Pelatihan ABCD
Modul 4-3: K Nearest Neighbors

Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung

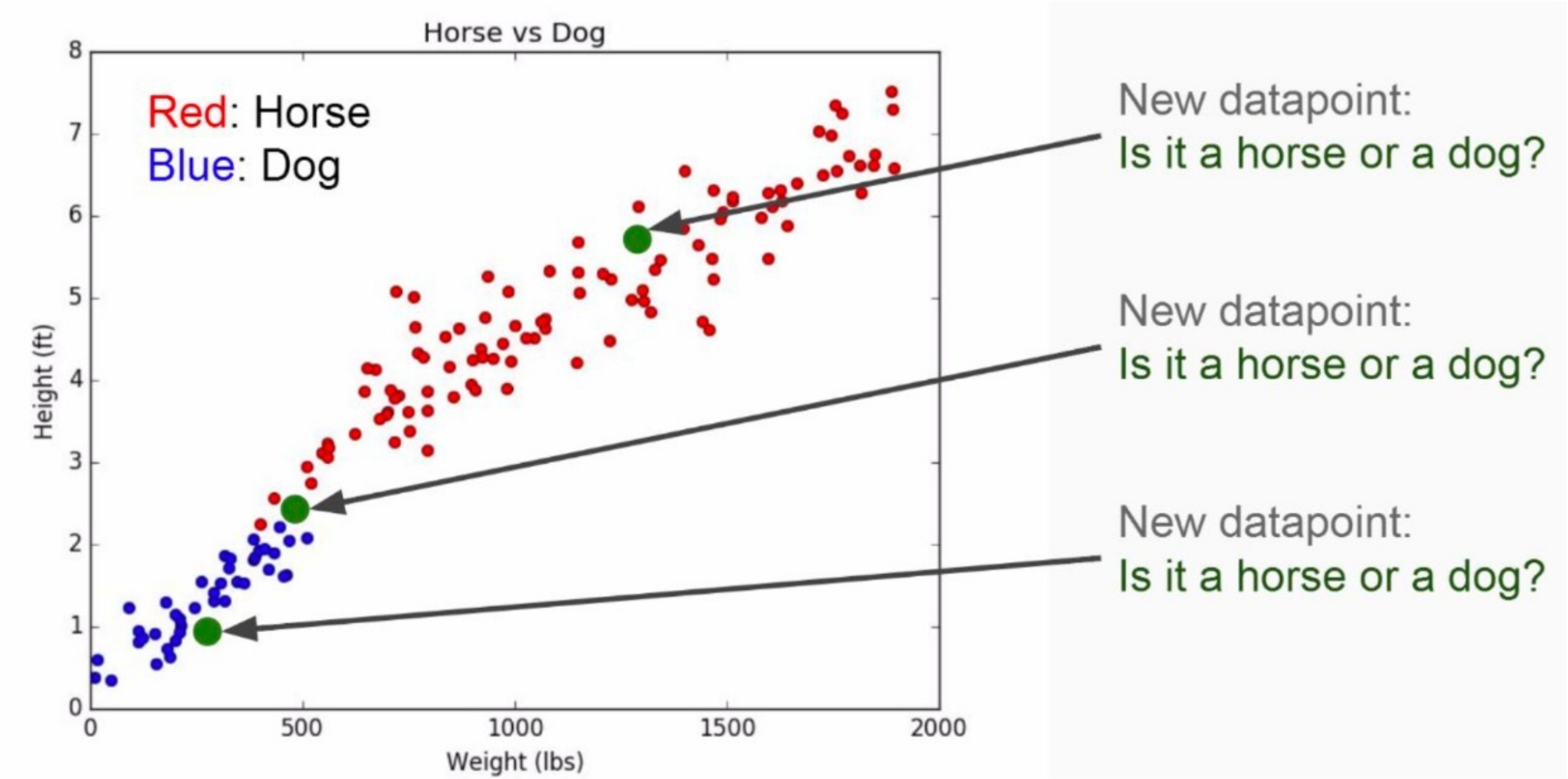
Unviersitas Singaperbangsa Karawang

Contents

- K Nearest Neighbors (KNN) Concepts
- KNN with Python

K Nearest Neighbors (KNN)

- K Nearest Neighbors (KNN) is a **classification algorithm** that operates on a very simple principle
- KNN illustration: data on Dogs and Horses, with heights and weights



New datapoint:
Is it a horse or a dog?

New datapoint:
Is it a horse or a dog?

New datapoint:
Is it a horse or a dog?

KNN

Training algorithm:

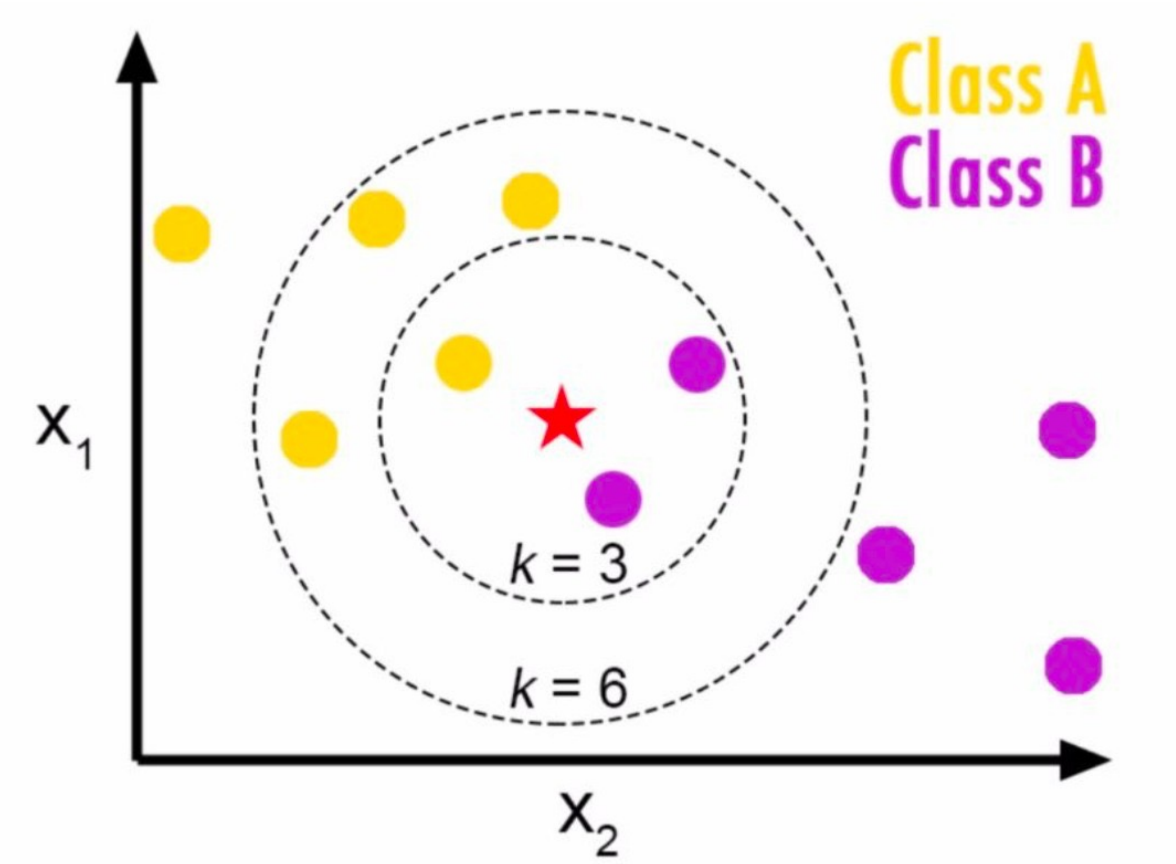
1. Store all the data

Prediction algorithm:

