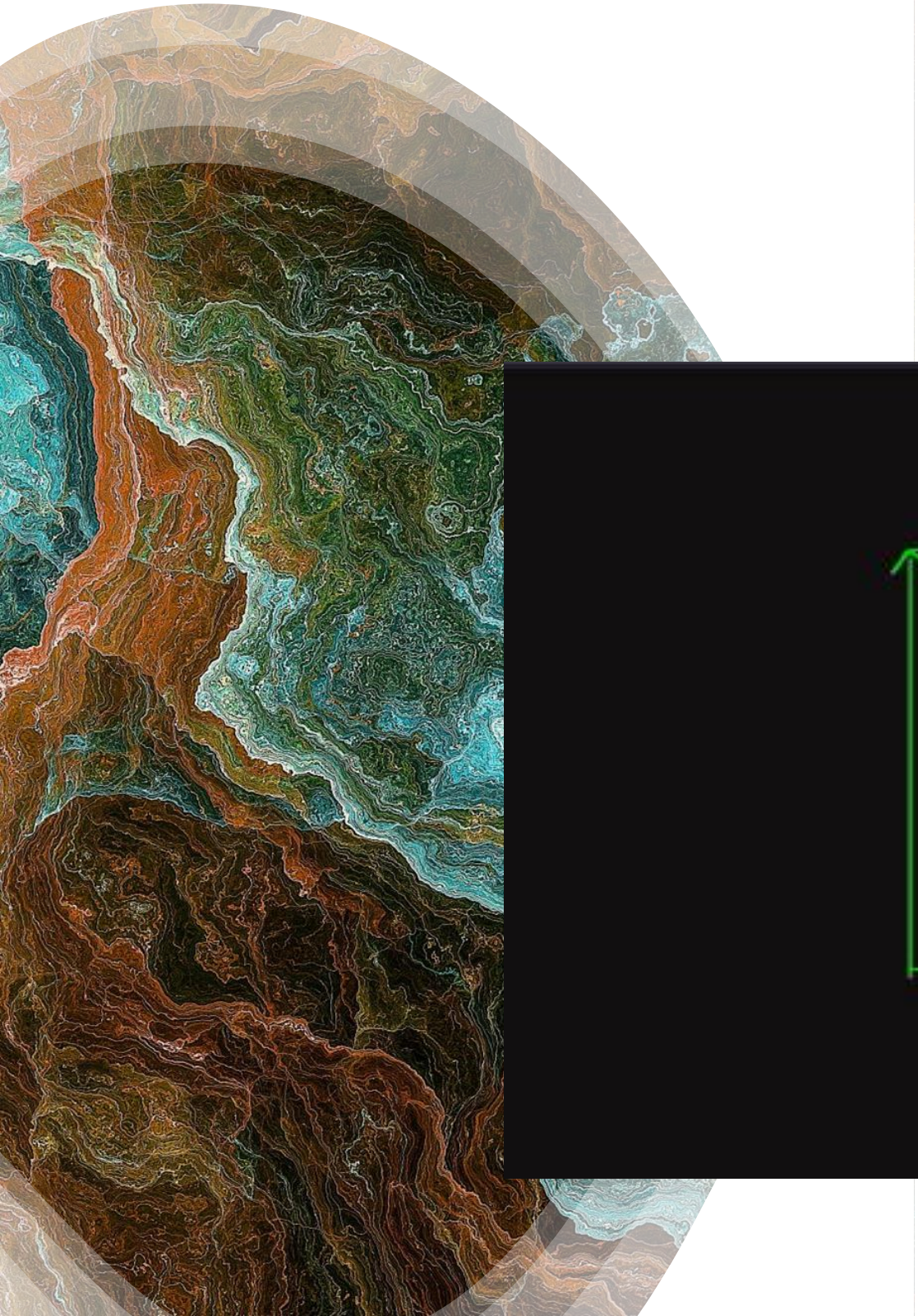
```
[ ]: plt.title('Varying numbers of neighbors')
     plt.plot(neighbors, train_accuracies.values(), label = 'Training Accuracy')
```

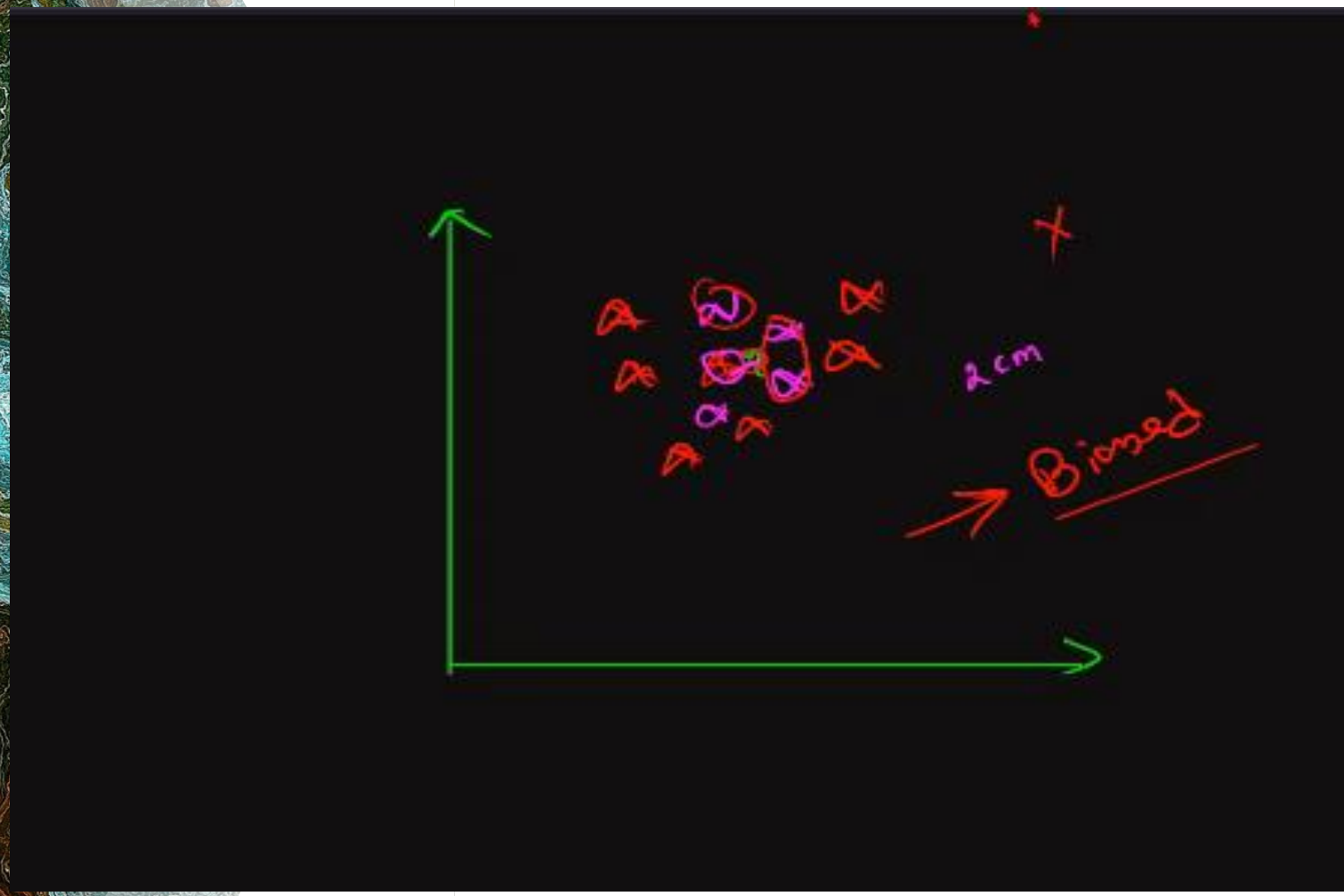
```
train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

print(f'Train Accuracies: \n{train_accuracies}\n\nTest Accuracies: \n{test_accuracies}')
```

```
Train Accuracies:
{3: 0.75375, 4: 0.71, 5: 0.70625, 6: 0.6775, 7: 0.68125, 8: 0.6575, 9: 0.68, 10: 0.65625, 11: 0.661
2: 0.64625, 13: 0.65, 14: 0.63875, 15: 0.63625, 16: 0.62, 17: 0.63375, 18: 0.63125, 19: 0.62875, 20
275, 21: 0.62375, 22: 0.6275, 23: 0.6275, 24: 0.61875, 25: 0.62875, 26: 0.62375, 27: 0.6275, 28: 0.
29: 0.62375, 30: 0.62625, 31: 0.6275, 32: 0.6325, 33: 0.63125, 34: 0.63125}

Test Accuracies:
{3: 0.555, 4: 0.54, 5: 0.55, 6: 0.565, 7: 0.59, 8: 0.575, 9: 0.605, 10: 0.595, 11: 0.585, 12: 0.58,
0.59, 14: 0.6, 15: 0.56, 16: 0.57, 17: 0.555, 18: 0.545, 19: 0.58, 20: 0.55, 21: 0.575, 22: 0.555,
0.59, 24: 0.595, 25: 0.615, 26: 0.595, 27: 0.61, 28: 0.615, 29: 0.595, 30: 0.59, 31: 0.595, 32: 0.6
3: 0.62, 34: 0.61}
```

```
[ ]: plt.title('Varying numbers of neighbors')
     plt.plot(neighbors, train_accuracies.values(), label = 'Training Accuracy')
```


