داده کاوی

تمرین سری دوم

بخش عملی

سوال ۱- درخت تصمیم گیری)

در ابتدا بازه های متغیر های عددی را پیدا کردم و بعد با استفاده از تابع data_pre_processing تبدیل داده عددی به فهرستی را انجام دادم.

```
----------------------------------------------------------
age range: 17 90
----------------------------------------------------------
income
----------------------------------------------------------
hours range: 1 99
----------------------------------------------------------
capital loss range: 0 4356
----------------------------------------------------------
capital gain range: 0 99999
----------------------------------------------------------
education num range: 1 16
----------------------------------------------------------
fnlwgt range: 12285 1484705
```

در شکل های زیر توابع مربوط به تبدیل فهرست عددی به اسمی را مشاهده میکنیم

```python
#function below is for coverting age numbers to names
def age_modification(data):
    for i in range(len(data)):
        if data['age'][i] < 10:
            data['age'][i] = 'kid'
        elif 10 <= data['age'][i] < 20:
            data['age'][i] = 'teenage'
        elif 20 <= data['age'][i] < 30:
            data['age'][i] = 'teen'
        elif 30 <= data['age'][i] < 40:
            data['age'][i] = 'mature'
        elif 40 <= data['age'][i] < 50:
            data['age'][i] = 'middle aged'
        elif 50 <= data['age'][i] < 70:
            data['age'][i] = 'old'
        else:
            data['age'][i] = 'phosil'
#function below is for coverting income numbers to names
def income_modification(data):
    for i in range(len(data)):
        if data['income'][i] == '<=50K':
            data['income'][i] = 'low'
        else:
            data['income'][i] = 'high'
#function below is for coverting hours per week numbers to names
def hour_modification(data):
    for i in range(len(data)):
        if int(data['hours-per-week'][i]) < 10:
           data['hours-per-week'][i] = 'lazy'
        elif 10 <= int(data['hours-per-week'][i]) < 30:
            data['hours-per-week'][i] = 'semi lazy'
        elif 30 <= int(data['hours-per-week'][i]) < 50:
            data['hours-per-week'][i] = 'middle working'
        elif 50 <= int(data['hours-per-week'][i]) < 70:
            data['hours-per-week'][i] = 'working'
        else:
            data['hours-per-week'][i] = 'hard working'
```

```
#function below is for coverting capital-loss numbers to names
def capital_loss_modification(data):
    for i in range(len(data)):
        if int(data['capital-loss'][i]) < 1000:
            data['capital-loss'][i] = 'very low'
        elif 1000 <= int(data['capital-loss'][i]) < 2000:
            data['capital-loss'][i] = 'low'
        elif 2000 <= int(data['capital-loss'][i]) < 3000:
            data['capital-loss'][i] = 'medium'
        elif 3000 <= int(data['capital-loss'][i]) < 4000:
            data['capital-loss'][i] = 'high'
        else:
            data['capital-loss'][i] = 'very high'
#function below is for coverting capital-gain numbers to names
def capital_gain_modification(data):
    for i in range(len(data)):
        if int(data['capital-gain'][i]) < 20000:
            data['capital-gain'][i] = 'very low'
        elif 20000 <= int(data['capital-gain'][i]) < 40000:
            data['capital-gain'][i] = 'low'
        elif 4000 <= int(data['capital-gain'][i]) < 60000:
            data['capital-gain'][i] = 'medium'
        elif 60000 <= int(data['capital-gain'][i]) < 8000:
            data['capital-gain'][i] = 'high'
        else:
            data['capital-gain'][i] = 'very high'
#function below is for coverting eduacation_num numbers to names
def eduaction_num_modification(data):
    for i in range(len(data)):
        if int(data['education-num'][i]) <= 4:
            data['education-num'][i] = 'first quarter'
        elif 4 < int(data['education-num'][i]) <= 8:
            data['education-num'][i] = 'second querter'
        elif 8 < int(data['education-num'][i]) <=12:
            data['education-num'][i] = 'third quarter'
        else:
            data['education-num'][i] = 'last quarter'
```

```
def eduaction_num_modification(data):
    for i in range(len(data)):
        if int(data['education-num'][i]) <= 4:
            data['education-num'][i] = 'first quarter'
        elif 4 < int(data['education-num'][i]) <= 8:
            data['education-num'][i] = 'second querter'
        elif 8 < int(data['education-num'][i]) <=12:
            data['education-num'][i] = 'third quarter'
        else:
            data['education-num'][i] = 'last quarter'
#function below is for coverting fnlwgt numbers to names
def fnlwgt_modification(data):
    for i in range(len(data)):
        if int(data['fnlwgt'][i]) < 25000:
            data['fnlwgt'][i] = 'very low'
        elif 25000 <= int(data['fnlwgt'][i]) < 50000:
            data['fnlwgt'][i] = 'low'
        elif 50000 <= int(data['fnlwgt'][i]) < 75000:
            data['fnlwgt'][i] = 'medium'
        elif 75000 <= int(data['fnlwgt'][i]) < 100000:
            data['fnlwgt'][i] = 'high'
        else:
            data['fnlwgt'][i] = 'very high'
#function below is for modification all numerical attributes for known data
def data_pre_processing(data,mode):
    if mode == 'known':
        age_modification(data)
        income_modification(data)
        hour_modification(data)
        capital_loss_modification(data)
        capital_gain_modification(data)
        eduaction_num_modification(data)
        fnlwgt_modification(data)
    elif mode = 'unknown':
        age_modification(data)
        hour_modification(data)
        capital_loss_modification(data)
        capital_gain_modification(data)
        eduaction_num_modification(data)
        fnlwgt_modification(data)
    else:
        print('unsupported mode')
```

