K Nearest Neighbors with Python

May 27, 2019

1 K Nearest Neighbors with Python

You've been given a classified data set from a company! They've hidden the feature column names but have given you 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 off of the features.

Let's grab it and use it!

1.1 Import Libraries

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

1.2 Get the Data

Set index_col=0 to use the first column as the index.

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

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

1.3 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.

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

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

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

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

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

1.4 Train Test Split

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

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

1.5 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 with k=1.

```
In [ ]: frofrom sklearn.neighbors import KNeighborsClassifier
```

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

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

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

1.6 Predictions and Evaluations

Let's evaluate our KNN model!

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

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

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

1.7 Choosing a K Value

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

```
In [ ]: 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 [ ]: 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')
```

Here we can see that that after arouns K>23 the error rate just tends to hover around 0.06-0.05 Let's retrain the model with that and check the classification report!

```
In [ ]: # 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))
```

```
In [ ]: # 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))
```