X
2 cm
Biased




2cm
Biased

jupyter Class_01_Classsification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab

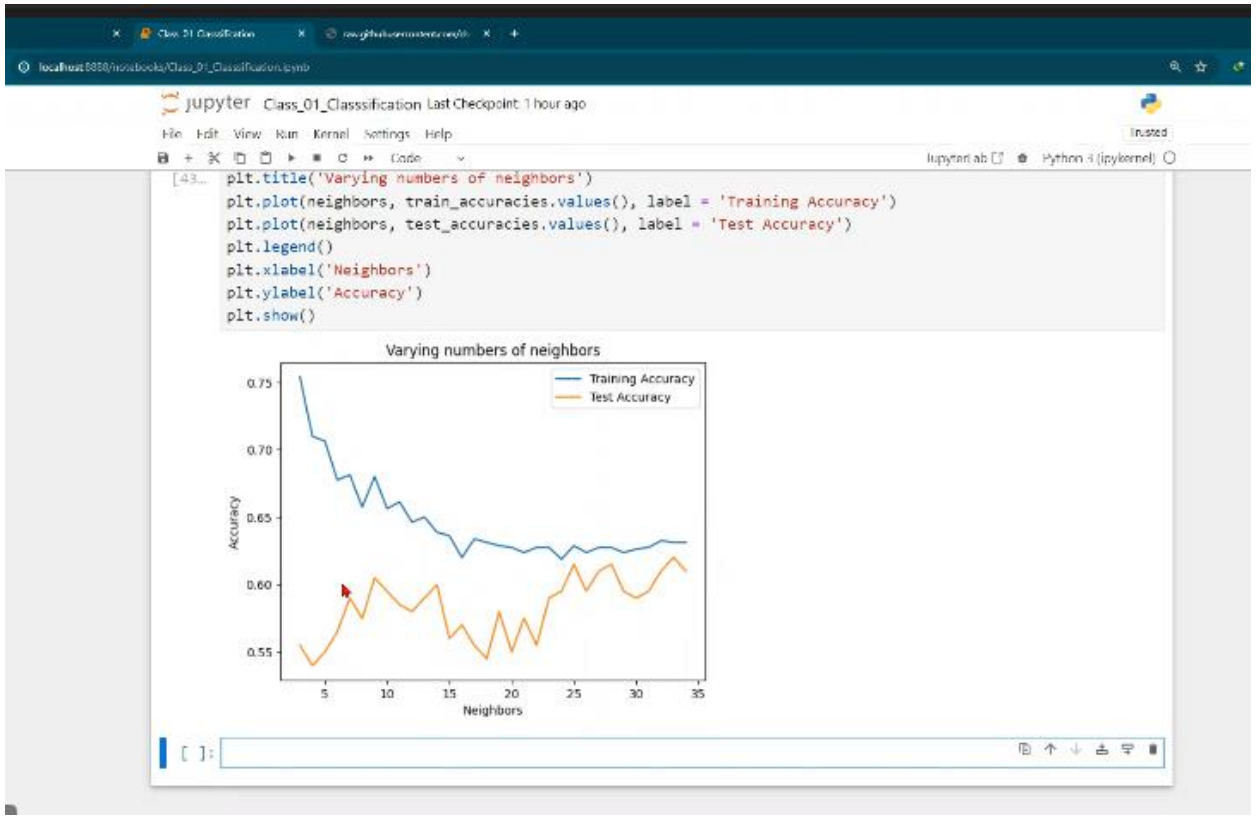
Python 3 (ipykernel)

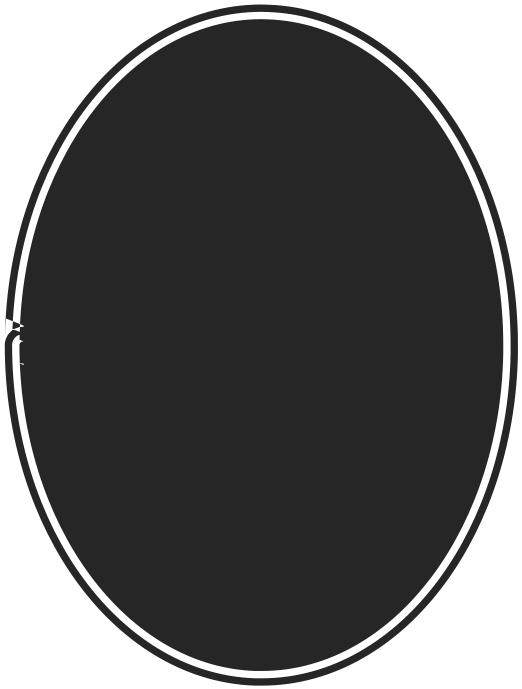
```
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

print(f'Train Accuracies: \n{train_accuracies}\n\nTest Accuracies: \n{test_accuracies}')
```

```
Train Accuracies:
{3: 0.75375, 4: 0.71, 5: 0.70625, 6: 0.6775, 7: 0.68125, 8: 0.6575, 9: 0.68, 10: 0.65625, 11: 0.66125, 1
2: 0.64625, 13: 0.65, 14: 0.63875, 15: 0.63625, 16: 0.62, 17: 0.63375, 18: 0.63125, 19: 0.62875, 20: 0.6
275, 21: 0.62375, 22: 0.6275, 23: 0.6275, 24: 0.61875, 25: 0.62875, 26: 0.62375, 27: 0.6275, 28: 0.6275,
29: 0.62375, 30: 0.62625, 31: 0.6275, 32: 0.6325, 33: 0.63125, 34: 0.63125}

Test Accuracies:
{3: 0.555, 4: 0.54, 5: 0.55, 6: 0.565, 7: 0.59, 8: 0.575, 9: 0.605, 10: 0.595, 11: 0.585, 12: 0.58, 13:
0.59, 14: 0.6, 15: 0.56, 16: 0.57, 17: 0.555, 18: 0.545, 19: 0.58, 20: 0.55, 21: 0.575, 22: 0.555, 23:
0.59, 24: 0.595, 25: 0.615, 26: 0.595, 27: 0.61, 28: 0.615, 29: 0.595, 30: 0.59, 31: 0.595, 32: 0.61, 3
3: 0.62, 34: 0.61}
```

[ ]:

```
plt.title('Varying numbers of neighbors')
plt.plot(neighbors, train_accuracies.values(), label = 'Training Accuracy')
plt.plot(neighbors, test_accuracies.values(), label = 'Test Accuracy')
plt.legend()
plt.xlabel('Neighbors')
plt.ylabel('Accuracy')
plt.
```


jupyter Class_01_Classsification Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Trusted
JupyterLab
Python 3 (ipykernel)
plt.title('Varying numbers of neighbors')
plt.plot(neighbors, train_accuracies.values(), label = 'Training Accuracy')
plt.plot(neighbors, test_accuracies.values(), label = 'Test Accuracy')
plt.legend()
plt.xlabel('Neighbors')
plt.ylabel('Accuracy')
plt.show()
Varying numbers of neighbors
Training Accuracy
Test Accuracy
Accuracy
0.75
0.70
0.65
0.60
0.55
5
10
15
20
25
30
35
Neighbors



Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab

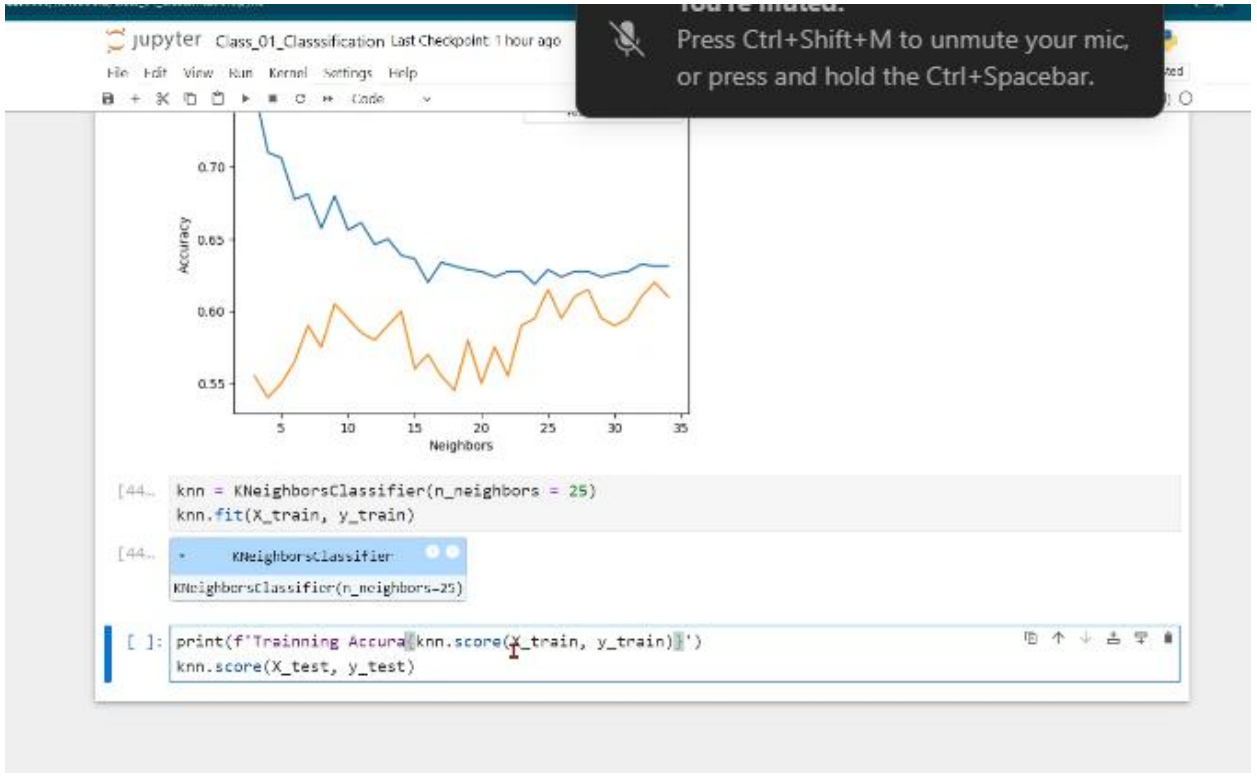
Python 3 (ipykernel)

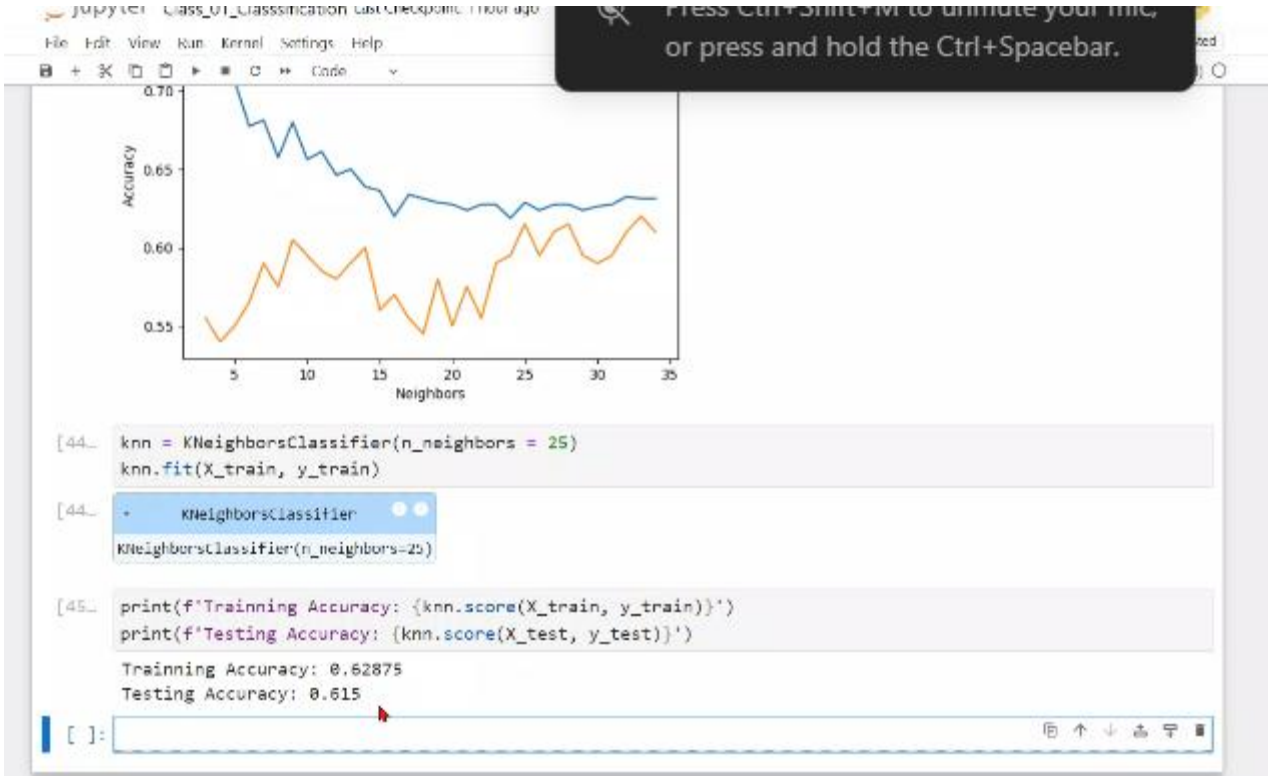
[ ]:

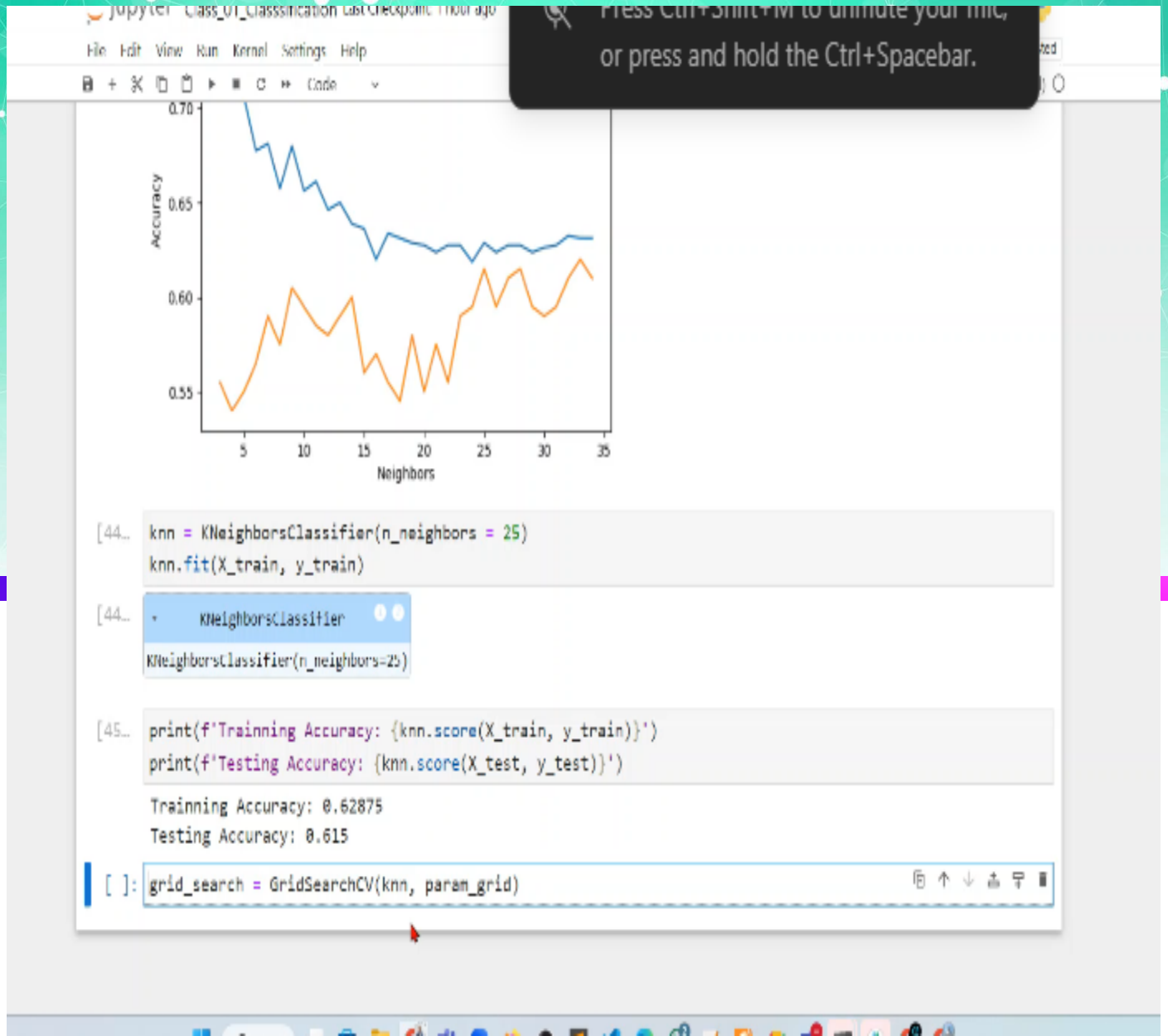
[42]:
```
# choosing k

neighbors = np.arange(3, 35)

train_accuracies = {}
test_accuracies = {}
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

print(f'Train Accuracies: \n{train_accuracies}\n\nTest Accuracies: \n{test_accuracies}')
```

```
Train Accuracies:
{3: 0.75375, 4: 0.71, 5: 0.70625, 6: 0.6775, 7: 0.68125, 8: 0.6575, 9: 0.68, 10: 0.65625, 11: 0.66125, 1
2: 0.64625, 13: 0.65, 14: 0.63875, 15: 0.63625, 16: 0.62, 17: 0.63375, 18: 0.63125, 19: 0.62875, 20: 0.6
275, 21: 0.62375, 22: 0.6275, 23: 0.6275, 24: 0.61875, 25: 0.62875, 26: 0.62375, 27: 0.6275, 28: 0.6275,
29: 0.62375, 30: 0.62625, 31: 0.6275, 32: 0.6325, 33: 0.63125, 34: 0.63125}

Test Accuracies:
{3: 0.555, 4: 0.54, 5: 0.55, 6: 0.565, 7: 0.59, 8: 0.575, 9: 0.605, 10: 0.595, 11: 0.585, 12: 0.58, 13:
0.59, 14: 0.6, 15: 0.56, 16: 0.57, 17: 0.555, 18: 0.545, 19: 0.58, 20: 0.55, 21: 0.575, 22: 0.555, 23:
0.59, 24: 0.595, 25: 0.615, 26: 0.595, 27: 0.61, 28: 0.615, 29: 0.595, 30: 0.59, 31: 0.595, 32: 0.61, 3
3: 0.62, 34: 0.61}
```

[43]:
```
plt.title('Varying numbers of neighbors')
plt.plot(neighbors, train_accuracies.values(), label = 'Training Accuracy')
```


jupyter Class_01_Classsification Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Code
You're muted.
Press Ctrl+Shift+M to unmute your mic,
or press and hold the Ctrl+Spacebar.
Accuracy
0.70
0.65
0.60
0.55
5
10
15
20
25
30
35
Neighbors
[44...
knn = KNeighborsClassifier(n_neighbors = 25)
knn.fit(X_train, y_train)
[44...
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)
[ ]:
print(f'Trainning Accura knn.score(X_train, y_train)}')
knn.score(X_test, y_test)


File Edit View Run Kernel Settings Help
Code
or press and hold the Ctrl+Spacebar.
Accuracy
0.70
0.65
0.60
0.55
5
10
15
20
25
30
35
Neighbors
[44] knn = KNeighborsClassifier(n_neighbors = 25)
knn.fit(X_train, y_train)
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)
[45] print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
Trainning Accuracy: 0.62875
Testing Accuracy: 0.615
[ ]:


File Edit View Run Kernel Settings Help
Code
or press and hold the Ctrl+Spacebar.
Accuracy
0.70
0.65
0.60
0.55
5
10
15
20
25
30
35
Neighbors
[44… knn = KNeighborsClassifier(n_neighbors = 25)
knn.fit(X_train, y_train)
[44… KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)
[45… print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
Trainning Accuracy: 0.62875
Testing Accuracy: 0.615
[ ]: grid_search = GridSearchCV(knn, param_grid)

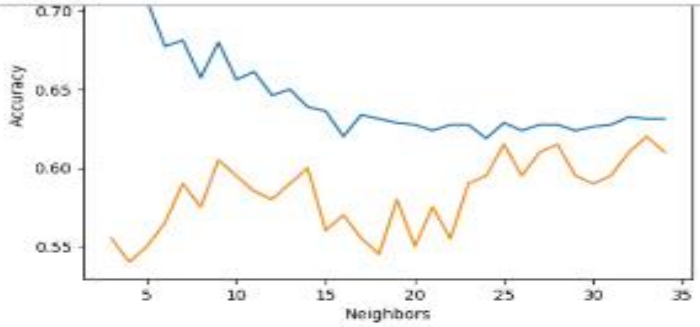
jupyter Class_01_Classsification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)



[44]
```
knn = KNeighborsClassifier(n_neighbors = 25)
knn.fit(X_train, y_train)
```

[44]
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)

[45]
```
print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
```

```
Trainning Accuracy: 0.62875
Testing Accuracy: 0.615
```

[ ]:
```
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27]
}
grid_search = GridSearchCV(knn, param_grid)
```

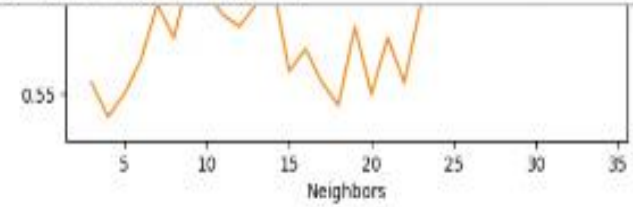
jupyter Class_01_Classsification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)



