

توجه:

*در فایل کد های ارسال شده اگر ران بگیرید چون تغییراتی کوچک در لحظات آخر اعمال شد خطاهایی برای نام متغیر ها می دهد.

*در حل مسائل توضیح کد ها را فقط موارد که ابهام وجود دارد و همچنین مواردی که واضح نیست و اینجانب فکر می کنم توضیح لازم است محدود می کنم.

*همچنین لازم بذکر است که هر بار که کد ها را اجرا می کنیم نتایجی متفاوت می بینیم(به علت فرایند تصادفی که استفاده شده است) بنابراین ممکن است نتایجی که عکسشان اینجا آورده شده با نتایج داخل ژوپیتر متفاوت باشند. ← عمدها برای فرایند جداسازی ترن و تست این اتفاق می افتد چون هر بار مدل جدید بر اساس داده های جدید تنظیم می شود و البته می توان با ۲ =random_state این موضوع کنترل کرد که با این کار چون این فرایند از shuffling استفاده می کند برای نمونه برداری، نمونه ها تصادفی دیگر انتخاب نمی شوند.

۱. مریبوط به Regression Decision Tree

فراخوانی کتابخانه های pandas و numpy

```
1 import numpy as np  
2 import pandas as pd
```

۱.۱. خواندن دیتاست Housing و قرار داده داخل متغیری به نام A

```
1 A = pd.read_csv('Housing.csv')  
2 A.head()
```

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0

۱.۲. مقدار دهی به X (متغیر ها) و Y(تابع هدف)

با توجه به اینکه محدوده تغییرات X و Y بسیار متفاوت است لازم است استانداردسازی شود و از Z score استفاده کردم.

```

1 b = A.values
2 from sklearn.preprocessing import StandardScaler
3 scaler = StandardScaler().fit(b)
4 standardized = scaler.transform(b)
5 A= pd.DataFrame(standardized,columns=["RM", "LSTAT", "PTRATIO", "MEDV"])
6 A

```

	RM	LSTAT	PTRATIO	MEDV
0	0.520554	-1.125077	-1.525083	0.300640
1	0.281048	-0.537070	-0.339748	-0.004498
2	1.469245	-1.259357	-0.339748	1.661047
3	1.178417	-1.413427	0.086973	1.495764
4	1.410146	-1.075605	0.086973	1.851759
...
484	0.548548	-0.462155	1.177482	0.097215
485	-0.187076	-0.545551	1.177482	-0.131639
486	1.144202	-1.031787	1.177482	0.287926
487	0.861150	-0.913055	1.177482	0.046358
488	-0.327047	-0.715168	1.177482	-1.237764

489 rows × 4 columns

```

1 x = A["LSTAT"]
2 y = A["MEDV"]

```

۱.۳. جدا کردن داده های test و train

```

1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

```

۱.۴. مدلسازی با استفاده از روش طبقه بندی Regression Decision Tree

- Max_depth: حداقل عمق درخت. اگر هیچ کدام نباشد، گرهها تا زمانی که همه برگها خالص باشد و یا تا زمانی که تمام برگها کمتر از نمونه های min_samples_split باشند، گسترش می یابند.
- criterion: تابع اندازه گیری کیفیت split. معیارهای پشتیبانی شده "mse" برای میانگین خطای مربع است

```

1 from sklearn.tree import DecisionTreeRegressor
2 regressor = DecisionTreeRegressor(criterion='mse', max_depth=3)
3 regressor.fit(X_train.values.reshape(-1, 1), y_train)

```

DecisionTreeRegressor(max_depth=3)

۱.۵. تست کردن داده های تست و ارزیابی مدل (بررسی میزان دقت مدل)

- برای بررسی دقت مدل می توان بررسی کرد که مقدار خطای چقدر است یا این که با استفاده از regressor.score بررسی کنیم که R² چه مقداری می باشد. ← مقدار خطای مدل کم نمی یاشد اما پوشش دهی نقاط بد نیست با این حال شاید بهتر است الگوریتم های دیگر بررسی شوند.