با استفاده از توابع بالا ما پیش پردازش داده را انجام میدهیم و در نهایت با استفاده از کتابخانه pickle ما عملیات ذخیره سازی را انجام میدهیم.

```
#saving modified known data
pickle.dump(knownData,open('knownP1','wb'))
pickle.dump(unknownData,open('unknownP1','wb'))
```

حال به پردازش داده اصلاح شده میپردازیم.

با استفاده از کتابخانه pickle داده های ذخیره شده را که اصلاح کرده بودیم میخوانیم و به داده های ناشناخته ستون درامد را اضافه میکنیم.

```
#reading data using pickle
knownData = pickle.load(open('knownP1','rb'))
unknownData = pickle.load(open('unknownP1','rb'))
#adding income column to unknown data
unknownData['income'] = ['N']*6512
```

با استفاده از sklearn.preprocessing داده ها را یک دور دیگر برای استفاده‌ی درخت تصمیم پیش پردازش میکنیم.

```
#using lable encoder for known and unknown data
lb = LabelEncoder()
for i in knownData:
    knownData[i] = lb.fit_transform(knownData[i])
    unknownData[i] = lb.fit_transform(unknownData[i])
```

با استفاده از کتابخانه sklearn ما tree را import میکنیم.

در شکل زیر دو نوع درخت میسازیم یکی با معیار جینی دیگری با معیار انتروپی و ویژگی هارا در یک آرایه نگهداری میکنیم.

```
decisionTree1 = tree.DecisionTreeClassifier(criterion = 'gini')
decisionTree2 = tree.DecisionTreeClassifier(criterion = 'entropy')
#saving features names
features = ['age','workclass','fnlwgt','education','education-num','marital-status',
            'occupation','relationship','race','sex','capital-gain','capital-loss',
            'hours-per-week','native-country']
```

۸۰ درصد داده ها برای اموزش است و ۲۰ درصد برای تست که در واقع ۲۰۸۳۹ از داده ها برای یادگیری و الباقی برای تست میباشد.