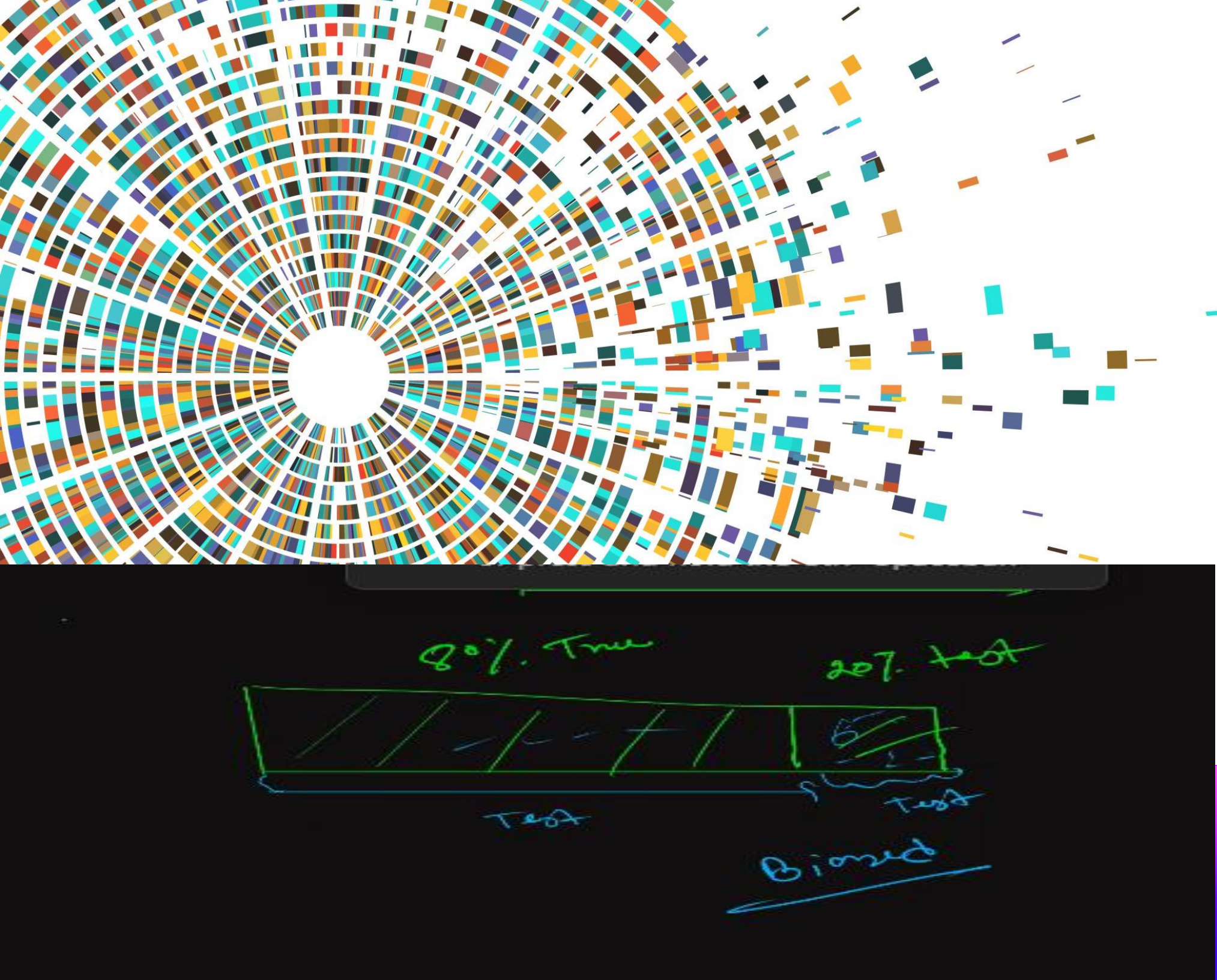
[44]
```
knn = KNeighborsClassifier(n_neighbors = 25)
knn.fit(X_train, y_train)
```

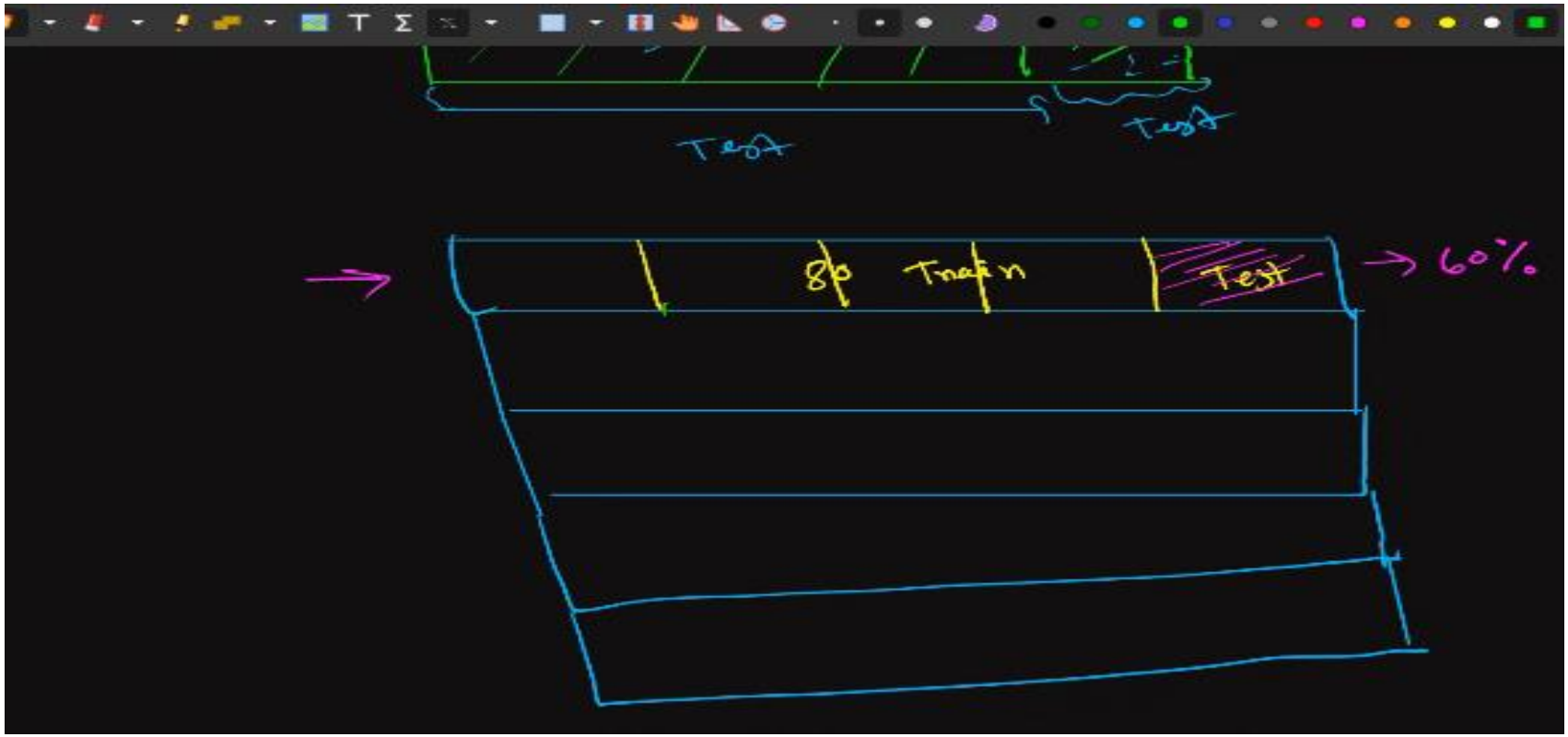
[44]
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)

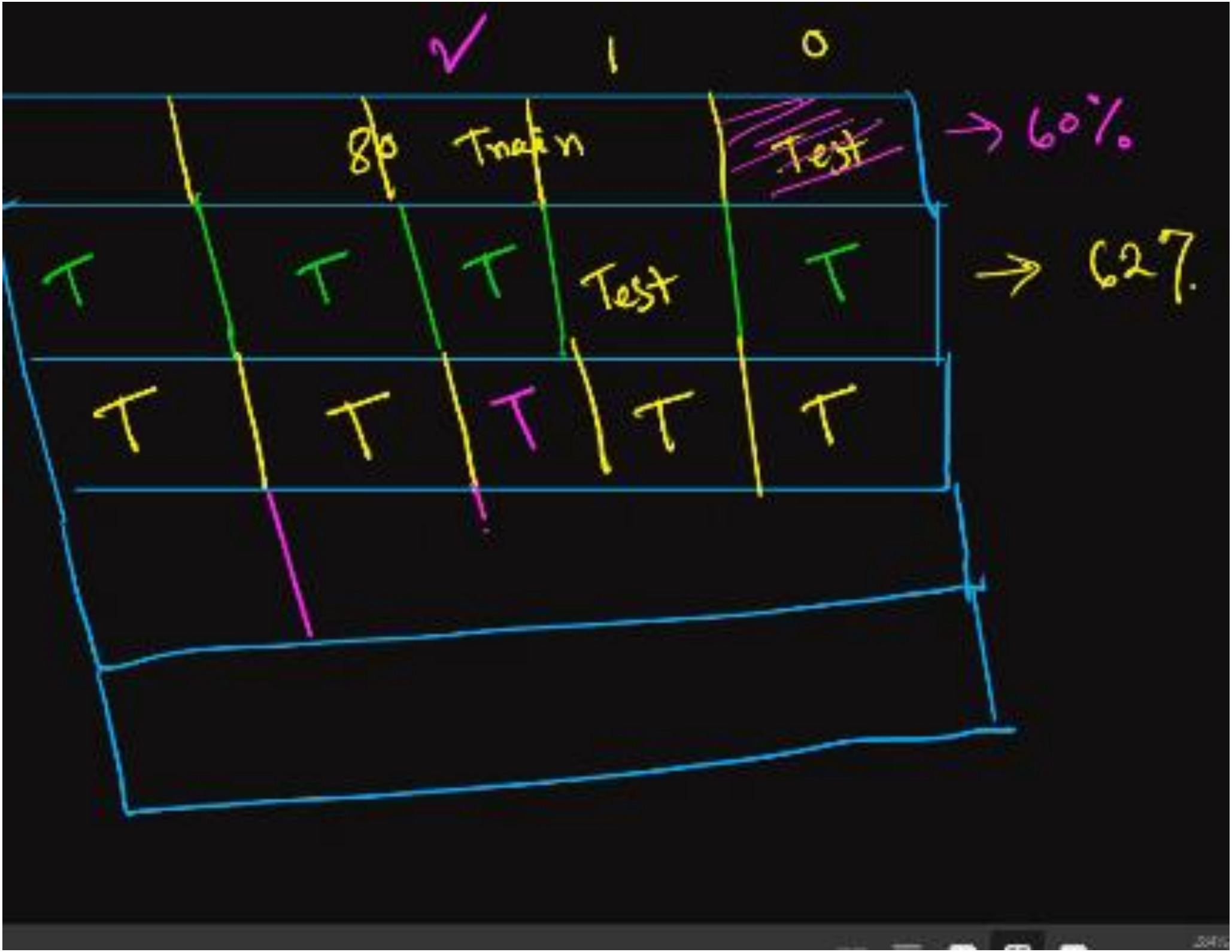
[45]
```
print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
```