1. Calculate the distance from x to all points in your data
2. Sort the points in your data by increasing distance from x
3. Predict the majority label of the “k” closest points

KNN

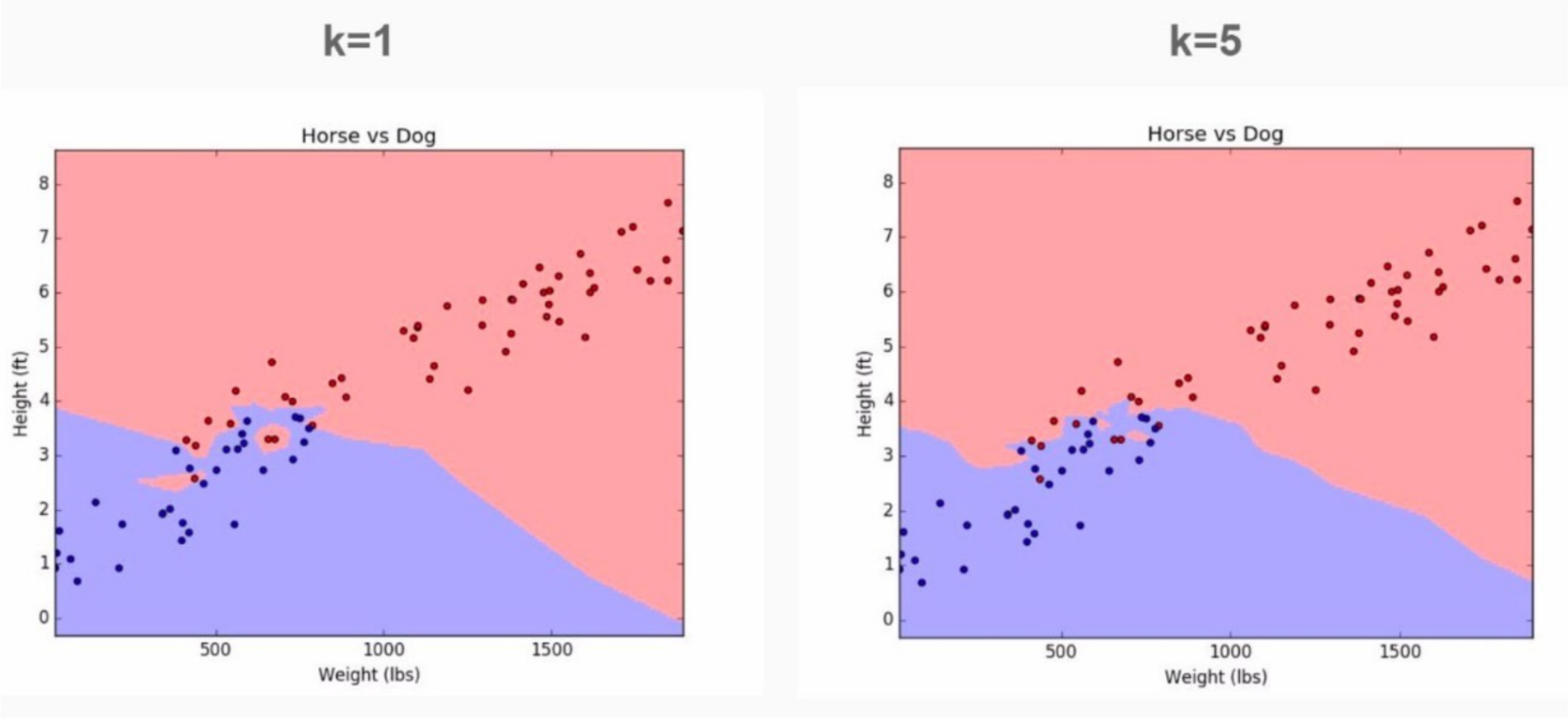
Choosing a K will affect what class a new point is assigned to



For k = 3, the new point belong to Class B
For k = 6, the new point belong to Class A