حال با استفاده از کد زیر پیش بینی را انجام میدهیم روی داده تست و دقت را محاسبه میکنیم

```
accuracy1 = 0
accuracy2 = 0
for i in range(len(testData)):
    predicted1 = decisionTree1.predict([testData[features].iloc[i]])
    predicted2 = decisionTree2.predict([testData[features].iloc[i]])
    real = testData['income'].iloc[i]
    if predicted1[0] == real:
        accuracy1 += 1
    if predicted2[0] == real:
        accuracy2 += 1
print('accuracy using gini:',accuracy1/len(testData))
print('accuracy using entropy:',accuracy2/len(testData))
```

```
accuracy using gini: 0.7856046065259117
accuracy using entropy: 0.7907869481765835
```

از انجایی که دقت برای انتروپی بیشتر است داده های ناشناخته را با درخت دوم پیش بینی میکنیم.

```
#predicting unknown data using tree2
for i in range(len(unknownData)):
    predicted = decisionTree2.predict([unknownData[features].iloc[i]])
    unknownData['income'][i] = predicted
print(unknownData)
```

```
      age  workclass  fnlwgt  ...  hours-per-week  native-country  income
0       4          4       1  ...               2              38       1
1       5          0       3  ...               3              38       1
2       1          6       3  ...               0              21       1
3       2          5       3  ...               2              38       0
4       1          4       3  ...               2              38       0
...   ...        ...     ...  ...             ...             ...     ...
6507    0          4       3  ...               4              38       1
6508    4          4       3  ...               2              38       1
6509    5          4       3  ...               1              38       1
6510    2          7       1  ...               2              38       0
6511    2          4       3  ...               2              38       0

[6512 rows x 15 columns]
```

سوال ۲-KNN)

همانند قست قبلی بازه های داده های عددی را پیدا میکنیم و آن هارا تبدیل به داده های فهرستی میکنیم.

با چک کردن داده ها متوجه میشویم که همه آن ها فهرستی هستند و داده عددی نداریم برای همین به سراغ پیاده سازی الگوریتم میرویم.

از آنجایی که knn یک الگوریتم lazy learning است نیازی به اموزش ندارد و کل هزینه را در زمان تست متحمل میشویم.

۸۰ درصد از داده هارا برای مقایسه به کار میبریم و ۲۰ درصد از داده هارا برای تست

الگوریتم مورد استفاده توی این سوال به این گونه است که هر داده تست را با تک تک داده های های اکوزش مقایسه میکنم و معیار شباهت یه این صورت است که هر ویژگی که از داده تست با آموزش یکی باشد یکی به مقدار شباهت اضافه میشود.و بر اساس k های مختلف k تای برتر رای گیری میکنند و لیبل را مشخص میسازند.

تعداد کل داده ها ۶۴۹۹ میباشد که ۵۲۰۰ برای اموزش و ۱۲۹۹ برای تست استفاده میشود.

کتاب خانه های مورد نظر را ایمپورت میکنیم و داده هارا میخوانیم

```
import pandas as pd
import matplotlib.pyplot as plt
import sys
#reading data from data sets
knownData = pd.read_csv('Dataset2.csv')
unknownData = pd.read_csv('Dataset2_Unknown.csv')
```

تابع محاسبه شباهت زیر برای مقایسه شباهت دو سطر از داده ما است به صورت که شباهت برابر است با تعداد ویژگی های یکسان

```
#function below is for calculating similarity between two rows
def similarity_calculate(data,i,j):
    similarity = 0
    for k in data:
        if data.loc[i][k] == data.loc[j][k]:
            similarity += 1
    return similarity
```

کلاس زیر برای نگهداری میزان شباهت و لیبل میباشد

```
#class below is for saving lable and similarity
class save():
    def __init__(self,similarity = 0,lable = ''):
        self.similarity = similarity
        self.lable = lable
    def __str__(self):
        return str(self.similarity)+' '+self.lable
    def __lt__(self,obj):
        return self.similarity > obj.similarity
```

