naive_bayes_total_exercise

June 3, 2019

0.0.1 simple naive bayes

In [ ]:
```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
```

In [ ]:
```
viagra_spam = {
    'viagra': [1,0,0,0,0,0,0,0,1,1,1,0,0,1,0,0,0,0,0,1],
    'spam': [
        1,0,0,0,0,0,1,0,1,0, 0,0,0,0,0,0,0,1,1,1
              ]}
df = pd.DataFrame(viagra_spam, columns = ['viagra', 'spam'])
df.head()
```

In [ ]:
```
np_data = df.as_matrix()
np_data[:3]
```

In [ ]:
```
# P(Viagra)
p_viagra = sum(np_data[:, 0] == 1) / len(np_data)
p_spam = sum(np_data[:, 1] == 1) / len(np_data)
p_v_cap_s = sum((np_data[:, 0] == 1) & \
                (np_data[:, 1] == 1)) / len(np_data)
p_n_v_cap_s = sum((np_data[:, 0] == 0) & \
                  (np_data[:, 1] == 1)) / len(np_data)
```

In [ ]:
```
# P(spam | viagra)
p_spam * (p_v_cap_s / p_spam ) / p_viagra
```

In [ ]:
```
# P(spam | ~viagra)
p_spam * (p_n_v_cap_s / p_spam ) / (1-p_viagra)
```

0.0.2 german_credit_application

In [ ]:
```
data_url = "./fraud.csv"
df= pd.read_csv(data_url, sep=',')
df.head(10)
```

```
In [ ]: del df["ID"]
        Y_data = df.pop("Fraud")
        Y_data = Y_data.as_matrix()
        Y_data

In [ ]: df.head()

In [ ]: x_df = pd.get_dummies(df)
        x_df.head()

In [ ]: x_data = x_df.as_matrix()
        x_data[:3]

In [ ]: Y_data == True

In [ ]: P_Y_True = sum(Y_data==True) / len(Y_data)
        P_Y_False = 1 - P_Y_True

        P_Y_True,P_Y_False

In [ ]: ix_Y_True = np.where(Y_data)
        ix_Y_False = np.where(Y_data==False)

        ix_Y_True, ix_Y_False

In [ ]: p_x_y_true = (x_data[ix_Y_True].sum(axis=0)) / sum(Y_data==True)
        p_x_y_false = (x_data[ix_Y_False].sum(axis=0)) / sum(Y_data==False)

        p_x_y_true, p_x_y_false

In [ ]: x_test = [0,1,0,0,0,1,0, 0,1,0]

        p_y_true_test = P_Y_True + p_x_y_true.dot(x_test)
        p_y_false_test = P_Y_False + p_x_y_false.dot(x_test)

        p_y_true_test , p_y_false_test

In [ ]: p_y_true_test < p_y_false_test
```

0.0.3 Sklearn Naive Bayes

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

        temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild',\
                              'Cool','Mild','Mild','Mild','Hot','Mild']

        play=['No','No','Yes','Yes','Yes','No','Yes','No',\
                              'Yes','Yes','Yes','Yes','Yes','No']
```

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

In [ ]: temp_encoded=le.fit_transform(temp)
        label=le.fit_transform(play)
        print( "Temp:",temp_encoded)
        print( "Play:",label)

In [ ]: # features=zip(weather_encoded,temp_encoded)
        # features_array = np.array(features)
        # features_array

        features_array = pd.DataFrame({'temp': temp_encoded, \
                                'weather':weather_encoded}).as_matrix()

In [ ]: from sklearn.naive_bayes import MultinomialNB

        #Create a Gaussian Classifier
        model = MultinomialNB()

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

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

In [ ]: # wine

In [ ]: from sklearn import datasets

        #Load dataset
        wine = datasets.load_wine()

In [ ]: print ("Features: ", wine.feature_names)

        # print the label type of wine(class_0, class_1, class_2)
        print ("Labels: ", wine.target_names)

In [ ]: wine.data.shape

In [ ]: print (wine.data[0:5])

In [ ]: print (wine.target)
```

```
In [ ]: # quiz

        # wine data niave bayes
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```