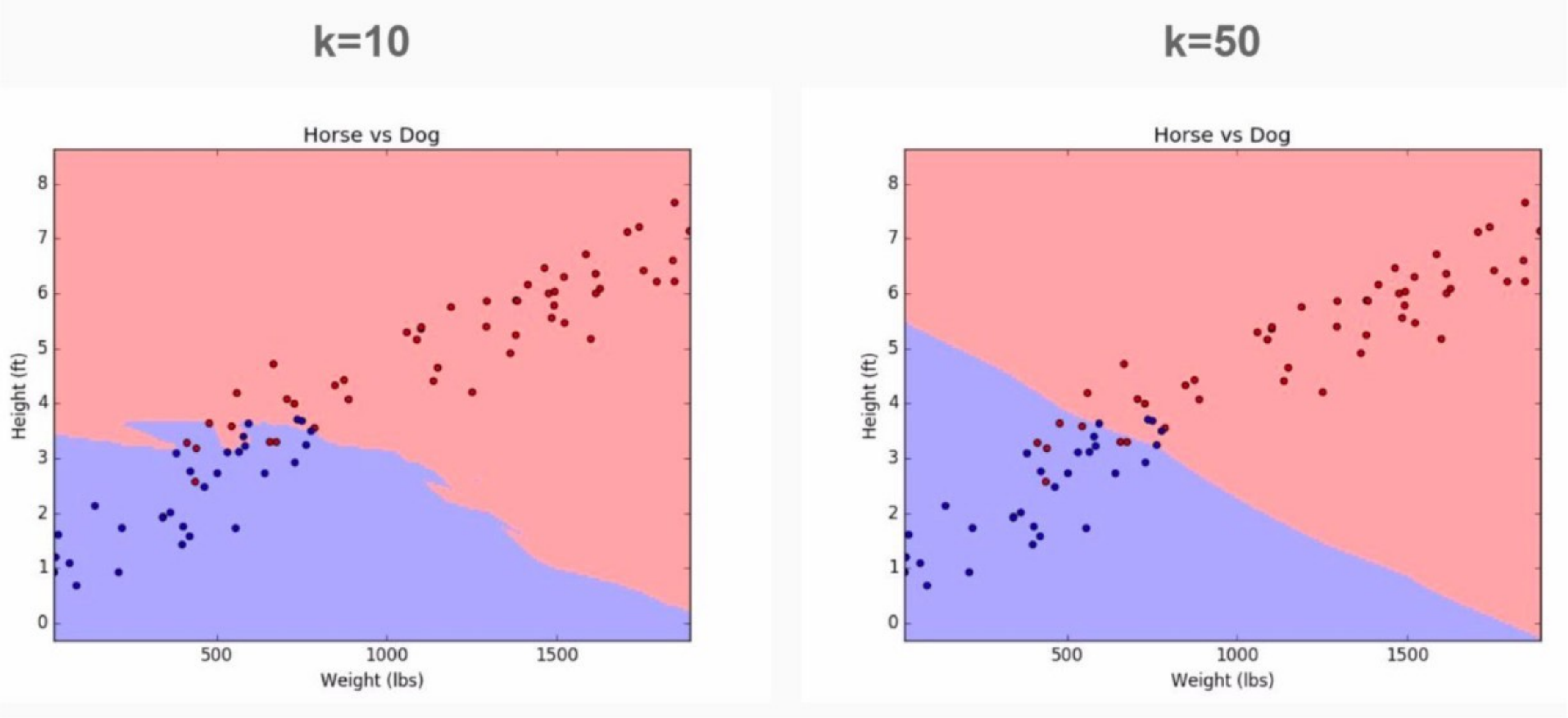
KNN

Choosing a K will affect what class a new point is assigned to



KNN

Choosing a K will affect what class a new point is assigned to



KNN

Pros

- Very simple
- Training is trivial
- Works with any number of classes
- Easy to add more data
- Few parameters
  - K
  - Distance Metric

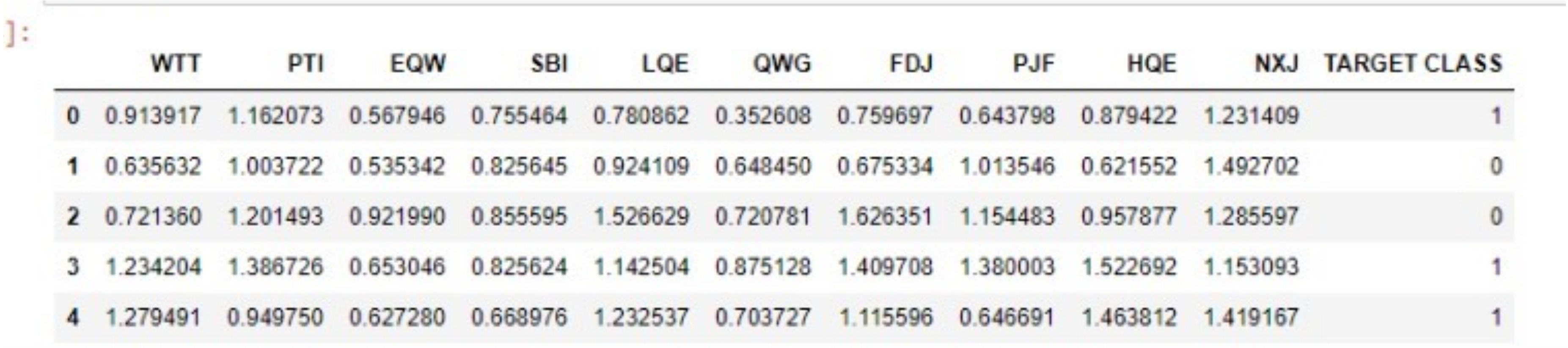
Cons

- High prediction Cost (worse for large data sets)
- Not good with high dimensional data
- Categorical features don’t work well

K Nearest Neighbors (KNN) with Python

Classification Problem Example using KNN

- We are given a classified data set from a company. They’ve hidden the feature column names but have given us the data and the target classes.
- We’ll try to use KNN to create a model that directly predicts a class for a new data point based of the features.
- See file: KNN.ipynb for the code. And CSV data file: “Classified Data”



	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ	TARGET CLASS
0	0.913917	1.162073	0.567946	0.755464	0.780862	0.352608	0.759697	0.643798	0.879422	1.231409	1
1	0.635632	1.003722	0.535342	0.825645	0.924109	0.648450	0.675334	1.013546	0.621552	1.492702	0
2	0.721360	1.201493	0.921990	0.855595	1.526629	0.720781	1.626351	1.154483	0.957877	1.285597	0
3	1.234204	1.386726	0.653046	0.825624	1.142504	0.875128	1.409708	1.380003	1.522692	1.153093	1
4	1.279491	0.949750	0.627280	0.668976	1.232537	0.703727	1.115596	0.646691	1.463812	1.419167	1