- اگر هدف یک مقدار پیوسته باشد، برای گره m، نشان دهنده ناحیه rm با مشاهدات nm، معیارهای مشترک برای به کمینه کردن تعیین مکان برای split های بعدی، میانگین مربعات خطای است که خطای L²(نم ۲) را با استفاده از

مقادیر متوسط در گره های پایانه کمینه می کند، و خطای میانگین مطلق ، که خطای L_1 (نرم ۱) را با استفاده از مقادیر میانه در گره های پایانی به حداقل می رساند.

Mean Squared Error:

$$\bar{y}_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - \bar{y}_m)^2$$

Mean Absolute Error:

$$\text{median}(y)_m = \text{median}_{i \in N_m}(y_i)$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - \text{median}(y)_m|$$

• R^2 اندازه گیری آماری نزدیک داده ها به خط رگرسیون برازش شده میباشد. به R^2 ضریب تعیین یا ضریب تشخیص نیز گفته می شود. تعریف ضریب تعیین (R^2) نسبتاً ساده است: ضریب تعیین (R^2) نشان میدهد که چند درصد تغییرات متغیر وابسته به وسیله متغیر مستقل تبیین می شود یا به عبارت دیگر ضریب تعیین نشان دهنده این است که چه مقدار از تغییرات متغیر وابسته تحت تاثیر متغیر مستقل مربوطه بوده و مابقی تغییرات متغیر وابسته مربوط به سایر عوامل میباشد." و از فرمول زیر بدست می آید:

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

که در آن \hat{y}_i مقدار پیش‌بینی شده عنصر i ام و y_i مقدار واقعی و $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ میانگین مقادیر واقعی در آن دسته که i قرار دارد، است.

```

1 from sklearn import metrics
2 regressor.fit(X_train.values.reshape(-1, 1), y_train)
3 y_pred = regressor.predict(X_test.values.reshape(-1, 1))
4 df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
5 print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
6 print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
7 print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
8 print('R^2:',regressor.score(X_test.values.reshape(-1, 1), y_test))

Mean Absolute Error: 0.44693990387923416
Mean Squared Error: 0.35191726528273953
Root Mean Squared Error: 0.5932261502013709
R^2: 0.7033104648336788

```

. ۶. پیدا کردن عمق مناسب برای مدل (Regression Decision Tree)

مدل بدست آمده در این قسمت به `train_test_split` بستگی دارد و در نتیجه عمق مناسب هم متفاوت خواهد بود و بستگی به دارد ولی عمدتا نتایج ما نشان می دهد عمق مناسب ۴ و ۳ می باشد که خطای کمتری داشت و مقدار عددی R^2 بیشتری داشت.

```

1 import matplotlib.pyplot as plt
2 Absolute=[]
3 MSE=[]
4 RMSE=[]
5 for i in range(1,10):
6     regressor = DecisionTreeRegressor(criterion='mse', max_depth=i)
7     regressor.fit(X_train.values.reshape(-1, 1), y_train)
8     y_pred = regressor.predict(X_test.values.reshape(-1, 1))
9     df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
10    print(i)
11    a=metrics.mean_absolute_error(y_test, y_pred)
12    print('Mean Absolute Error:',a)
13    b=metrics.mean_squared_error(y_test, y_pred)
14    print('Mean Squared Error:',b)
15    c=np.sqrt(metrics.mean_squared_error(y_test, y_pred))
16    print('Root Mean Squared Error:',c)
17    print('R^2:',regressor.score(X_test.values.reshape(-1, 1), y_test))
18    print("\n")
19    Absolute.append(a)
20    MSE.append(b)
21    RMSE.append(c)
22 plt.figure(figsize=(12, 6))
23 plt.plot(range(1, 10), Absolute, color='red', linestyle='dashed', marker='o',markerfacecolor='blue', markersize=10)
24 plt.title('Mean Absolute Error')
25 plt.xlabel('max_depth')
26 plt.ylabel('Mean Absolute Error')

1
Mean Absolute Error: 0.6375922433509191
Mean Squared Error: 0.6972076110758083
Root Mean Squared Error: 0.8349895874056205
R^2: 0.4122078611905817

2
Mean Absolute Error: 0.4882239098501209
Mean Squared Error: 0.42957000731793443
Root Mean Squared Error: 0.6554159040776585
R^2: 0.6378440663030402

3
Mean Absolute Error: 0.44693990387923416
Mean Squared Error: 0.35191726528273953
Root Mean Squared Error: 0.5932261502013709
R^2: 0.7033104648336788

4
Mean Absolute Error: 0.46166242791207585
Mean Squared Error: 0.3677634090416033
Root Mean Squared Error: 0.6064349998487911
R^2: 0.6899511173682488

