02-K Nearest Neighbors Project

March 21, 2023

1 K Nearest Neighbors Project

1.1 Import Libraries

Import pandas,seaborn, and the usual libraries.

[1]:

1.2 Get the Data

** Read the 'KNN_Project_Data csv file into a dataframe **

[2]:

Check the head of the dataframe.

[23]:

[23]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MGJM	JHZC	TARGET CLASS
0	1636.670614	817.988525	2565.995189	358.347163	550.417491	1618.870897	2147.641254	330.727893	1494.878631	845.136088	0
1	1013.402760	577.587332	2644.141273	280.428203	1161.873391	2084.107872	853.404981	447.157619	1193.032521	861.081809	1
2	1300.035501	820.518697	2025.854469	525.562292	922.206261	2552.355407	818.676686	845.491492	1968.367513	1647.186291	1
3	1059.347542	1066.866418	612.000041	480.827789	419.467495	685.666983	852.867810	341.664784	1154.391368	1450.935357	0
4	1018.340526	1313.679056	950.622661	724.742174	843.065903	1370.554164	905.469453	658.118202	539.459350	1899.850792	0

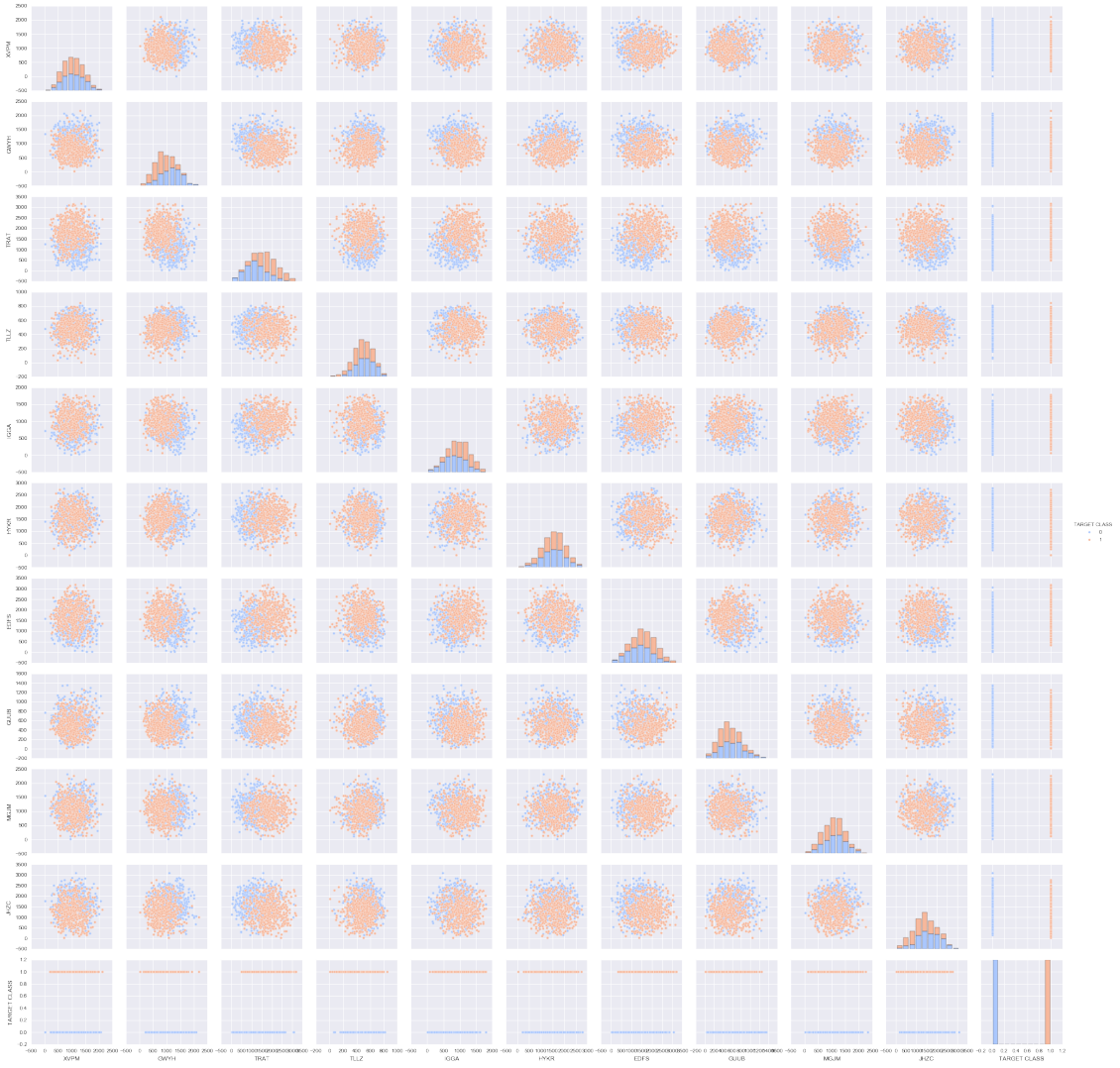
2 EDA

Since this data is artificial, we'll just do a large pairplot with seaborn.

Use seaborn on the dataframe to create a pairplot with the hue indicated by the TARGET CLASS column.

[4]:

[4]: <seaborn.axisgrid.PairGrid at 0x1197505f8>



3 Standardize the Variables

Time to standardize the variables.