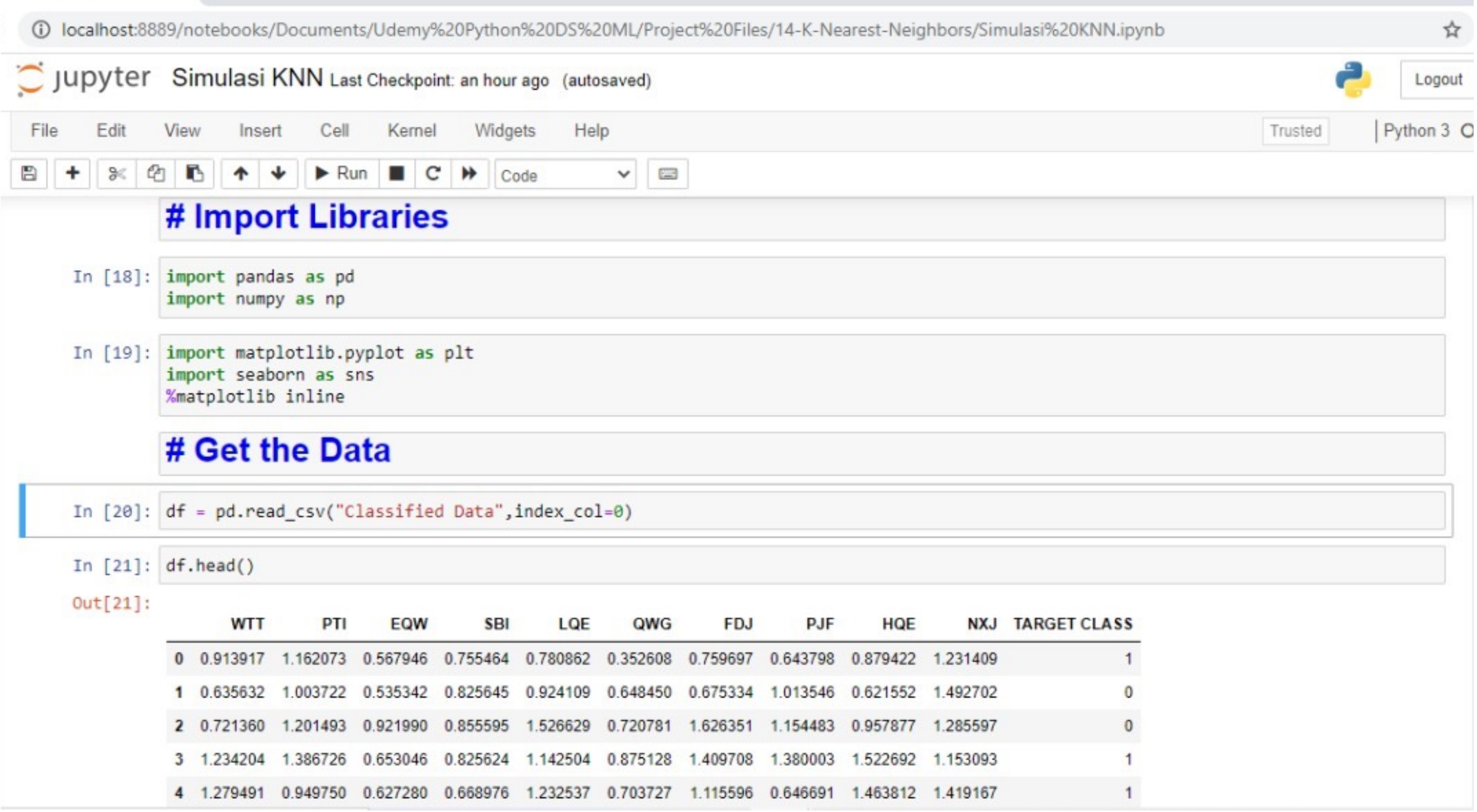
The Python KNN Process

1. Import Libraries
2. Get the Data
   - Set index_col = 0 to use the first column as the index
3. Standardize the variables
   - Because the KNN classifier predicts the class of a given test observation by identifying the observations that are nearest to it, the scale of the variables matters. Any variables that are on a large scale will have a much larger effect on the distance between the observations, and hence on the KNN classifier, than variables that are on a small scale.
4. Train Test Data Split

The Python KNN Process (cont.)

5. Build model Using KNN
   - Build a models to predict whether someone will TARGET CLASS or not. We'll start with k=1.
6. Prediction and Evaluations
   - Evaluation the KNN model
7. Choosing a K Value
   - Use the elbow method to pick a good K value

1. Import Libraries and 2. Get the Data


localhost:8889/notebooks/Documents/Udemy%20Python%20DS%20ML/Project%20Files/14-K-Nearest-Neighbors/Simulasi%20KNN.ipynb
jupyter
Simulasi KNN Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

Import Libraries

```
In [18]: import pandas as pd
         import numpy as np
```

```
In [19]: import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
```

Get the Data

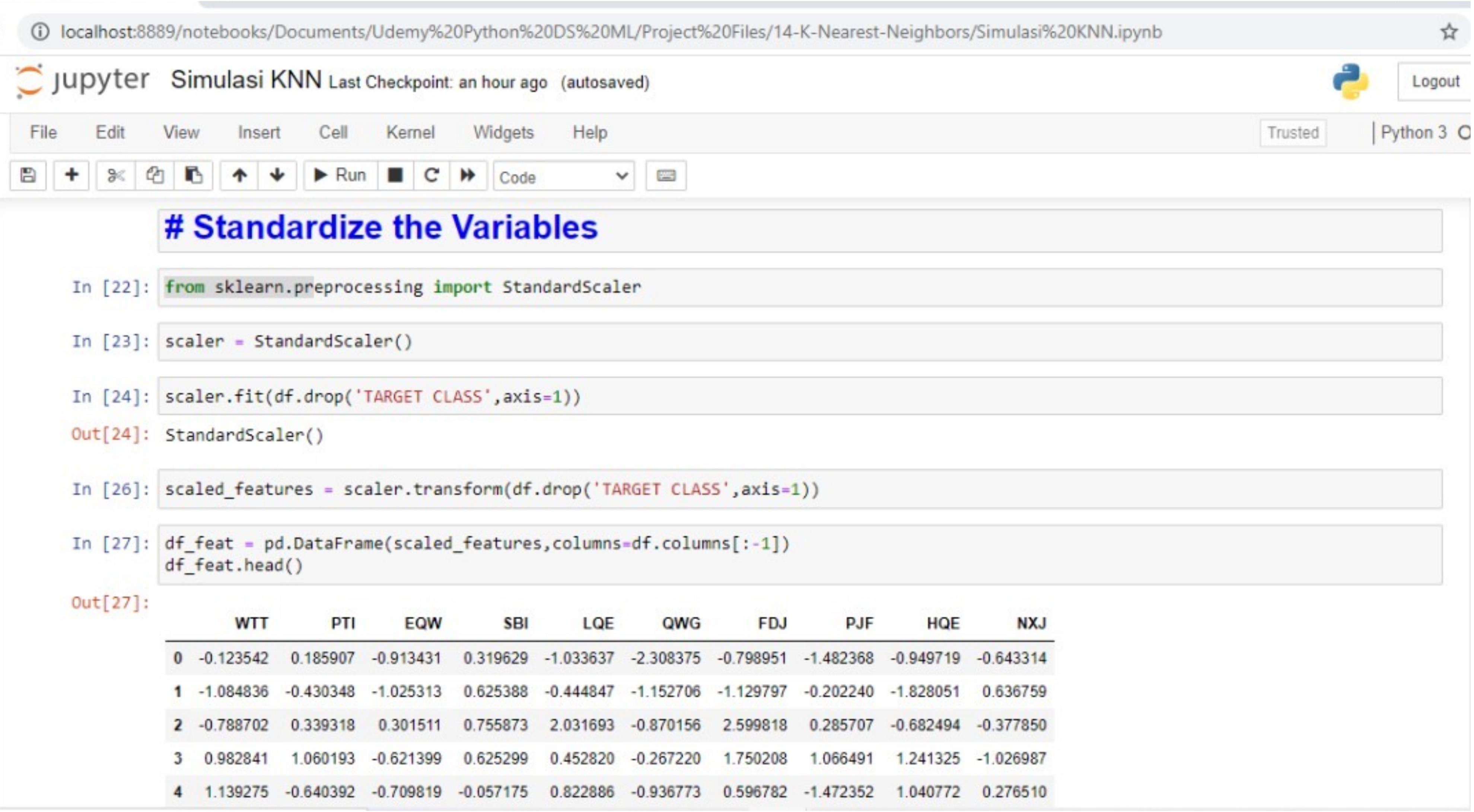
```
In [20]: df = pd.read_csv("Classified Data",index_col=0)
```

```
In [21]: df.head()
```

Out[21]:

	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ	TARGET CLASS
0	0.913917	1.162073	0.567946	0.755464	0.780862	0.352608	0.759697	0.643798	0.879422	1.231409	1
1	0.635632	1.003722	0.535342	0.825645	0.924109	0.648450	0.675334	1.013546	0.621552	1.492702	0
2	0.721360	1.201493	0.921990	0.855595	1.526629	0.720781	1.626351	1.154483	0.957877	1.285597	0
3	1.234204	1.386726	0.653046	0.825624	1.142504	0.875128	1.409708	1.380003	1.522692	1.153093	1
4	1.279491	0.949750	0.627280	0.668976	1.232537	0.703727	1.115596	0.646691	1.463812	1.419167	1