```

```
5  
Mean Absolute Error: 0.49555404402055414  
Mean Squared Error: 0.48432049639864855  
Root Mean Squared Error: 0.6959313877090533  
R^2: 0.5916857820755386
```

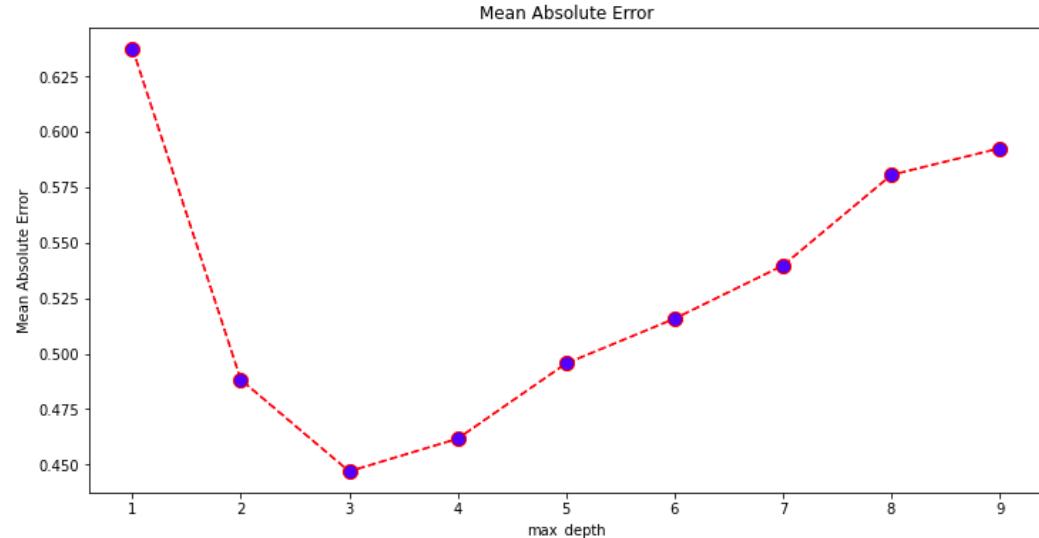
```
6  
Mean Absolute Error: 0.5157497820538934  
Mean Squared Error: 0.5372128723362658  
Root Mean Squared Error: 0.7329480693311538  
R^2: 0.5470940101482189
```

```
7  
Mean Absolute Error: 0.5396942018543597  
Mean Squared Error: 0.588535344613569  
Root Mean Squared Error: 0.7671605729008556  
R^2: 0.5038257708610504
```

```
8  
Mean Absolute Error: 0.5805284021831405  
Mean Squared Error: 0.6389432142056977  
Root Mean Squared Error: 0.7993392359978945  
R^2: 0.4613286021415858
```

```
9  
Mean Absolute Error: 0.5925573439352768  
Mean Squared Error: 0.6732791123105476  
Root Mean Squared Error: 0.8205358689969303  
R^2: 0.4323811686018828
```

```
Text(0, 0.5, 'Mean Absolute Error')
```