تابع زیر برای پیش بینی داده میباشد بر اساس k داده شده

```
#function below is for predicting a lable
def predict_lable(saveList,k):
    e = 0
    p = 0
    for i in range(len(saveList)-k,len(saveList)):
        if saveList[i].lable == 'e':
            e += 1
        else:
            p += 1
    if e > p:
        return 'e'
    return 'p'
```

دلیل استفاده از merge sort این است که بار محاسبات کمتر شود این کار برای مرتب کردن لیستی از سیو هاست به این صورت که به صورت نزولی بر اساس شباهت مرتب میشوند

```
#functions below is for sorting saveList using merge sort
def merge(saveList,start,middle,end):
    n1 = middle-start+1
    n2 = end-middle
    left = []
    right = []
    for i in range(0,n1):
        left.append(saveList[start+i])
    for j in range(0,n2):
        right.append(saveList[middle+j+1])
    left.append(save(sys.maxsize,'e'))
    right.append(save(sys.maxsize,'e'))
    i = 0
    j = 0
    for k in range(start,end+1):
        if left[i].similarity <= right[j].similarity:
            saveList[k] = left[i]
            i += 1
        else:
            saveList[k] = right[j]
            j += 1
def merge_sort(saveList,start,end):
    if start < end:
        middle = (start+end)//2
        merge_sort(saveList, start, middle)
        merge_sort(saveList, middle+1, end)
        merge(saveList, start, middle, end)
```

محاسبه و چاپ کردن دقت بر اساس ۳ مقدار k

```
#function below is for testing our knn algorithm
def check_accuracy(data,train_numbers,test_numbers):
    accuracy_k1 = 0
    accuracy_k3 = 0
    accuracy_k5 = 0
    #iterating test data
    for i in range(train_numbers+1,len(data)):
        saveList = []
        for j in range(train_numbers):
            similarity = similarity_calculate(data, i, j)
            lable = data.loc[j]['poisonous']
            saveList.append(save(similarity,lable))
        merge_sort(saveList, 0, len(saveList)-1)
        predictedLable_k1 = predict_lable(saveList, 1)
        predictedLable_k3 = predict_lable(saveList, 3)
        predictedLable_k5 = predict_lable(saveList, 5)
        if predictedLable_k1 == data.loc[i]['poisonous']:
            accuracy_k1 += 1
        if predictedLable_k3 == data.loc[i]['poisonous']:
            accuracy_k3 += 1
        if predictedLable_k5 == data.loc[i]['poisonous']:
            accuracy_k5 += 1
    return accuracy_k1/test_numbers,accuracy_k3/test_numbers,accuracy_k5/test_numbers
accuracy_k1,accuracy_k3,accuracy_k5 = check_accuracy(knownData, 5200, 1299)
print('k = 1','accuracy:',accuracy_k1)
print('k = 3','accuracy:',accuracy_k3)
print('k = 5','accuracy:',accuracy_k5)
```

دو تابع زیر نیز برای محاسبه شباهت و پیش بینی میباشند اما با این تفاوت که بین دو داده مختلف میباشد

```
#function below is for calculating similarity between two rows of two differnt data sets
def similarity_calculate2(data1,data2,i,j):
    similarity = 0
    for k in data2:
        if data1.loc[i][k] == data2.loc[j][k]:
            similarity += 1
    return similarity
#function below is for predicting unknown data lables based on known data
def predict(data1,data2,number,k):
    predictList = []
    for i in range(number):
        saveList = []
        for j in range(len(data1)):
            similarity  = similarity_calculate2(data1, data2, j, i)
            lable = data1.loc[j]['poisonous']
            saveList.append(save(similarity,lable))
        merge_sort(saveList,0,len(saveList)-1)
        predictedLable = predict_lable(saveList, k)
        predictList.append(predictedLable)
    return predictList
```