3. Standardize the Variables



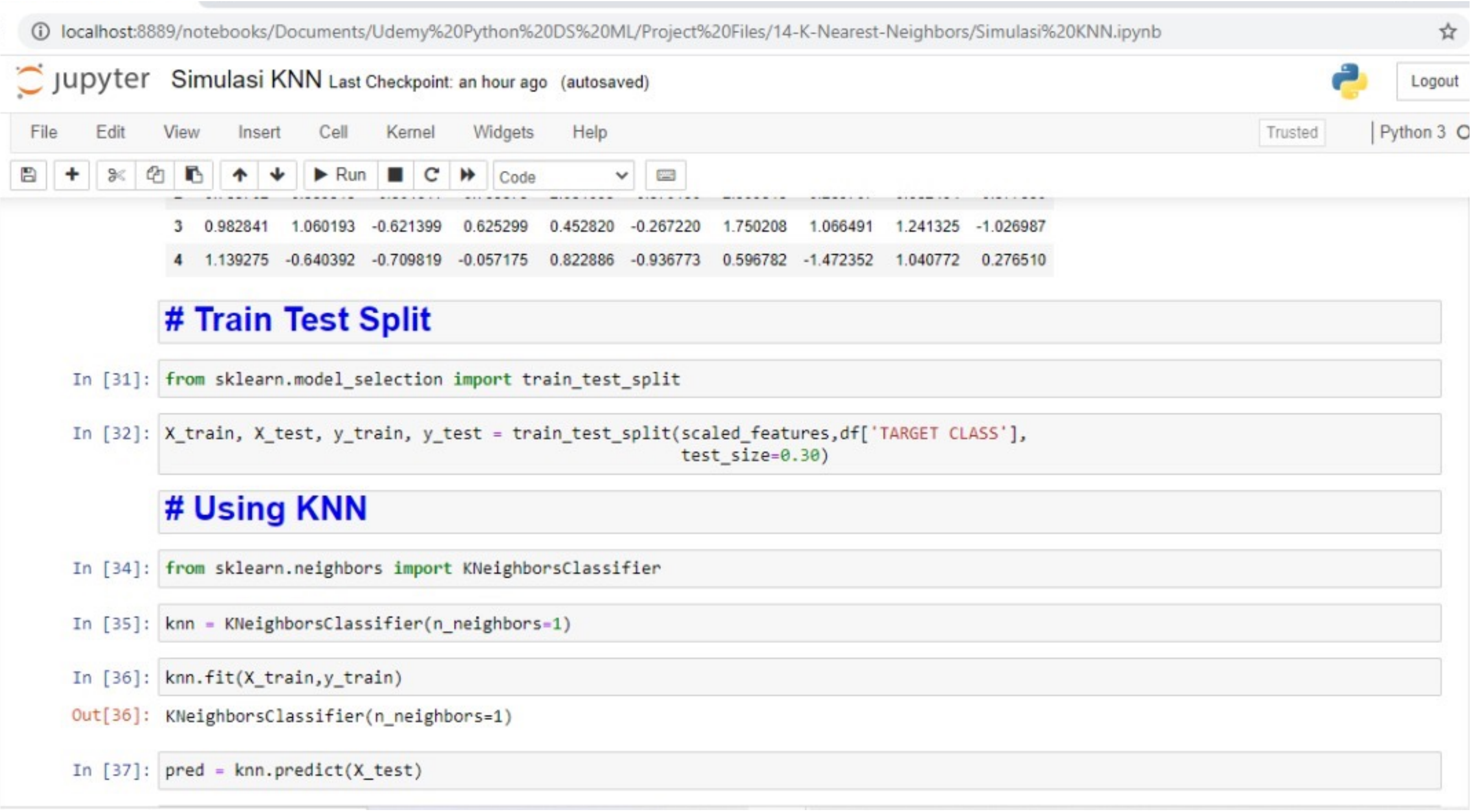
localhost:8889/notebooks/Documents/Udemy%20Python%20DS%20ML/Project%20Files/14-K-Nearest-Neighbors/Simulasi%20KNN.ipynb
jupyter
Simulasi KNN Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

Standardize the Variables

```
In [22]: from sklearn.preprocessing import StandardScaler

In [23]: scaler = StandardScaler()

In [24]: scaler.fit(df.drop('TARGET CLASS',axis=1))

Out[24]: StandardScaler()

In [26]: scaled_features = scaler.transform(df.drop('TARGET CLASS',axis=1))

In [27]: df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
         df_feat.head()

Out[27]:
```

	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ
0	-0.123542	0.185907	-0.913431	0.319629	-1.033637	-2.308375	-0.798951	-1.482368	-0.949719	-0.643314
1	-1.084836	-0.430348	-1.025313	0.625388	-0.444847	-1.152706	-1.129797	-0.202240	-1.828051	0.636759
2	-0.788702	0.339318	0.301511	0.755873	2.031693	-0.870156	2.599818	0.285707	-0.682494	-0.377850
3	0.982841	1.060193	-0.621399	0.625299	0.452820	-0.267220	1.750208	1.066491	1.241325	-1.026987
4	1.139275	-0.640392	-0.709819	-0.057175	0.822886	-0.936773	0.596782	-1.472352	1.040772	0.276510

4. Splitting and 5. Build model


localhost:8889/notebooks/Documents/Udemy%20Python%20DS%20ML/Project%20Files/14-K-Nearest-Neighbors/Simulasi%20KNN.ipynb
jupyter
Simulasi KNN Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

3	0.982841	1.060193	-0.621399	0.625299	0.452820	-0.267220	1.750208	1.066491	1.241325	-1.026987
4	1.139275	-0.640392	-0.709819	-0.057175	0.822886	-0.936773	0.596782	-1.472352	1.040772	0.276510

Train Test Split

```
In [31]: from sklearn.model_selection import train_test_split
```

```
In [32]: X_train, X_test, y_train, y_test = train_test_split(scaled_features,df['TARGET CLASS'],
                                                             test_size=0.30)
```