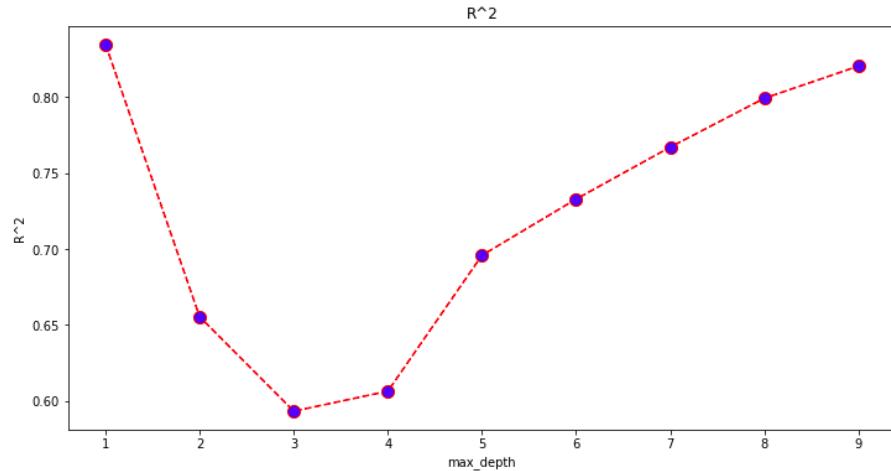


```

1 r2=[]
2 for i in range(1,10):
3     regressor = DecisionTreeRegressor(criterion='mse', max_depth=i)
4     regressor.fit(X_train.values.reshape(-1, 1), y_train)
5     y_pred = regressor.predict(X_test.values.reshape(-1, 1))
6     df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
7     c=regressor.score(X_test.values.reshape(-1, 1), y_test)
8     r2.append(c)
9 plt.figure(figsize=(12, 6))
10 plt.plot(range(1, 10), RMSE, color='red', linestyle='dashed', marker='o',markerfacecolor='blue', markersize=10)
11 plt.title('R^2')
12 plt.xlabel('max_depth')
13 plt.ylabel('R^2')

```

Text(0, 0.5, 'R^2')



نتایج بعد از جداسازی و مدل سازی دوباره متفاوت است.

بنظر می آید در سوال نام دیتاست اشتباه ذکر شده است. از دیتاست Pima-Diabetes.csv استفاده کردیم. ۲.

۲،۱ خواندن دیتاست و بررسی چند متغیر اول

```

1 import numpy as np
2 import pandas as pd
3 import pandas as pd
4 B = pd.read_csv('Pima-Diabetes.csv',names=["Number of times pregnant","Plasma glucose concentration a 2 hours i
5 B.head()

```

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
1 B.describe(include = 'all')
```

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```

1 import numpy as np
2 import pandas as pd
3 import pandas as pd
4 B = pd.read_csv('Pima-Diabetes.csv',names=["Number of times pregnant","Plasma glucose concentration a 2 hours i
5 B.head()

```

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
1 B.describe(include = 'all')
```

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
1 B.isnull().sum().sum()
```

0

```
1 B.dtypes
```

```
Number of times pregnant          int64
Plasma glucose concentration a 2 hours in an oral glucose tolerance test    int64
Diastolic blood pressure (mm Hg)      int64
Triceps skin fold thickness (mm)      int64
2-Hour serum insulin (mu U/ml)       int64
Body mass index (weight in kg/(height in m)^2) float64
Diabetes pedigree function          float64
Age (years)                         int64
Class variable (0 or 1)              int64
dtype: object
```

```
1 B.unique()
```

```
Number of times pregnant          17
Plasma glucose concentration a 2 hours in an oral glucose tolerance test    136
Diastolic blood pressure (mm Hg)      47
Triceps skin fold thickness (mm)      51
2-Hour serum insulin (mu U/ml)       186
Body mass index (weight in kg/(height in m)^2) 248
Diabetes pedigree function          517
Age (years)                         52
Class variable (0 or 1)              2
dtype: int64
```

```
1 B.groupby('Class variable (0 or 1)').size()
```

```
Class variable (0 or 1)
0    500
1    268
dtype: int64
```