در نهایت چاپ کردن نتیجه به ازای ۳ مقدار مختلف k

```
print('k=1')
print(predict(knownData, unknownData, len(unknownData), 1))
print('-------------------------------------------')
print('k=3')
print(predict(knownData, unknownData, len(unknownData), 3))
print('-------------------------------------------')
print('k=5')
print(predict(knownData, unknownData, len(unknownData), 5))
```

سوال ۳-بیض ساده)

در قدم اول باید پیش پردازش داده را انجام دهیم به این صورت که داده های عددی را تبدیل به داده های فهرستی میکنیم

از توابع زیر برای انجام این کار استفاده میکنیم.

```python
def modify_age(data):
    for i in range(len(data)):
        if int(data['age'][i]) <= 20:
            data['age'][i] = 'A'
        elif 20 < int(data['age'][i]) <= 40:
            data['age'][i] = 'B'
        elif 40 < int(data['age'][i]) <= 60:
            data['age'][i] = 'C'
        else:
            data['age'][i] = 'D'
def modify_sex(data):
    for i in range(len(data)):
        if int(data['sex'][i]) == 0:
            data['sex'][i] = 'female'
        else:
            data['sex'][i] = 'male'
def modify_cp(data):
    for i in range(len(data)):
        if int(data['cp'][i]) == 0:
            data['cp'][i] = 'cp0'
        elif int(data['cp'][i]) == 1:
            data['cp'][i] = 'cp1'
        elif int(data['cp'][i]) == 2:
            data['cp'][i] = 'cp2'
        else:
            data['cp'][i] = 'cp3'
def modify_trestbps(data):
    for i in range(len(data)):
        if int(data['trestbps'][i]) <= 130:
            data['trestbps'][i] = 'very low'
        elif 130 < int(data['trestbps'][i]) <= 150:
            data['trestbps'][i] = 'low'
        elif 150 < int(data['trestbps'][i]) <= 160:
            data['trestbps'][i] = 'medium'
        else:
            data['trestbps'][i] = 'high'
```

```python
def modify_chol(data):
    for i in range(len(data)):
        if int(data['chol'][i]) <= 150:
            data['chol'][i] = 'very low'
        elif 150 < int(data['chol'][i]) <= 300:
            data['chol'][i] = 'low'
        elif 300 < int(data['chol'][i]) <= 450:
            data['chol'][i] = 'medium'
        else:
            data['chol'][i] = 'high'
def modify_fbs(data):
    for i in range(len(data)):
        if int(data['fbs'][i]) == 0:
            data['fbs'][i] = 'zero'
        else:
            data['fbs'][i] = 'one'
def modify_restecg(data):
    for i in range(len(data)):
        if int(data['restecg'][i]) == 0:
            data['restecg'][i] = 'restecg0'
        elif int(data['restecg'][i]) == 1:
            data['restecg'][i] = 'restecg1'
        else:
            data['restecg'][i] = 'restecg2'
def modify_thalach(data):
    for i in range(len(data)):
        if int(data['thalach'][i]) <= 90:
            data['thalach'][i] = 'very low'
        elif 90 < int(data['thalach'][i]) <= 120:
            data['thalach'][i] = 'low'
        elif 120 < int(data['thalach'][i]) <= 150:
            data['thalach'][i] = 'medium'
        else:
            data['thalach'][i] = 'high'
def modify_exang(data):
    for i in range(len(data)):
        if int(data['exang'][i]) == 0:
            data['exang'][i] = 'zero'
        else:
            data['exang'][i] = 'one'
```

```python
def modify_oldpeak(data):
    for i in range(len(data)):
        if float(data['oldpeak'][i]) <= 1.25:
            data['oldpeak'][i] = 'very low'
        elif 1.25 < float(data['oldpeak'][i]) <= 2.5:
            data['oldpeak'][i] = 'low'
        elif 2.5 < float(data['oldpeak'][i]) <= 3.75:
            data['oldpeak'][i] = 'medium'
        else:
            data['oldpeak'][i] = 'high'
def modify_slope(data):
    for i in range(len(data)):
        if int(data['slope'][i]) == 0:
            data['slope'][i] = 'slope0'
        elif int(data['slope'][i]) == 1:
            data['slope'][i] = 'slope1'
        else:
            data['slope'][i] = 'slope2'
def modify_ca(data):
    for i in range(len(data)):
        if int(data['ca'][i]) == 0:
            data['ca'][i] = 'ca0'
        elif int(data['ca'][i]) == 1:
            data['ca'][i] = 'ca1'
        elif int(data['ca'][i]) == 2:
            data['ca'][i] = 'ca2'
        elif int(data['ca'][i]) == 3:
            data['ca'][i] = 'ca3'
        else:
            data['ca'][i] = 'ca4'
```

```
def modify_thal(data):
    for i in range(len(data)):
        if int(data['thal'][i]) == 0:
            data['thal'][i] = 'thal0'
        elif int(data['thal'][i]) == 1:
            data['thal'][i] = 'thal1'
        elif int(data['thal'][i]) == 2:
            data['thal'][i] = 'thal2'
        else:
            data['thal'][i] = 'thal3'
def modify_disease(data):
    for i in range(len(data)):
        if int(data['disease'][i]) == 1:
            data['disease'][i] = 'y'
        else:
            data['disease'][i] = 'n'
```

