Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

Read the guide

tonydiana1 / Data602 Private

Code Issues Pull requests Actions Projects Security Insights Settings

master Data602 / Copy_of_DATA_602_KNN_Models.ipynb Go to file

tonydiana1 Created using Colaboratory Latest commit c3573e1 on Mar 31 History

1 contributor

518 lines (518 sloc) 15.1 KB Raw Blame

Open in Colab
(https://colab.research.google.com/github/tonydiana1/Data602/blob/master/Copy_of_DATA_602_KNN_Models.ipynb)

EXAMPLE 1--KNN: How to Select K with Elbow or Scree Plot Method

In [0]:
```
# Selecting K in KNN ++++ The Elbow or Scree Plot Method ++++
import numpy as np
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt

cluster1 = np.random.uniform(0.5, 1.5, (2, 10))
cluster2 = np.random.uniform(3.5, 4.5, (2, 10))
X = np.hstack((cluster1, cluster2)).T

K = range(1, 10)
meandistortions = []
for k in K:
  kmeans = KMeans(n_clusters=k)
  kmeans.fit(X)
  meandistortions.append(sum(np.min(cdist(X, kmeans.cluster_centers_,
'euclidean'), axis=1)) / X.shape[0])
plt.plot(K, meandistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
plt.show()
```

KNN 2--KNN: Predicting Outcome

In [0]:
```
# Assigning features and label variables
# First Feature
weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast'
,'Sunny','Sunny',
'Rainy','Sunny','Overcast','Overcast','Rainy']

# Second Feature
temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mil
d','Mild','Mild','Hot','Mild']

# Label or target variable
play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Y
es','Yes','No']
```

The categorical columns need to be transformed into numerical columns.

To encode the data, we map each value to a number, for instance, 'Overcast'=0, 'Rainy'=1, and 'Sunny'=2.

This process is called label encoding. We use Label Encoder with sklearn to make that transformation.

In [0]:
```
# Import LabelEncoder
from sklearn import preprocessing
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
weather_encoded=le.fit_transform(weather)
print(weather_encoded)
```

```
[2 2 0 1 1 1 0 2 2 1 2 0 0 1]
```

In [0]:
```
# converting string labels into numbers
temp_encoded=le.fit_transform(temp)
label=le.fit_transform(play)
```

We combine multiple columns or features into a single set of data using the "zip" function.

In [0]:
```
#combinig weather and temp into single listof tuples
features=list(zip(weather_encoded,temp_encoded))
```

We build the KNN classifier model with sklearn.neighbors.

In [0]:
```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets
model.fit(features,label)

#Predict Output
predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
print(predicted)
```

```
[1]
```

The model predicted 'play'.

EXAMPLE 3--KNN: Let's say that we have multiple categories.

```
In [0]: #Import scikit-learn dataset library
        from sklearn import datasets

        #Load dataset
        wine = datasets.load_wine()
```

Let's explore the data

```
In [0]: # print the names of the features
        print("Feature Names:\n",wine.feature_names)
        # print the label species(class_0, class_1, class_2)
        print("Target Names:",wine.target_names)
        # print the wine data (top 5 records)
        print("Top Five Records: \n",wine.data[0:5])
        # print the wine labels (0:Class_0, 1:Class_1, 2:Class_3)
        print("Target Set:\n", wine.target)
        # print data(feature)shape
        print("Wine Data Shape:\n", wine.data.shape)
        # print target(or label)shape
        print("Print Target Data Shape:\n",wine.target.shape)
```

```
Feature Names:
 ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 't
otal_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanin
s', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'prolin
e']
Target Names: ['class_0' 'class_1' 'class_2']
Top Five Records:
 [[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e+
00
  2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 1.120e+01 1.000e+02 2.650e+00 2.760e+0
0
  2.600e-01 1.280e+00 4.380e+00 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 1.860e+01 1.010e+02 2.800e+00 3.240e+0
0
  3.000e-01 2.810e+00 5.680e+00 1.030e+00 3.170e+00 1.185e+03]
 [1.437e+01 1.950e+00 2.500e+00 1.680e+01 1.130e+02 3.850e+00 3.490e+0
0
  2.400e-01 2.180e+00 7.800e+00 8.600e-01 3.450e+00 1.480e+03]
 [1.324e+01 2.590e+00 2.870e+00 2.100e+01 1.180e+02 2.800e+00 2.690e+0
0
  3.900e-01 1.820e+00 4.320e+00 1.040e+00 2.930e+00 7.350e+02]]
Target Set:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
Wine Data Shape:
 (178, 13)
Print Target Data Shape:
 (178,)
```

Let's split the dataset

```
In [0]: # Import train_test_split function
        from sklearn.model_selection import train_test_split

        # Split dataset into training set and test set
        X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.ta
        rget, test_size=0.3) # 70% training and 30% test
```

Let's build the classifier model.

```
In [0]: #Import knearest neighbors Classifier model
        from sklearn.neighbors import KNeighborsClassifier

        #Create KNN Classifier
        knn = KNeighborsClassifier(n_neighbors=5)

        #Train the model using the training sets
        knn.fit(X_train, y_train)

        #Predict the response for test dataset
        y_pred = knn.predict(X_test)
```

What is the model accuracy?

```
In [0]: #Import scikit-learn metrics module for accuracy calculation
        from sklearn import metrics
        # Model Accuracy, how often is the classifier correct?
        print("Accuracy:",metrics.accuracy_score(y_test, y_pred).round(3))
```

```
Accuracy: 0.722
```

Let's try K=7