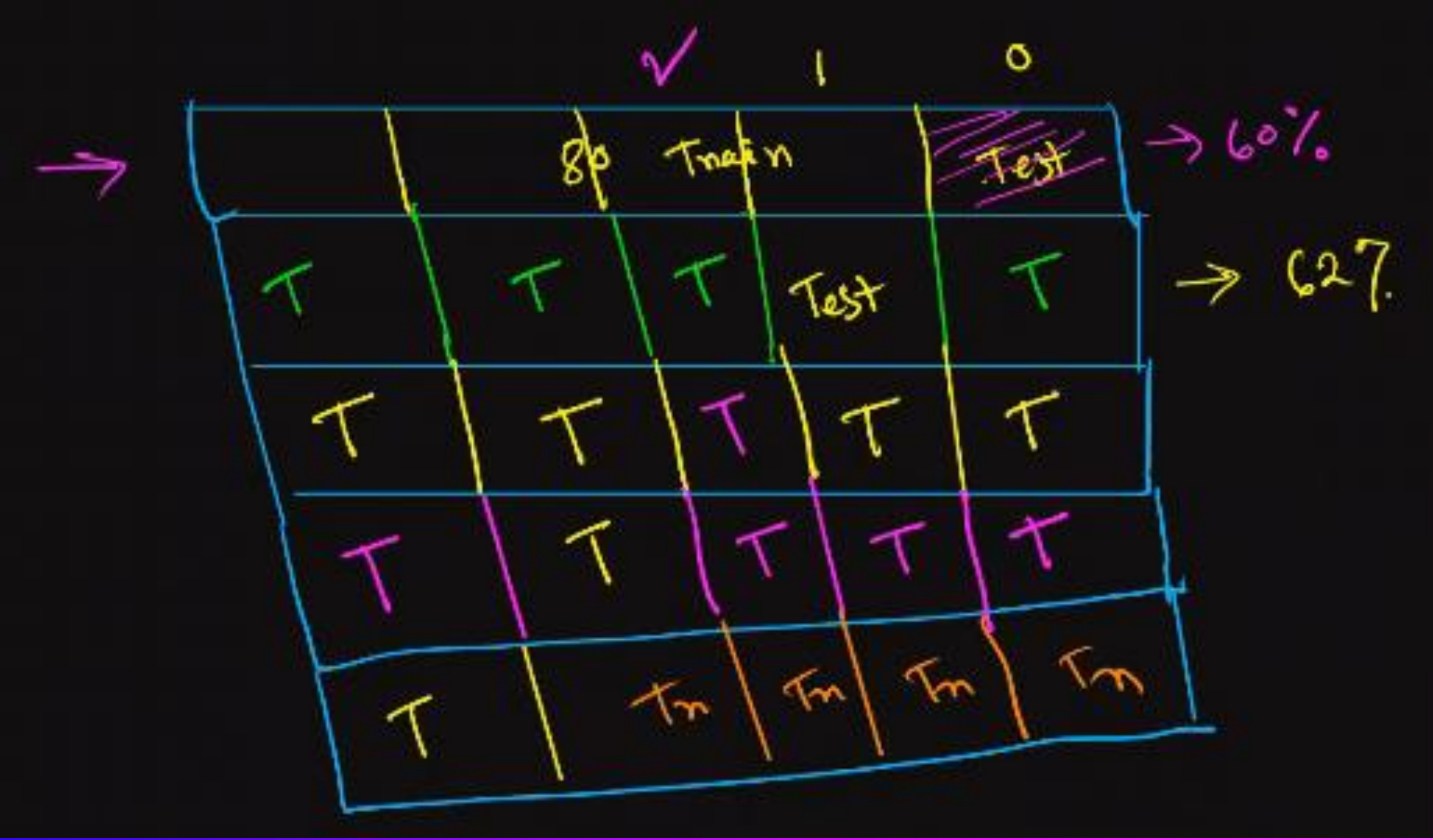
```
Trainning Accuracy: 0.62875
Testing Accuracy: 0.615
```

[ ]
```
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidian', 'manhattan'],
}
grid_search = GridSearchCV(knn, param_grid)
```


80% Train
20% Test
Test
Test
Biased


Test
Test
80
Train
Test
60%


✓
1
0
8/p
Train
Test
→ 60%
T
T
T
Test
T
→ 627.
T
T
T
T
T


✓
1
0
Train
Test
→ 60%
Test
→ 627.

[44...
```
knn = KNeighborsClassifier(n_neighbors = 25)
knn.fit(X_train, y_train)
```

[44...
KNeighborsClassifier

KNeighborsClassifier(n_neighbors=25)

[45...
```
print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
```

```
Trainning Accuracy: 0.62875
Testing Accuracy: 0.615
```

[ ]:
```
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidian', 'manhattan']
}
grid_search = GridSearchCV(knn, param_grid, cv = 5, scoring = 'accuracy')

grid_search.fit()
```

```
[44...] knn = KNeighborsClassifier(n_neighbors = 25)
        knn.fit(X_train, y_train)
```

```
[44...] KNeighborsClassifier
        KNeighborsClassifier(n_neighbors=25)
```

```
[45...] print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
        print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
```

```
Trainning Accuracy: 0.62875
Testing Accuracy: 0.615
```

```
[ ]: param_grid = {
        'n_neighbors': [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27],
        'weights': ['uniform', 'distance'],
        'metric': ['euclidian', 'manhattan']
     }
     grid_search = GridSearchCV(knn, param_grid, cv = 5, scoring = 'accuracy')

     grid_search.fit(X_train, y_train)

     best_params = grid_search.best_params_
     print(f'Best Hyperparameters: {best_params}')
```

File Edit View Run Kernel Settings Help

Code

JupyterLab Python 3 (ipykernel)

[44]
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)
```

[45]
```
print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
```

```
Trainning Accuracy: 0.62875
Testing Accuracy: 0.615
```

[46]
```
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidian', 'manhattan']
}
grid_search = GridSearchCV(knn, param_grid, cv = 5, scoring = 'accuracy')

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print(f'Best Hyperparameters: {best_params}')
```

```
Best Hyperparameters: {'metric': 'manhattan', 'n_neighbors': 27, 'weights': 'uniform'}
```

[ ]:

File Edit View Run Kernel Settings Help

Code

JupyterLab

Python 3 (ipykernel)

[44]:
```
KNeighborsClassifier(n_neighbors=25)
```

[45]:
```
print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
```

```
Trainning Accuracy: 0.62875
Testing Accuracy: 0.615
```

[46]:
```
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidian', 'manhattan']
}
grid_search = GridSearchCV(knn, param_grid, cv = 5, scoring = 'accuracy')

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print(f'Best Hyperparameters: {best_params}')
```

```
Best Hyperparameters: {'metric': 'manhattan', 'n_neighbors': 27, 'weights': 'uniform'}
```

[ ]:
```
knn = KNeighborsClassifier(n_neighbors = 11)
knn.fit(X_train, y_train)
```

```
    'metric': ['euclidian', 'manhattan']
}
grid_search = GridSearchCV(knn, param_grid, cv = 5, scoring = 'accuracy')