Using KNN

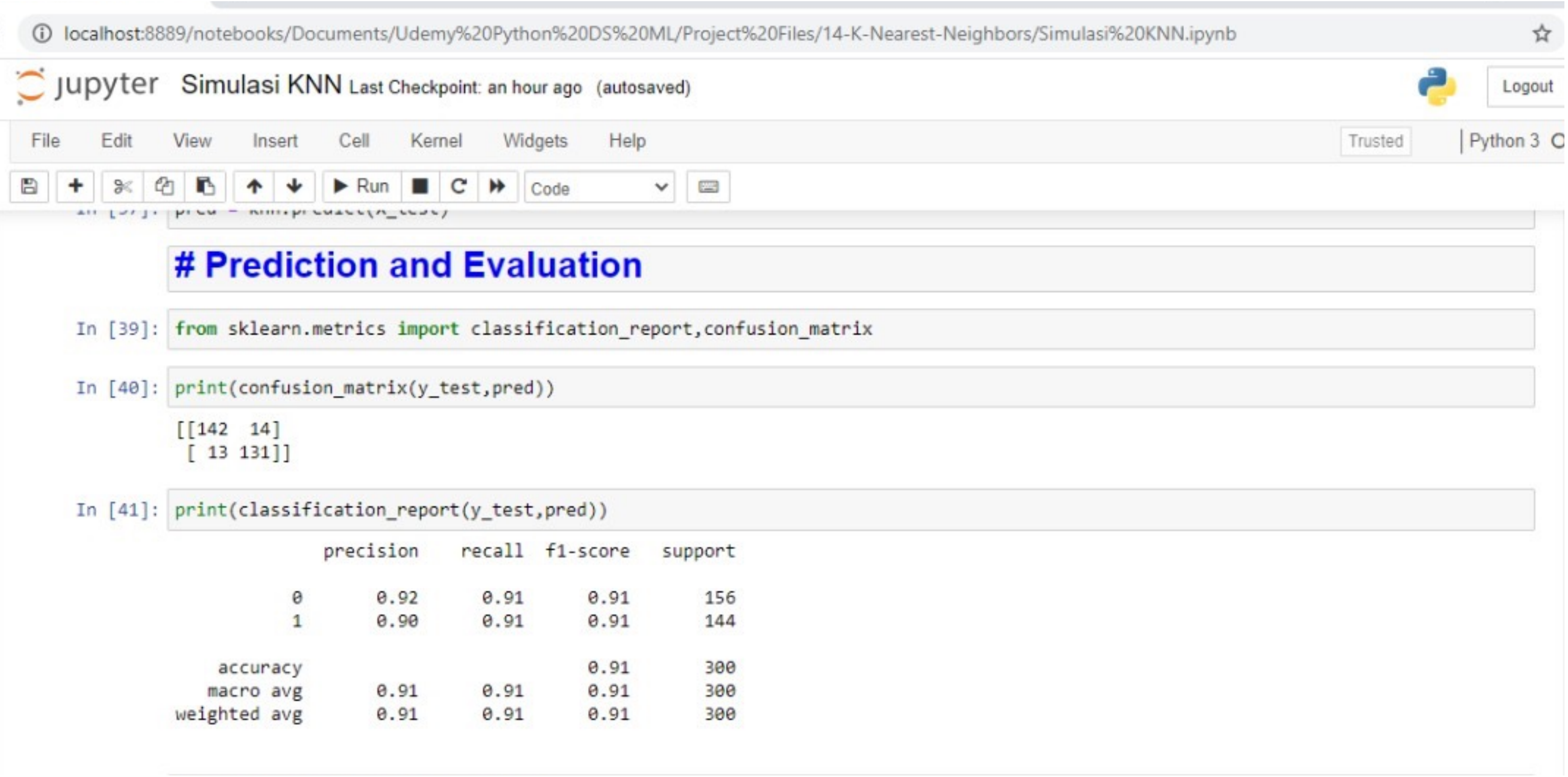
```
In [34]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [35]: knn = KNeighborsClassifier(n_neighbors=1)
```

```
In [36]: knn.fit(X_train,y_train)

Out[36]: KNeighborsClassifier(n_neighbors=1)
```

```
In [37]: pred = knn.predict(X_test)
```

6. Prediction and Evaluation


localhost:8889/notebooks/Documents/Udemy%20Python%20DS%20ML/Project%20Files/14-K-Nearest-Neighbors/Simulasi%20KNN.ipynb
jupyter Simulasi KNN Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

Prediction and Evaluation

```
In [39]: from sklearn.metrics import classification_report,confusion_matrix
```

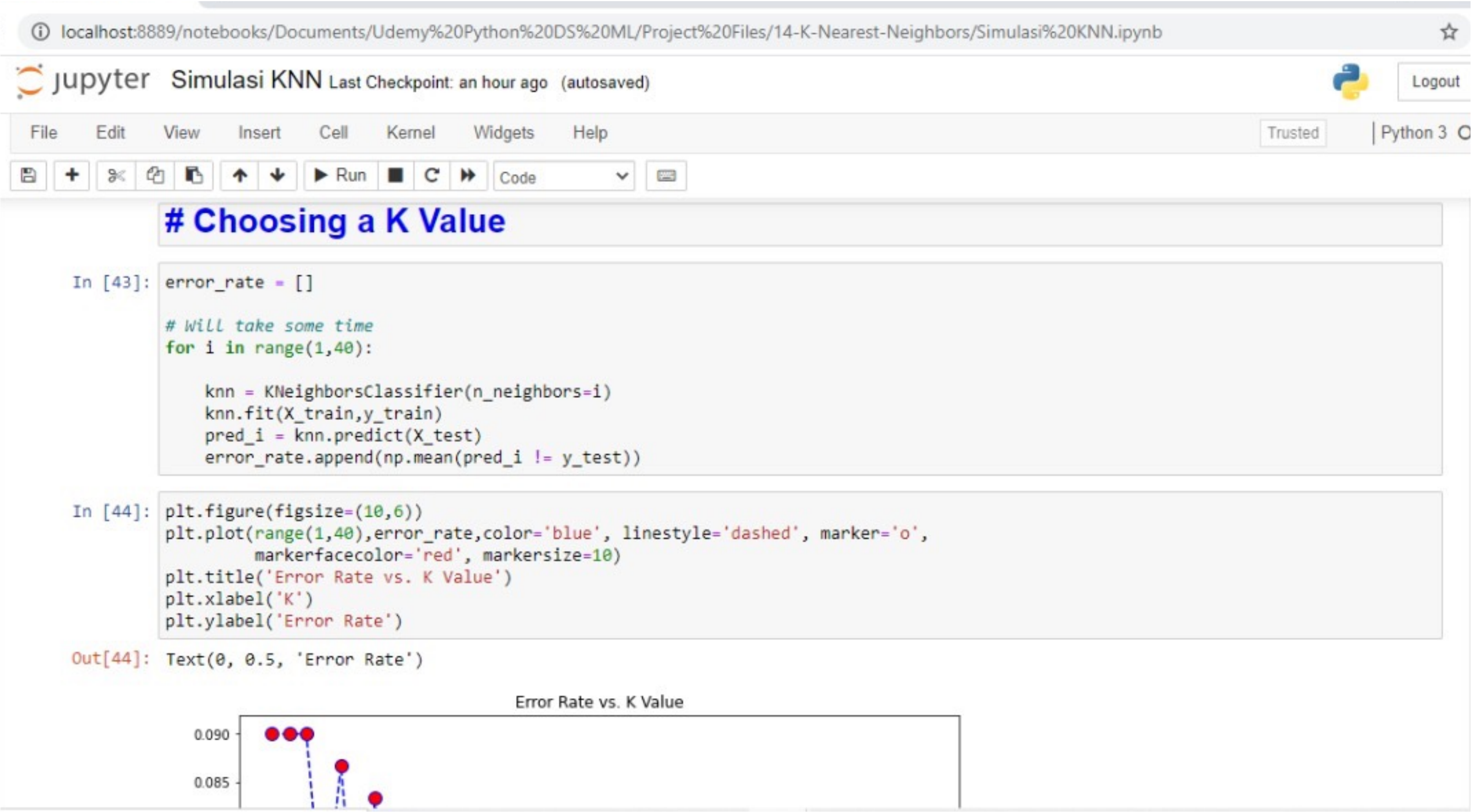
```
In [40]: print(confusion_matrix(y_test,pred))
```

```
[[142  14]
 [ 13 131]]
```

```
In [41]: print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.91      0.91       156
           1       0.90      0.91      0.91       144

    accuracy                           0.91       300
   macro avg       0.91      0.91      0.91       300
weighted avg       0.91      0.91      0.91       300
```