در این قسمت علاوه بر head از چند متود دیگر هم برای شناخت بهتر داده ها استفاده شد. اولاً داده نال ندارد(اما ما متوجه مقادیر صفر در برخی ویژگی ها می شویم که معنایی ندارند؛ مثل انسولین، ضخامت و فشار خون و...) که با توجه به این موضوع لازم است که این داده ها تا حدی تمیز شوند و مقادیر صفر را جایگذاری کنیم.

```

1 d = B
2 d['Plasma glucose concentration a 2 hours in an oral glucose tolerance test'] = B['Plasma glucose concentration
3 d['Diastolic blood pressure (mm Hg)'] = B['Diastolic blood pressure (mm Hg)'].replace(0,B['Diastolic blood pres
4 d['Triceps skin fold thickness (mm)'] = B['Triceps skin fold thickness (mm)'].replace(0,B['Triceps skin fold th
5 d['2-Hour serum insulin (mu U/ml)'] = B['2-Hour serum insulin (mu U/ml)'].replace(0,B['2-Hour serum insulin (mu
6 d['Body mass index (weight in kg/(height in m)^2)'] = B['Body mass index (weight in kg/(height in m)^2)'].repla
7 d['Class variable (0 or 1)'] = B['Class variable (0 or 1)']
8 d.head()

```

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
0	6	148	72	35	30.5	33.6	0.627	50	1
1	1	85	66	29	30.5	26.6	0.351	31	0
2	8	183	64	23	30.5	23.3	0.672	32	1
3	1	89	66	23	94.0	28.1	0.167	21	0
4	0	137	40	35	168.0	43.1	2.288	33	1

```
1 d.describe(include = 'all')
```

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.656250	72.386719	27.334635	94.652344	32.450911	0.471876	33.240885	0.348958
std	3.369578	30.438286	12.096642	9.229014	105.547598	6.875366	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	23.000000	30.500000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	31.250000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

*ما در بالا یک رویکرد سطحی را استفاده کردیم چرا که پیدا کردن میانگین و جایگذاری آن باعث می شود که ما تفاوتی میان افرادی که دیابت دارند و ندارند قابل نشویم. بهتر است با استفاده از از این که مقدار انسولین صفر مورد نظر مربوط به چه افرادی است (دیابت دارند یا ندارند) مقدار میانگین یا میانه مناسب را جایگذاری کنیم. (چرا که اگر میانگین کل برای انسولین مثل ۷۰

باشد در صورتی که ما برای کسی که دیابت دارد مقدار انسولین را ۷۰ بگذاریم ممکن است باعث ایجاد bias شویم)

*نوع داده ها مشخص شده و یک خلاصه ای از داده ها نمایش داده شده است. و همچنین مقادیر متغیر هدف را بررسی کردیم.

.۲.۲ در قسمت قبل هم زمان با خواندن دیتابست برای ستون ها نام گذاری انجام شد.

.۲.۳ نمودار همبستگی متغیر ها را در این جا مشاهده می کنیم از مواردی که توجه ما را جلب می کند همبستگی

گلوکز و BMI و پس از آن سن و ... با داشتن بیماری دیابت می باشد. و همینطور ما همبستگی