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print(f'Best Hyperparameters: {best_params}')
```

```
Best Hyperparameters: {'metric': 'manhattan', 'n_neighbors': 27, 'weights': 'uniform'}
```

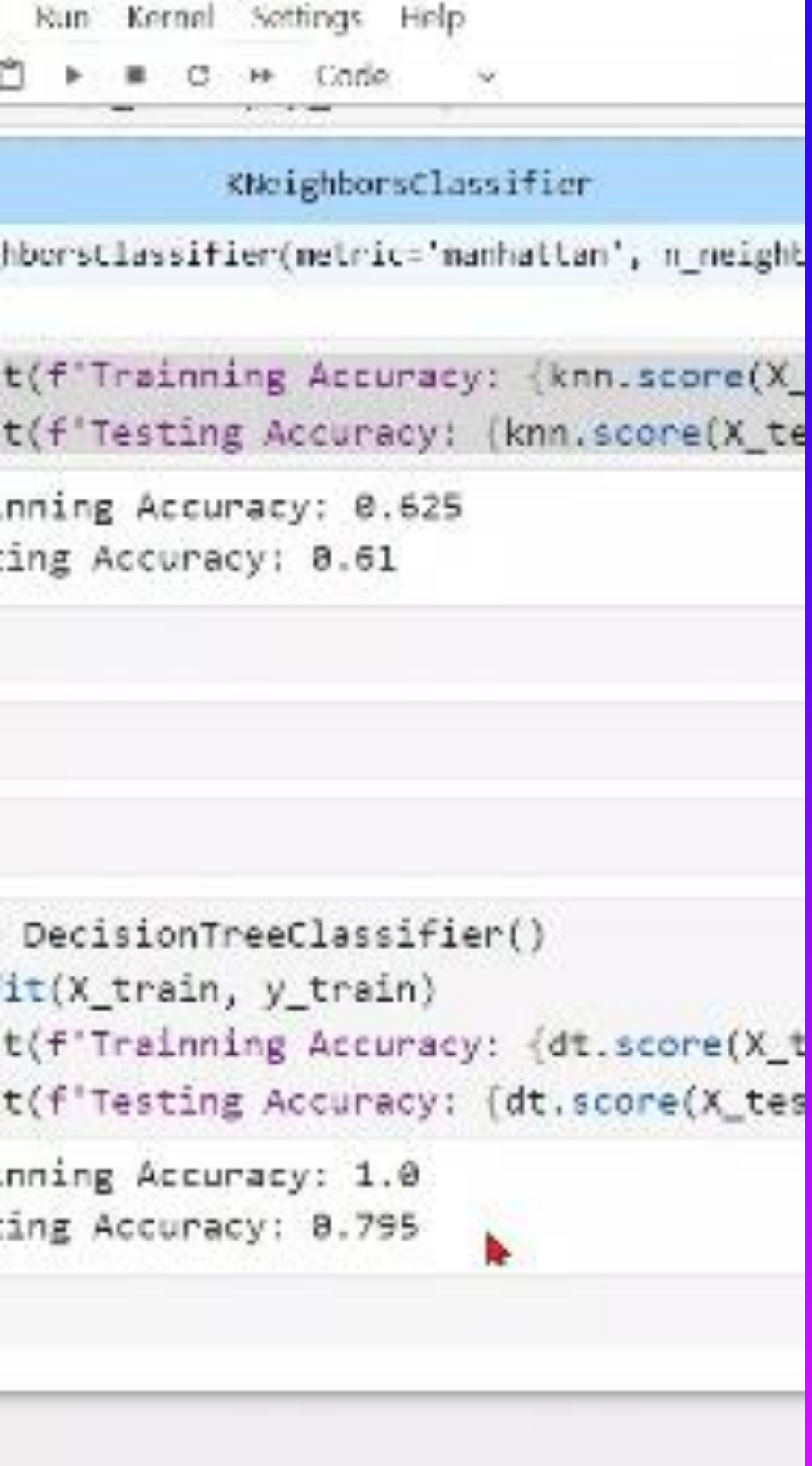
[48...]
```
knn = KNeighborsClassifier(metric = 'manhattan', n_neighbors = 27, weights='uniform')
knn.fit(X_train, y_train)
```

[48...]
```
KNeighborsClassifier(metric='manhattan', n_neighbors=27)
```

[49...]
```
print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
```

```
Trainning Accuracy: 0.625
Testing Accuracy: 0.61
```

[ ]:


Run Kernel Settings Help
Code
KNeighborsClassifier
hborsClassifier(metric='manhattan', n_neighb
t(f'Trainning Accuracy: {knn.score(X_
t(f'Testing Accuracy: {knn.score(X_te
nning Accuracy: 0.625
ting Accuracy: 0.61
DecisionTreeClassifier()
it(X_train, y_train)
t(f'Trainning Accuracy: {dt.score(X_t
t(f'Testing Accuracy: {dt.score(X_tes
nning Accuracy: 1.0
ting Accuracy: 0.795

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab

Python 3 (ipykernel)

[48...

KNeighborsClassifier

KNeighborsClassifier(metric='manhattan', n_neighbors=27)

[49...

```
print(f'Trainning Accuracy: {knn.score(X_train, y_train)}')
print(f'Testing Accuracy: {knn.score(X_test, y_test)}')
```