هرکدام از توابع بالا برای تبدیل داده عددی به فهرستی یک ستون از جدول هستند

```
def modify_data(data,mode):
    modify_age(data)
    modify_ca(data)
    modify_chol(data)
    modify_cp(data)
    modify_exang(data)
    modify_fbs(data)
    modify_oldpeak(data)
    modify_restecg(data)
    modify_sex(data)
    modify_slope(data)
    modify_thal(data)
    modify_thalach(data)
    modify_trestbps(data)
    if mode == 'known':
        modify_disease(data)
```

و تابع بالا کار نهایی اصلاح داده را انجام میدهد که اگر داده شناخته شده باشد مقدار ستون disease را نیز اصلاح میکند.(توجه شود که داده های ناشناخته ستون disease را ندارد لذا دستی این ستون را اضافه کردم.)

```
#adding new disese column to unknown data
unknownData['disease'] = ['u']*len(unknownData)
```

حال باید از ۱۹۳ داده اول برای یادگیری مدل استفاده کنیم و بقیه داده ها برای تست با استفاده از قطعه کد زیر تعداد و احتمال حالت های بله/خیر بیماری را محاسبه میکنیم.

```
diseaseCount = knownData['disease'][:193].value_counts()
print(diseaseCount)
print('---------------------------------')
diseasePositiveNum = diseaseCount[0]
diseaseNegativeNum = diseaseCount[1]
p_diseasePositive = diseaseCount[0]/193
p_diseaseNegative = diseaseCount[1]/193
```

```
y    138
n     55
Name: disease, dtype: int64
```

برای اعمال شرط با محدودیت روی دو ستون از دستور groupby استفاده میکنیم.

و مشاهده میکنیم که حالت های مختلف هر ستون به ازای y/n بودن ستون بیماری چه تعداد تکرار دارد و با استفاده از آن احتمال شرطی را مینیویسیم.

در شکل های زیر نمونه ای از محاسبات را مشاهده میکنیم.

```
#calculating p(coulumn|disease) in dictionary the form is {(column,diseaseStatus):count/len}
ageCount = knownData[:193].groupby(['age','disease']).size()
print(ageCount)
p_age = {('A','y'):0,('A','n'):0,('B','y'):11/diseasePositiveNum,('B','n'):1/diseaseNegativeNum,('C','y
```

```
---------------------------
age  disease
B    n           1
     y          11
C    n          35
     y          96
D    n          19
     y          31
dtype: int64
```

```
chol_count = knownData[:193].groupby(['chol','disease']).size()
print(chol_count)
p_chol = {('very low','y'):3/diseasePositiveNum,('very low','n'):0/diseaseNegativeNum,
          ('low','y'):114/diseasePositiveNum,('low','n'):46/diseaseNegativeNum,
          ('medium','y'):20/diseasePositiveNum,('medium','n'):9/diseaseNegativeNum,
          ('high','y'):1/diseasePositiveNum,('high','n'):0/diseaseNegativeNum}
```

```
chol       disease
high       y              1
low        n             46
           y            114
medium     n              9
           y             20
very low   y              3
dtype: int64
```

با توجه به شکل بالا تعداد حالت های مختلف هر ستون با ستون بیماری قابل مشاهده است که میتوان احتمال شرطی را با داشتن این اعداد نوشت.

