K Nearest Neighbors With Python

We've been given a classified data set from a company! They've hidden the feature columns names but have given you the data and target classes. We'll try to use KNN to create a model that directly predicts a class for a new data point based off of the features.

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [33]:
```
dataset = pd.read_csv("Classified Data",index_col = 0)
dataset.head()
```

Out[33]:

	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	
0	0.913917	1.162073	0.567946	0.755464	0.780862	0.352608	0.759697	0.643798	0.879422	1.231
1	0.635632	1.003722	0.535342	0.825645	0.924109	0.648450	0.675334	1.013546	0.621552	1.492
2	0.721360	1.201493	0.921990	0.855595	1.526629	0.720781	1.626351	1.154483	0.957877	1.285
3	1.234204	1.386726	0.653046	0.825624	1.142504	0.875128	1.409708	1.380003	1.522692	1.153
4	1.279491	0.949750	0.627280	0.668976	1.232537	0.703727	1.115596	0.646691	1.463812	1.419

In [35]:
```
dataset.shape
```

Out[35]:
```
(1000, 11)
```

Standardize the Variables

In [3]:
```
from sklearn.preprocessing import StandardScaler
```

In [4]:
```
scaler = StandardScaler()
```

In [5]:
```
scaler.fit(dataset.drop("TARGET CLASS",axis =1))
```

Out[5]:
```
StandardScaler()
```

In [6]:
```
scaled_features = scaler.transform(dataset.drop("TARGET CLASS",axis = 1))
scaled_features
```

Out[6]:
```
array([[-0.12354188,  0.18590747, -0.91343069, ..., -1.48236813,
        -0.9497194 , -0.64331425],
       [-1.08483602, -0.43034845, -1.02531333, ..., -0.20224031,
        -1.82805088,  0.63675862],
       [-0.78870217,  0.33931821,  0.30151137, ...,  0.28570652,
        -0.68249379, -0.37784986],
       ...,
       [ 0.64177714, -0.51308341, -0.17920486, ..., -2.36249443,
        -0.81426092,  0.11159651],
       [ 0.46707241, -0.98278576, -1.46519359, ..., -0.03677699,
         0.40602453, -0.85567   ],
       [-0.38765353, -0.59589427, -1.4313981 , ..., -0.56778932,
         0.3369971 ,  0.01034996]])
```

In [7]:
```
dataset_scaled = pd.DataFrame(scaled_features,columns=dataset.columns[:-1])
dataset_scaled.head()
```

Out[7]:

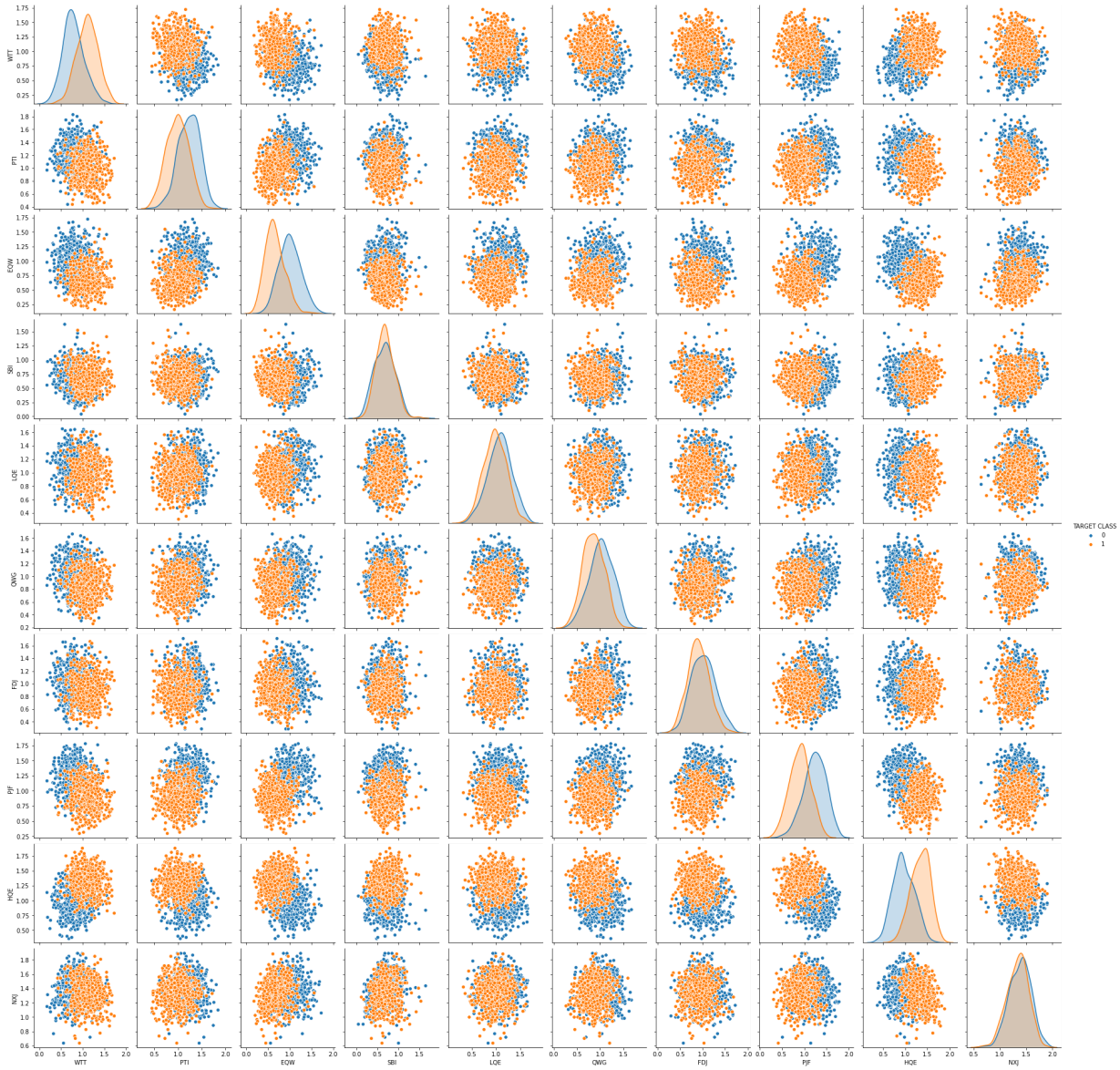
	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE
0	-0.123542	0.185907	-0.913431	0.319629	-1.033637	-2.308375	-0.798951	-1.482368	-0.949719
1	-1.084836	-0.430348	-1.025313	0.625388	-0.444847	-1.152706	-1.129797	-0.202240	-1.828051
2	-0.788702	0.339318	0.301511	0.755873	2.031693	-0.870156	2.599818	0.285707	-0.682494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

Pair Plot

In [8]:
```
import seaborn as sns
```

In [9]:
```
sns.pairplot(dataset,hue = "TARGET CLASS")
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x20e7a1d0c48>
```



In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(dataset_scaled,dataset["TARGE
```

USING KNN

Remember that we are trying to come up with a model to predict whether someone will TARGET CLASS or not. We'll start k=1.

In [50]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [51]:
```
knn = KNeighborsClassifier(n_neighbors=4)
```

In [52]:
```
knn.fit(X_train,y_train)
```

Out[52]:
```
KNeighborsClassifier(n_neighbors=4)
```

In [53]:
```
pred = knn.predict(X_test)
pred
```

Out[53]:
```
array([1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0], dtype=int64)
```

Predictions and Evaluations

Let's evaluate our KNN model!

In [54]:
```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

In [55]:
```
print(confusion_matrix(y_test,pred))
```

```
[[143  10]
 [  7 140]]
```

In [56]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94       153
           1       0.93      0.95      0.94       147