```
Trainning Accuracy: 0.625
Testing Accuracy: 0.61
```

[ ]:

[ ]:

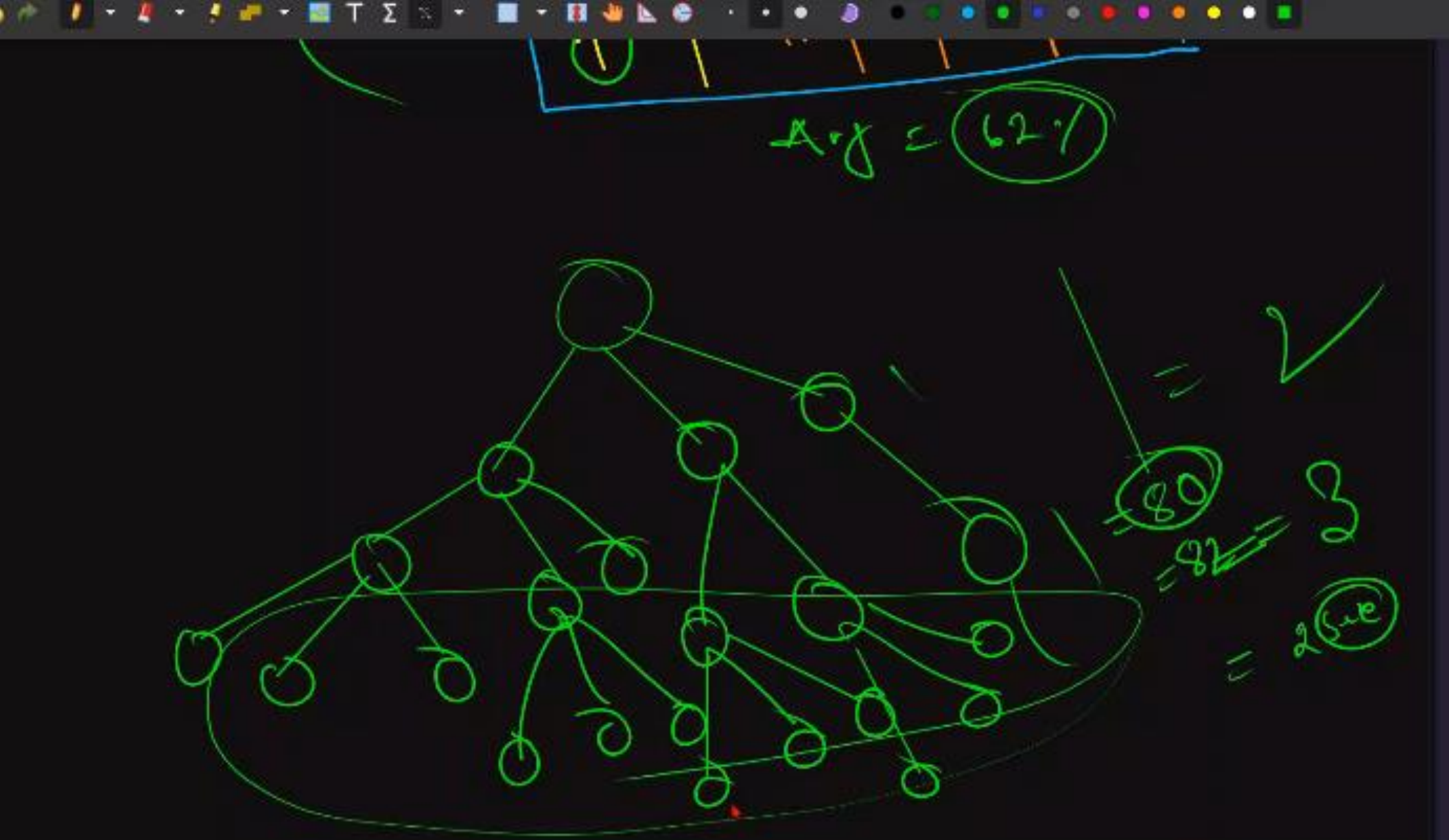
[ ]:

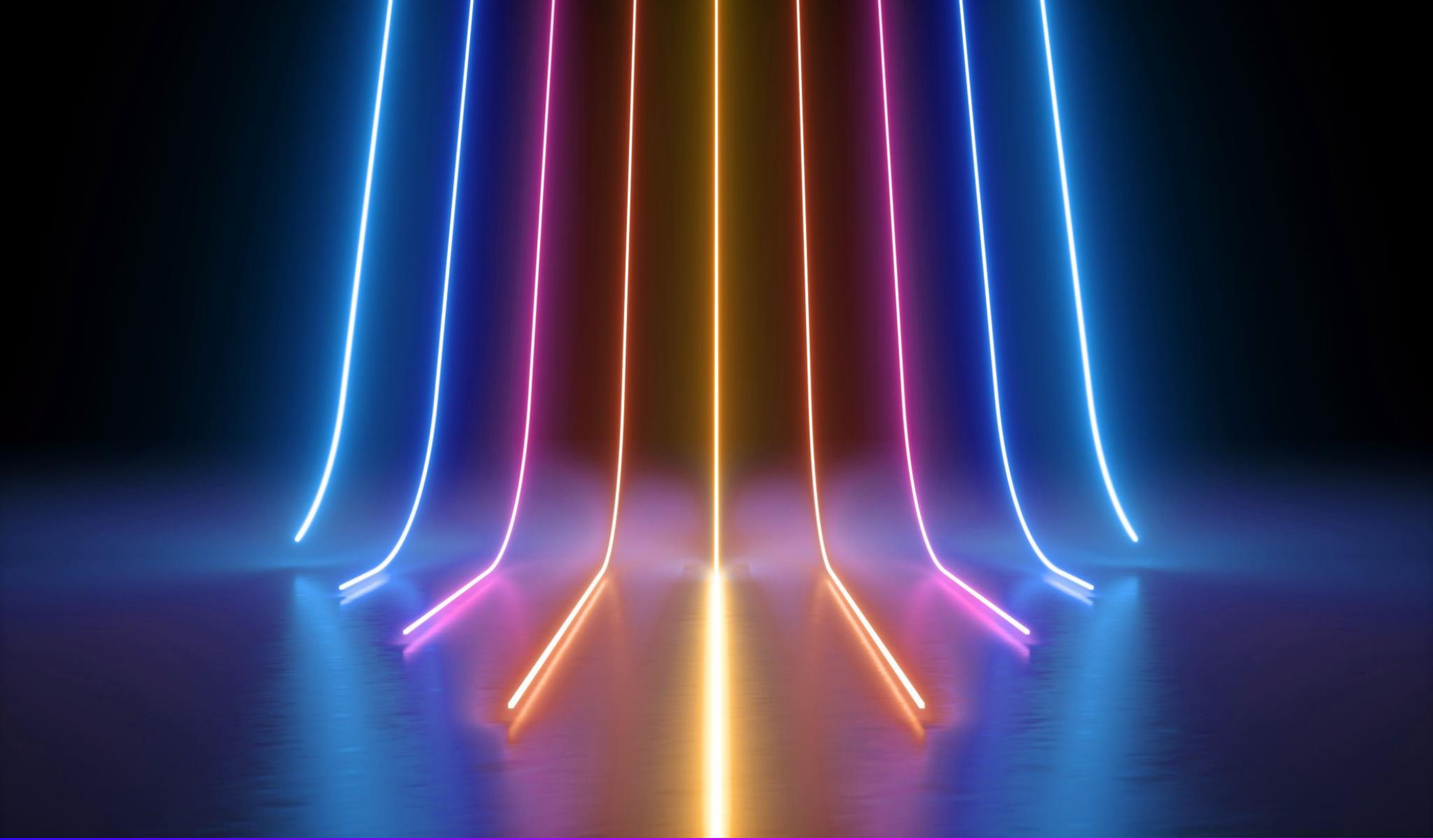
[50...

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
print(f'Trainning Accuracy: {dt.score(X_train, y_train)}')
print(f'Testing Accuracy: {dt.score(X_test, y_test)}')
```

```
Trainning Accuracy: 1.0
Testing Accuracy: 0.795
```

[ ]:


Avg = 62



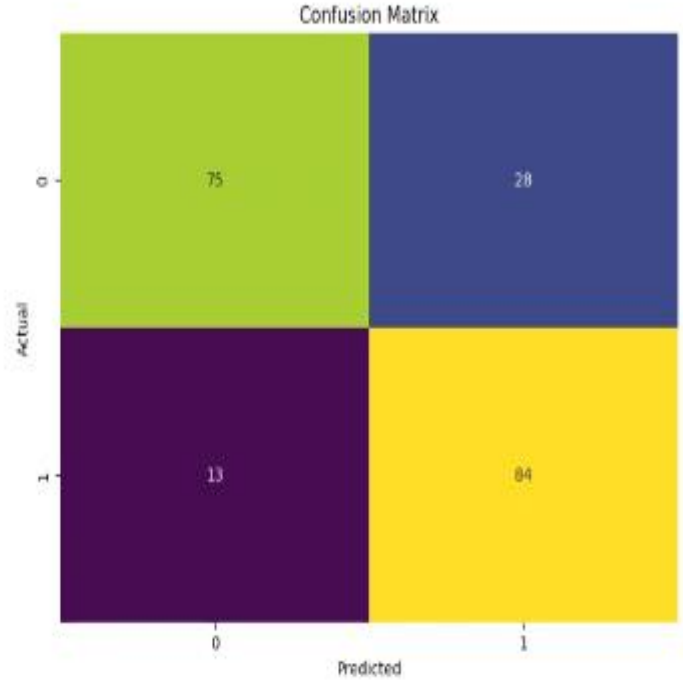
jupyter Class_01_Classsification Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

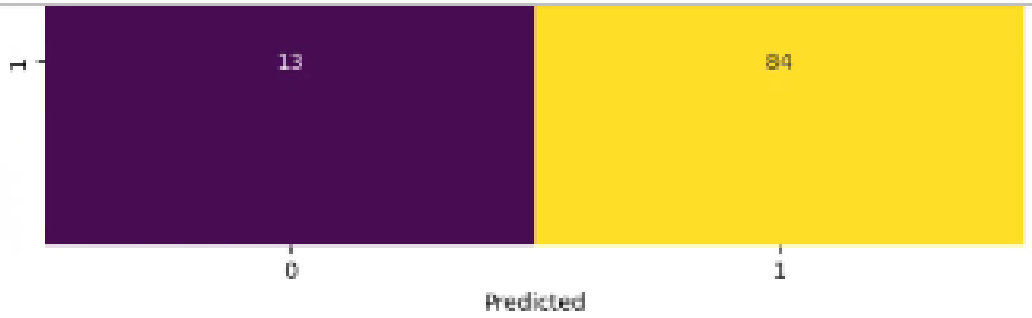
Trusted

Code

JupyterLab Python 3 (ipykernel)



[ ]:
```
param_grid = {
    'max_depth': [2, 5, 7, None],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 3, 4]
}
```



[53...
```
param_grid = {
    'max_depth': [2, 5, 7, None],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 3, 4]
}
grid_search = GridSearchCV(dt, param_grid, cv = 5, scoring = 'accuracy')

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print(f'Best Hyperparameters: {best_params}')
```

```
Best Hyperparameters: {'max_depth': 2, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

[ ]:
```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
print(f'Trainning Accuracy: {dt.score(X_train, y_train)}')
```