7. Choosing a K Value



localhost:8889/notebooks/Documents/Udemy%20Python%20DS%20ML/Project%20Files/14-K-Nearest-Neighbors/Simulasi%20KNN.ipynb
jupyter Simulasi KNN Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

Choosing a K Value

In [43]:

```
error_rate = []

# Will take some time
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

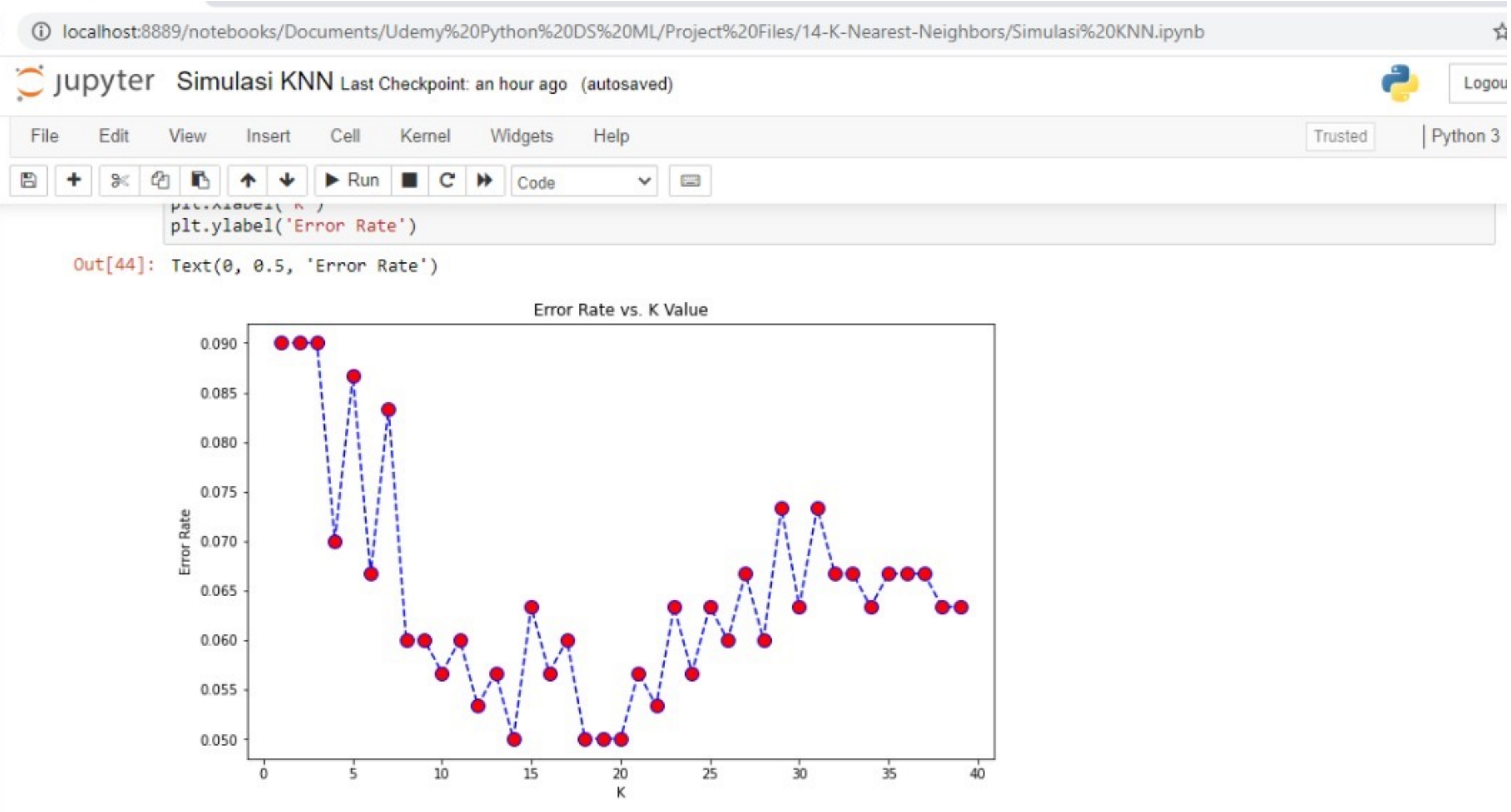
In [44]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Out[44]:

```
Text(0, 0.5, 'Error Rate')
```

7. Choosing a K Value



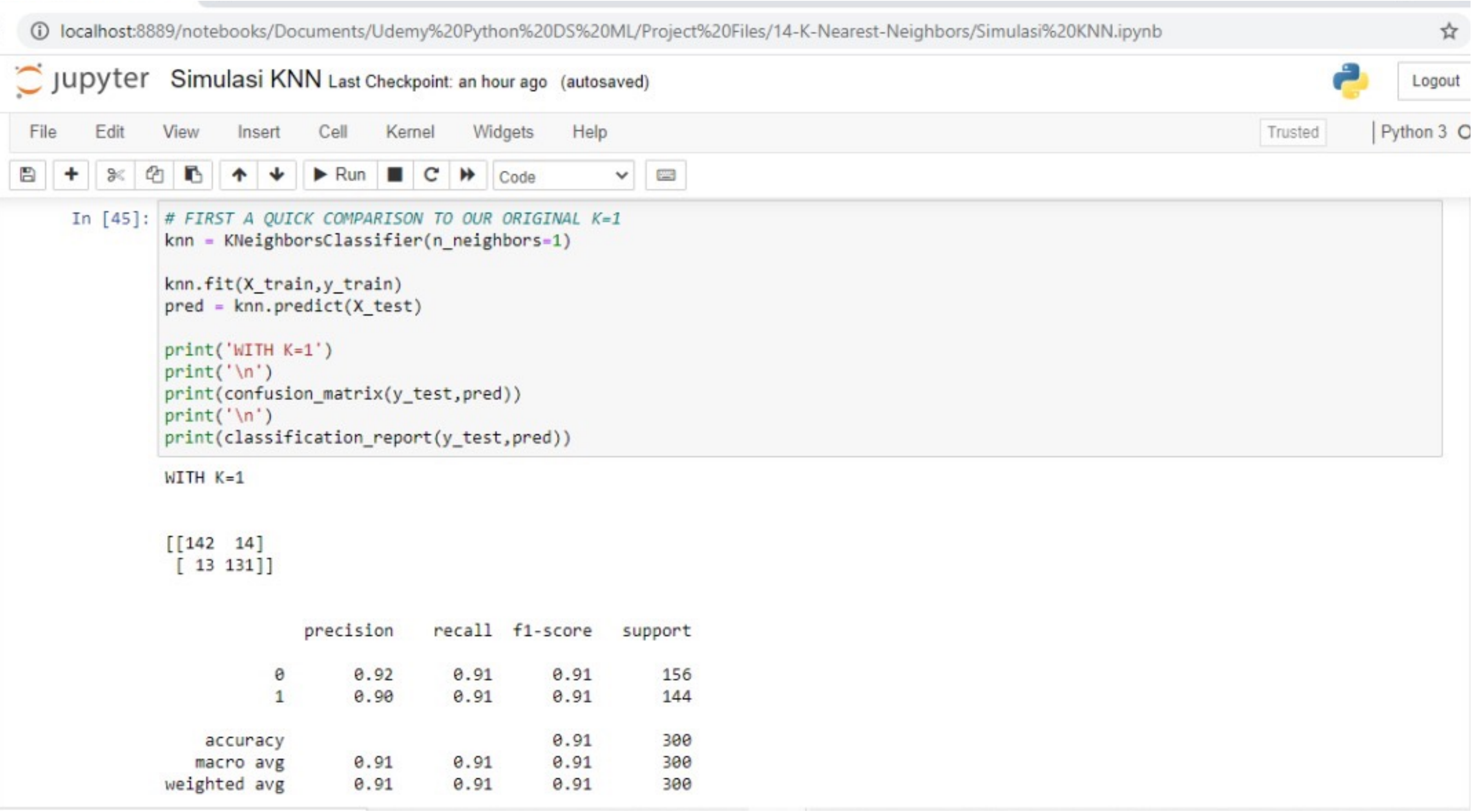
localhost:8889/notebooks/Documents/Udemy%20Python%20DS%20ML/Project%20Files/14-K-Nearest-Neighbors/Simulasi%20KNN.ipynb
jupyter
Simulasi KNN Last Checkpoint: an hour ago (autosaved)
Logou
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

