Pranav Nair
2019130042
TE Comps Batch – C
8th November, 2021

Experiment-4

Aim: To train a machine learning model using naïve bayes algorithm to classify whether a patient has a high chance of having diabetes or not.

Code:

```
import pandas as pd
import numpy as np

diabetes = pd.read_csv('diabetes.csv')
diabetes = diabetes.dropna()
print(diabetes)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test =
train_test_split(diabetes,diabetes['Outcome'],test_size=0.2)

df = x_train.copy()
diabetes_yes = df.where(df['Outcome']==1).dropna()
diabetes_yes_prob = len(diabetes_yes)/len(df)
diabetes_no = df.where(df['Outcome']==0).dropna()
diabetes_no_prob = len(diabetes_no)/len(df)

print(diabetes_yes_prob,diabetes_no_prob)

columns = df.iloc[:,:6].columns.to_list()
classes = 15
column_details = []
for column_name in columns:
    min_val = min(df[column_name].to_list())
    max_val = max(df[column_name].to_list())
    class_size = round((max_val-min_val)/classes , 2)

    low_bounds = list(np.arange(min_val,max_val,class_size))
    for i in range(len(low_bounds)):
        low_bounds[i] = round(low_bounds[i],2)
    x = low_bounds[-1]
    low_bounds.append(round(x + class_size,2))
    # print(low_bounds)
    # mid_class = round((class_size/2) , 2)
    dict = {}
```

```
    # start_mid = mid_class
    # df_temp = df[column_name]
    # print(df_temp)
    # print(len(diabetes_yes) , len(diabetes_no))

    for i in range(classes):
#         # print(start_mid)
        low_bound = low_bounds[i]
        upp_bound = low_bounds[i+1]

        yes_count =
diabetes_yes.where(diabetes_yes[column_name]>=low_bound).dropna()
        if i<classes-1:
            yes_count =
yes_count.where(yes_count[column_name]<upp_bound).dropna()
        else:
            yes_count =
yes_count.where(yes_count[column_name]<=upp_bound).dropna()

        no_count =
diabetes_no.where(diabetes_no[column_name]>=low_bound).dropna()
        if i<classes-1:
            no_count =
no_count.where(no_count[column_name]<upp_bound).dropna()
        else:
            no_count =
no_count.where(no_count[column_name]<=upp_bound).dropna()


#         # print(len(yes_count)/len(diabetes_yes))
#         # print(len(yes_count)/len(diabetes_yes) ,
len(no_count)/len(diabetes_no))

        dict[low_bounds[i]] = [len(yes_count)/len(diabetes_yes) ,
len(no_count)/len(diabetes_no)]
#         # print([low_bound,upp_bound])
#         start = round((start_mid + class_size),2)

    column_details.append([column_name, dict , class_size])
print(column_details)

def get_outcome(input_data):
    probabilities = []

    for i in range(6):
        value = input_data[i]
        prob_dict = column_details[i][1]
        class_range = column_details[i][2]
```

```
        key_list = list(prob_dict.keys())

        for key in key_list:
            low = key
            up = key+class_range

            if key_list[-1] == key:
                if value>=low and value <= up:
                    probabilities.append(prob_dict[key])
                    break
            else:
                if value>=low and value < up:
                    probabilities.append(prob_dict[key])
                    break

    yes=1
    no=1
    for i in probabilities:
        if i[0] == 0 and i[1]==0:
            continue
        yes=yes*i[0]
        no=no*i[1]

    if yes==1 or no==1:
        return 0

    product = [yes*diabetes_yes_prob , no*diabetes_no_prob]
    total = yes*diabetes_yes_prob + no*diabetes_no_prob


    outcome = [product[0]/total, product[1]/total]

    if outcome[0] > outcome[1]:
        return 1
    else:
        return 0

test = x_test.iloc[:,:6].copy()
test_set = []
for i in range(154):
    test_set.append(test.iloc[i].to_list())

by_nb = []
for i in test_set:
    by_nb.append(get_outcome(i))

by_dataset = y_test.to_list()
correct = 0
```

```
for i in range(len(by_nb)):
    if by_nb[i]==by_dataset[i]:
        correct = correct + 1

print(correct/len(by_nb)*100)

print(x_test)

print("Enter values for the following columns:\n" + ", ".join(columns) + "\n")
user_input  = input()
user_input = user_input.split(' ')
input_data = [float(i) for i in user_input]
print(input_data)
get_outcome(input_data)
```

Conclusion: In this experiment, the aim is to implement the Naive Bayes Algorithm to train a machine learning model which helps to predict whether a patient is highly likely to be diabetic or not. The dataset used for the preparation of the model, is taken from Kaggle namely, Pima Indians Diabetes Database. The dataset contains numerous patient's medical details like Glucose, Blood Pressure, Insulin, Skin Thickness, Age, etc. To train the model, the data set was split into an 80:20 ratio and was trained on the 80% of data and the rest was kept as test data. Using the Naive Bayes Algorithm, the probabilities for each individual feature were calculated. Then using the test data, I calculated the accuracy of the model trained which came around 70-80%. An error observed in the model was, for certain values present in the test data and absent from the data used for training the model, the probability assigned was zero and the prediction couldn't be made. For such cases, the model needed to be trained again with these data included in training set.

GitHub: https://github.com/pranav567/AI-ML-Lab/tree/master/experiment-4