کد زیر نیز برای ارزیابی مدل با استفاده از داده های تست میباشد که از داده ۱۹۳ به بعد میباشد که با حاصل ضرب احتمال ها به شرط بله/خیر هر کدام که احتمال بیشتری داشته باشد به عنوان لیبل پیش بینی شده در نظر گرفته میشود.(هر احتمال نیز با مقدار ۰/۰۰۱ جمع میشود تا اگر احتمالی برابر صفر باشد تاثیر منفی شدیدی روی احتمال نهایی نگذارد.)

```
#function below is for calculating accuracy
def accuracy_calc(data,start,end):
    accuracy = 0
    for i in range(start,end):
        disease_y = p_diseasePositive*(p_age[(data['age'][i],'y')]+0.001)*(p_sex[(data['sex'][i],'y')]+0
        disease_n = p_diseaseNegative*(p_age[(data['age'][i],'n')]+0.001)*(p_sex[(data['sex'][i],'n')]+0
        predictedLable = 'n'
        if disease_y > disease_n:
            predictedLable = 'y'
        if predictedLable == data['disease'][i]:
            accuracy += 1
    return accuracy/(len(data)-193)
print('accuracy: ',accuracy_calc(knownData,193,len(knownData)))
```

حال مقدار دقت را مشاهده میکنیم که برابر با ۰/۷۱ میباشد.

```
------------------------------------------------------
accuracy:  0.7142857142857143
------------------------------------------------------
```

کد زیر نیز برای پیش بینی لیبل داده ناشناخته میباشد.

```
#algothm below is for predicting new data
def predict_data_lable(data):
    for i in range(len(data)):
        disease_y = p_diseasePositive*(p_age[(data['age'][i],'y')]+0.001)*(p_sex[(data['sex'][i],'y')]+
        disease_n = p_diseaseNegative*(p_age[(data['age'][i],'n')]+0.001)*(p_sex[(data['sex'][i],'n')]+
        if disease_y > disease_n:
            data['disease'][i] = 'y'
        else:
            data['disease'][i] = 'n'
predict_data_lable(unknownData)
print(unknownData)
```

در شکل زیر مشاهده میشود برای تمامی نمونه ها با استفاده از بیض ساده پیش بینی انجام شده است.

```
------------------------------------------------
   age     sex   cp  trestbps    chol  ...   oldpeak   slope   ca   thal disease
0    D    male  cp0  very low     low  ...  very low  slope1  ca0  thal2       y
1    C  female  cp2  very low     low  ...  very low  slope2  ca0  thal0       y
2    C  female  cp0       low     low  ...  very low  slope1  ca0  thal2       y
3    D    male  cp0       low     low  ...    medium  slope1  ca1  thal3       n
4    C  female  cp2       low     low  ...  very low  slope1  ca0  thal2       y
..  ..     ...  ...       ...     ...  ...       ...     ...  ...   ...     ...
56   C    male  cp0       low  medium  ...  very low  slope1  ca1  thal3       n
57   D    male  cp3       low     low  ...    medium  slope1  ca2  thal2       n
58   C    male  cp0  very low     low  ...  very low  slope2  ca0  thal3       n
59   D    male  cp0  very low  medium  ...       low  slope2  ca3  thal3       n
60   C    male  cp0  very low     low  ...    medium  slope1  ca2  thal3       n

[61 rows x 14 columns]
```