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

Using KNN

In [19]:
```
# Will take some time
accuracy_score1 = []
for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train, y_train)
    y_predict = knn.predict(X_test)
    accuracy = accuracy_score(y_test,y_predict)
    accuracy_score1.append(accuracy)
    print("Accuracy Score=",i, "is",accuracy)
```

```
Accuracy Score= 1 is 0.9066666666666666
Accuracy Score= 2 is 0.9066666666666666
Accuracy Score= 3 is 0.9066666666666666
Accuracy Score= 4 is 0.9433333333333334
Accuracy Score= 5 is 0.9233333333333333
Accuracy Score= 6 is 0.9233333333333333
Accuracy Score= 7 is 0.9133333333333333
Accuracy Score= 8 is 0.9266666666666666
Accuracy Score= 9 is 0.9166666666666666
Accuracy Score= 10 is 0.93
Accuracy Score= 11 is 0.91
Accuracy Score= 12 is 0.9266666666666666
Accuracy Score= 13 is 0.9166666666666666
Accuracy Score= 14 is 0.9233333333333333
Accuracy Score= 15 is 0.92
Accuracy Score= 16 is 0.92
Accuracy Score= 17 is 0.9133333333333333
Accuracy Score= 18 is 0.9133333333333333
Accuracy Score= 19 is 0.9133333333333333
Accuracy Score= 20 is 0.9133333333333333
Accuracy Score= 21 is 0.9133333333333333
Accuracy Score= 22 is 0.9133333333333333
Accuracy Score= 23 is 0.9133333333333333
Accuracy Score= 24 is 0.9166666666666666
Accuracy Score= 25 is 0.9166666666666666
Accuracy Score= 26 is 0.91
Accuracy Score= 27 is 0.9133333333333333
Accuracy Score= 28 is 0.91
Accuracy Score= 29 is 0.9033333333333333
Accuracy Score= 30 is 0.9066666666666666
Accuracy Score= 31 is 0.9066666666666666
Accuracy Score= 32 is 0.9066666666666666
Accuracy Score= 33 is 0.9066666666666666
Accuracy Score= 34 is 0.91
Accuracy Score= 35 is 0.91
Accuracy Score= 36 is 0.9133333333333333
Accuracy Score= 37 is 0.9066666666666666
Accuracy Score= 38 is 0.9066666666666666
Accuracy Score= 39 is 0.9066666666666666
```

In [22]:
```
data = pd.DataFrame(accuracy_score1,columns=["accuracy"])
data
```

Out[22]:

	accuracy
0	0.906667
1	0.906667
2	0.906667
3	0.943333
4	0.923333
5	0.923333
6	0.913333
7	0.926667
8	0.916667
9	0.930000
10	0.910000
11	0.926667
12	0.916667
13	0.923333
14	0.920000
15	0.920000
16	0.913333
17	0.913333
18	0.913333
19	0.913333
20	0.913333
21	0.913333
22	0.913333
23	0.916667
24	0.916667
25	0.910000
26	0.913333
27	0.910000
28	0.903333
29	0.906667
30	0.906667
31	0.906667
32	0.906667

	accuracy
33	0.910000
34	0.910000
35	0.913333
36	0.906667
37	0.906667
38	0.906667

In [23]:
```
data["k_value"]= range(1,40)
```

In [24]:
```
data
```

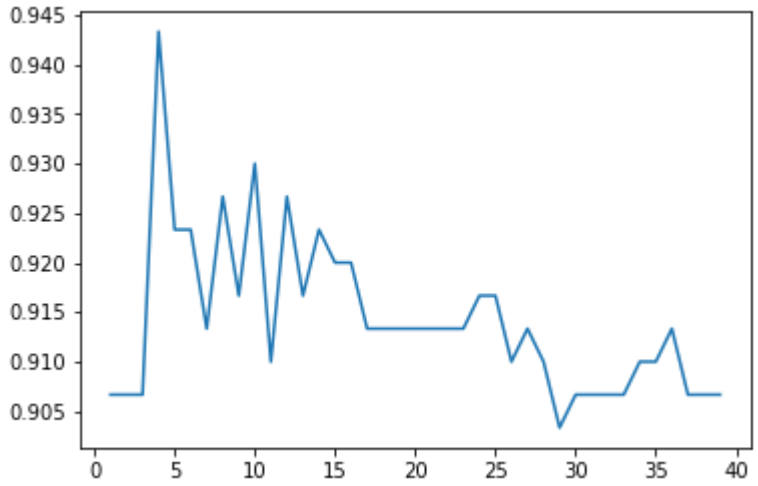
Out[24]:

	accuracy	k_value
0	0.906667	1
1	0.906667	2
2	0.906667	3
3	0.943333	4
4	0.923333	5
5	0.923333	6
6	0.913333	7
7	0.926667	8
8	0.916667	9
9	0.930000	10
10	0.910000	11
11	0.926667	12
12	0.916667	13
13	0.923333	14
14	0.920000	15
15	0.920000	16
16	0.913333	17
17	0.913333	18
18	0.913333	19
19	0.913333	20
20	0.913333	21
21	0.913333	22
22	0.913333	23
23	0.916667	24
24	0.916667	25
25	0.910000	26
26	0.913333	27
27	0.910000	28
28	0.903333	29
29	0.906667	30
30	0.906667	31
31	0.906667	32
32	0.906667	33
33	0.910000	34

	accuracy	k_value
34	0.910000	35
35	0.913333	36
36	0.906667	37
37	0.906667	38
38	0.906667	39

In [28]:
```
plt.plot(data["k_value"],data["accuracy"])
```

Out[28]:
```
[<matplotlib.lines.Line2D at 0x20e7e0b5b08>]
```



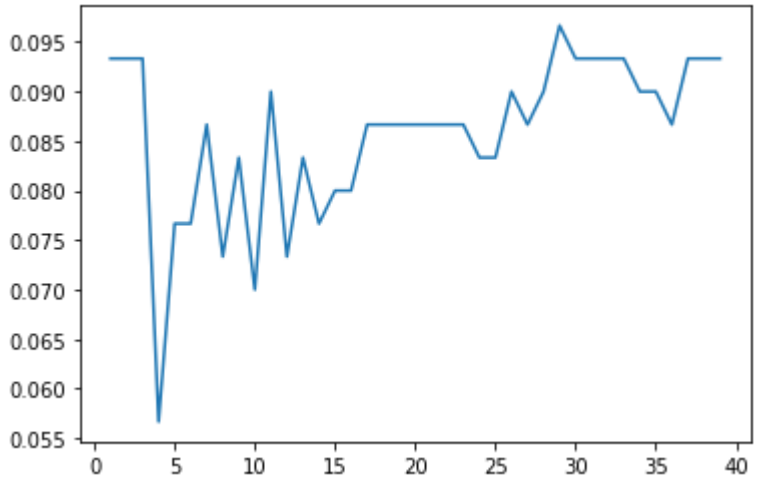
In [29]:
```
error_rate = []
# Will take some time
for i in range(1,40):

 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 [30]:
```
error_rate
```

Out[30]:
```
[0.09333333333333334,
 0.09333333333333334,
 0.09333333333333334,
 0.056666666666666664,
 0.07666666666666666,
 0.07666666666666666,
 0.08666666666666667,
 0.07333333333333333,
 0.08333333333333333,
 0.07,
 0.09,
 0.07333333333333333,
 0.08333333333333333,
 0.07666666666666666,
 0.08,
 0.08,
 0.08666666666666667,
 0.08666666666666667,
 0.08666666666666667,
 0.08666666666666667,
 0.08666666666666667,
 0.08666666666666667,
 0.08666666666666667,
 0.08333333333333333,
 0.08333333333333333,
 0.09,
 0.08666666666666667,
 0.09,
 0.09666666666666666,
 0.09333333333333334,
 0.09333333333333334,
 0.09333333333333334,
 0.09333333333333334,
 0.09,
 0.09,
 0.08666666666666667,
 0.09333333333333334,
 0.09333333333333334,
 0.09333333333333334]
```

In [31]:
```
plt.plot(range(1,40),error_rate)
```

Out[31]:
```
[<matplotlib.lines.Line2D at 0x20e7fa53f88>]
```



In [ ]:
```

```