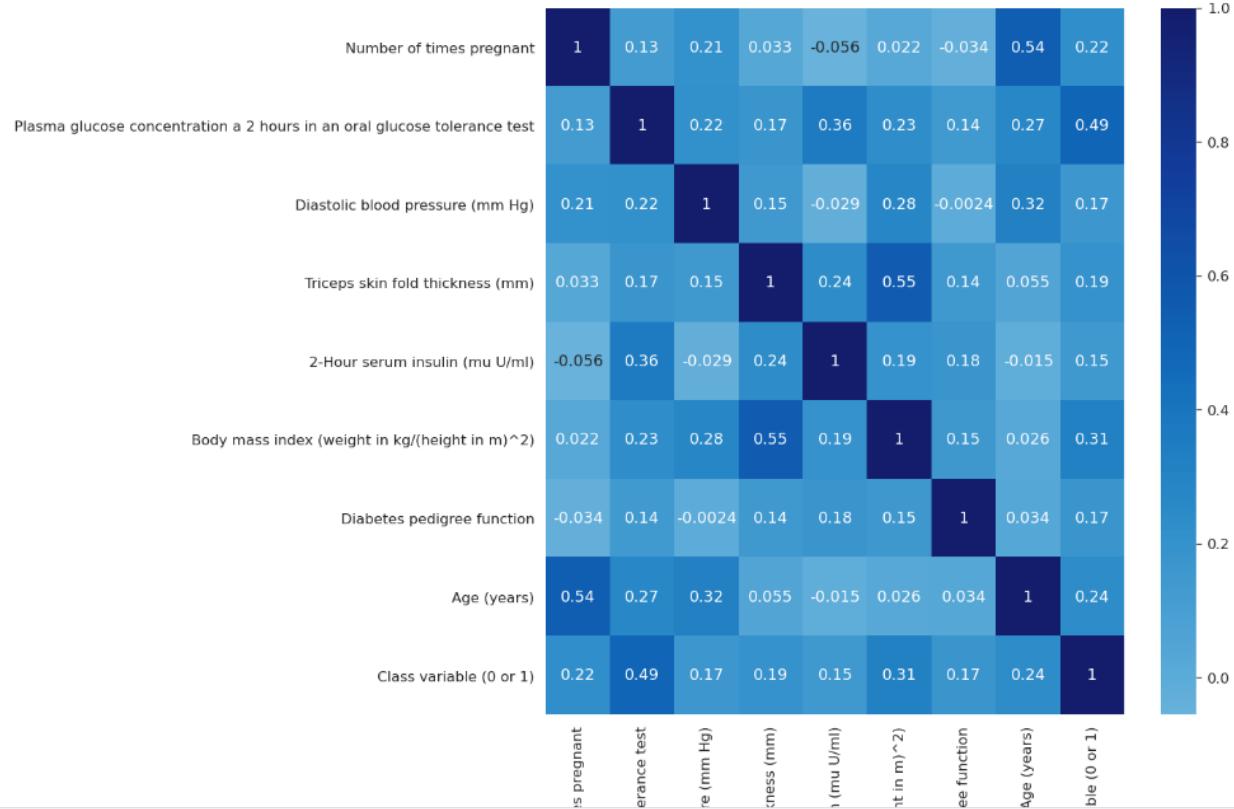
هایی بین برخی دیگر از متغیر ها با یکدیگر مشاهده می کنیم.

```

1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 fig, ax = plt.subplots(figsize=(15,15))
4 sns.set(font_scale=1.5)
5 #sns.heatmap(d.corr(), center=0, cmap='Blues', annot=True)
6 sns.heatmap(B.corr(), center=0, cmap='Blues', annot=True)

<matplotlib.axes._subplots.AxesSubplot at 0x7f6011d55090>

```



```

1 X = B.drop("Class variable (0 or 1)", axis=1)
2 y = B["Class variable (0 or 1)"]

```

۵. جدا کردن داده های train و test

```

1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

```

۶. مدلسازی با استفاده از روش طبقه بندی Decision Tree Classifier

```

1 from sklearn.tree import DecisionTreeClassifier
2 from sklearn.tree import tree
3 classifier = DecisionTreeClassifier(criterion='entropy', max_features=4, max_depth=5)
4 classifier.fit(X_train, y_train)
5 #tree.plot_tree(classifier)

```

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, max_features=4)
```

۷. تست کردن داده های تست و ارزیابی مدل (بررسی میزان دقت مدل)

دقت مدل برای تربین و تست در شکل زیر می بینیم که هر دو بد نیستند. (زیاد هم خوب نیستند)