** Import StandardScaler from Scikit learn.**

[5]:

** Create a StandardScaler() object called scaler.**

[6]:

** Fit scaler to the features.**

[7]:

[7]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

Use the .transform() method to transform the features to a scaled version.

[8]:

Convert the scaled features to a dataframe and check the head of this dataframe to make sure the scaling worked.

[9]:

[9]:
```
       XVPM      GWYH      TRAT      TLLZ      IGGA      HYKR      EDFS  \
0  1.568522 -0.443435  1.619808 -0.958255 -1.128481  0.138336  0.980493
1 -0.112376 -1.056574  1.741918 -1.504220  0.640009  1.081552 -1.182663
2  0.660647 -0.436981  0.775793  0.213394 -0.053171  2.030872 -1.240707
3  0.011533  0.191324 -1.433473 -0.100053 -1.507223 -1.753632 -1.183561
4 -0.099059  0.820815 -0.904346  1.609015 -0.282065 -0.365099 -1.095644

       GUUB      MGJM      JHZC
0 -0.932794  1.008313 -1.069627
1 -0.461864  0.258321 -1.041546
2  1.149298  2.184784  0.342811
3 -0.888557  0.162310 -0.002793
4  0.391419 -1.365603  0.787762
```

4 Train Test Split

Use train_test_split to split your data into a training set and a testing set.

[10]:

[11]:

5 Using KNN

Import KNeighborsClassifier from scikit learn.

[12]:

Create a KNN model instance with n_neighbors=1

[13]:

Fit this KNN model to the training data.

[14]:

[14]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

6 Predictions and Evaluations

Let's evaluate our KNN model!

Use the predict method to predict values using your KNN model and X_test.

[24]:

** Create a confusion matrix and classification report.**

[16]:

[17]:

```
[[112  40]
 [ 34 114]]
```

[18]:

```
             precision    recall  f1-score   support

          0       0.77      0.74      0.75       152
          1       0.74      0.77      0.75       148

avg / total       0.75      0.75      0.75       300
```

7 Choosing a K Value

Let's go ahead and use the elbow method to pick a good K Value!

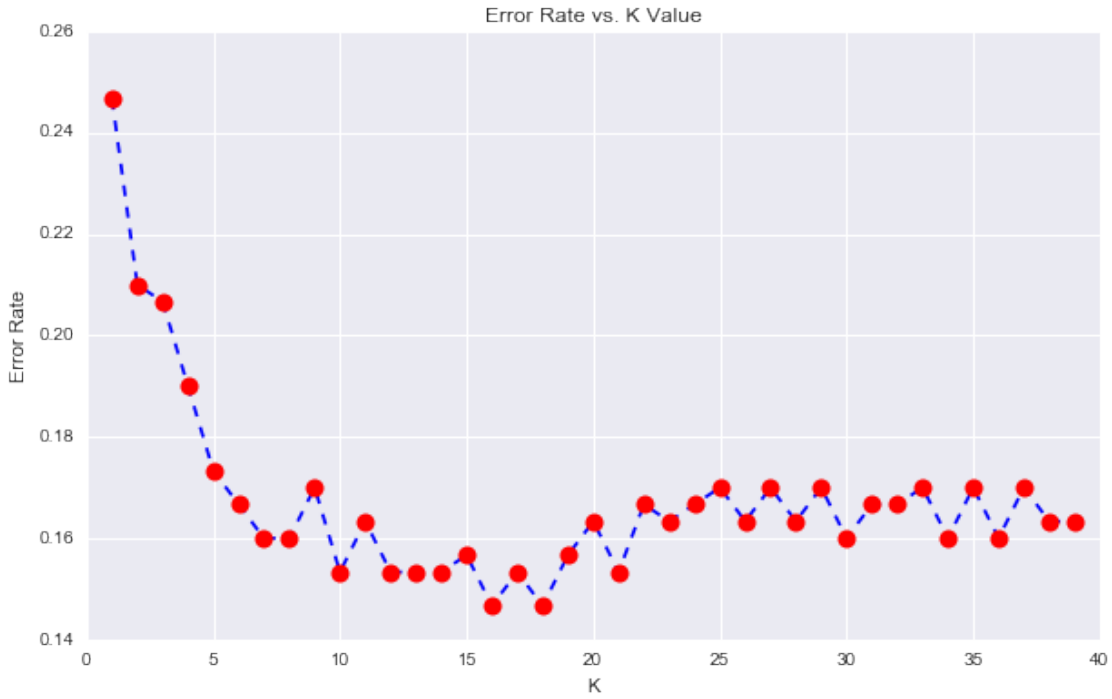
** Create a for loop that trains various KNN models with different k values, then keep track of the error_rate for each of these models with a list. Refer to the lecture if you are confused on this step.**

[25]:

Now create the following plot using the information from your for loop.

[20]:

[20]: <matplotlib.text.Text at 0x11cbdb710>



7.1 Retrain with new K Value

Retrain your model with the best K value (up to you to decide what you want) and re-do the classification report and the confusion matrix.

[21]:

```
WITH K=30


[[127  25]
 [ 23 125]]


             precision    recall  f1-score   support

          0       0.85      0.84      0.84       152
```

1	0.83	0.84	0.84	148
avg / total	0.84	0.84	0.84	300

8 Great Job!