```
plt.xlabel('K')
plt.ylabel('Error Rate')
```

```
Out[44]: Text(0, 0.5, 'Error Rate')
```

Error Rate vs. K Value
Error Rate
0.090
0.085
0.080
0.075
0.070
0.065
0.060
0.055
0.050
0
5
10
15
20
25
30
35
40
K

K = 1



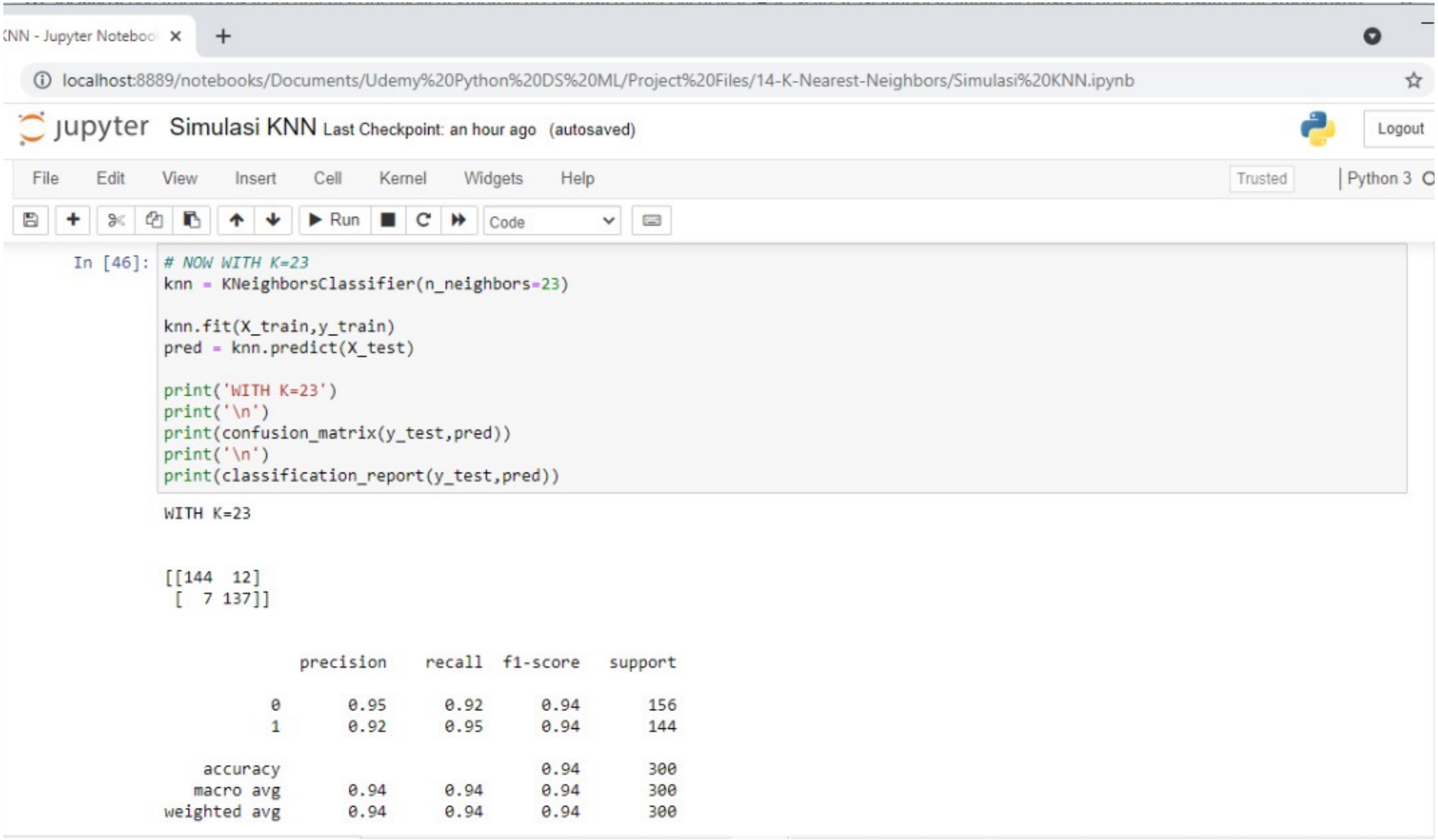
localhost:8889/notebooks/Documents/Udemy%20Python%20DS%20ML/Project%20Files/14-K-Nearest-Neighbors/Simulasi%20KNN.ipynb
jupyter
Simulasi KNN Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

In [45]:

```
# FIRST A QUICK COMPARISON TO OUR ORIGINAL K=1
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH K=1')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH K=1


[[142  14]
 [ 13 131]]


              precision    recall  f1-score   support

           0       0.92      0.91      0.91       156
           1       0.90      0.91      0.91       144

    accuracy                           0.91       300
   macro avg       0.91      0.91      0.91       300
weighted avg       0.91      0.91      0.91       300
```

K = 23



KNN - Jupyter Noteboo
localhost:8889/notebooks/Documents/Udemy%20Python%20DS%20ML/Project%20Files/14-K-Nearest-Neighbors/Simulasi%20KNN.ipynb
jupyter Simulasi KNN Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [46]: # NOW WITH K=23
         knn = KNeighborsClassifier(n_neighbors=23)

         knn.fit(X_train,y_train)
         pred = knn.predict(X_test)

         print('WITH K=23')
         print('\n')
         print(confusion_matrix(y_test,pred))
         print('\n')
         print(classification_report(y_test,pred))
```

```
WITH K=23


[[144  12]
 [  7 137]]


              precision    recall  f1-score   support

           0       0.95      0.92      0.94       156
           1       0.92      0.95      0.94       144

    accuracy                           0.94       300
   macro avg       0.94      0.94      0.94       300
weighted avg       0.94      0.94      0.94       300
```