```

1 y_pred = classifier.predict(X_test)

1 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
2 print(confusion_matrix(y_test, y_pred))
3 print(classification_report(y_test, y_pred))
4 print('Testing-set accuracy score: {:.4f}'.format(accuracy_score(y_test, y_pred)))
5 print('Training set score: {:.4f}'.format(classifier.score(X_train, y_train)))
6 print('Test set score: {:.4f}'.format(classifier.score(X_test, y_test)))

[[76 25]
 [14 39]]
      precision    recall   f1-score   support
          0       0.84      0.75      0.80      101
          1       0.61      0.74      0.67       53

   accuracy         0.7468
macro avg       0.73      0.74      0.73      154
weighted avg    0.76      0.75      0.75      154

Testing-set accuracy score: 0.7468
Training set score: 0.8241
Test set score: 0.7468

```

پیدا کردن عمق مناسب برای مدل .۲،۸

```

1 a=[]
2 b=[]
3 for i in range(3,10):
4     print(i)
5     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
6     classifier = DecisionTreeClassifier(criterion='entropy', max_depth=i)
7     classifier.fit(X_train, y_train)
8     y_pred = classifier.predict(X_test)
9     #print(classification_report(y_test, y_pred))
10    a.append(classifier.score(X_train, y_train))
11    b.append(classifier.score(X_test, y_test))
12    print('Training set score: {:.4f}'.format(classifier.score(X_train, y_train)))
13    print('Test set score: {:.4f}'.format(classifier.score(X_test, y_test)))
14 plt.figure(figsize=(12, 6))
15 plt.plot(range(3, 10), a, linestyle='dashed', marker='o', markerfacecolor='blue', markersize=10, label="train")
16 plt.plot(range(3, 10), b, linestyle='dashed', marker='o', markerfacecolor='green', markersize=10, label="test")
17 plt.title('Accuracy')
18 plt.xlabel('max_depth')
19 plt.ylabel('Accuracy')
20 plt.legend()

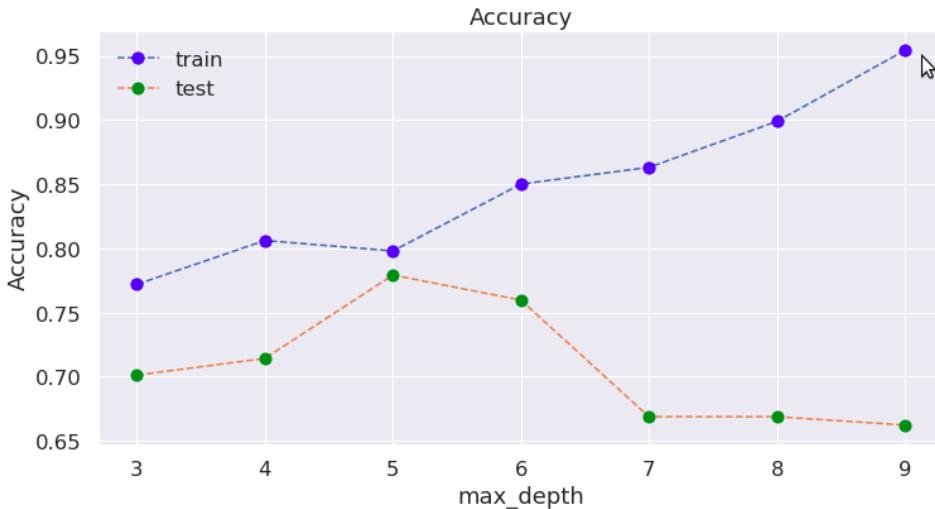
3
Training set score: 0.7720
Test set score: 0.7013
4
Training set score: 0.8062
Test set score: 0.7143
5
Training set score: 0.7980
Test set score: 0.7792
6
Training set score: 0.8502
Test set score: 0.7597
7
Training set score: 0.8632
Test set score: 0.6688
8
Training set score: 0.8990
Test set score: 0.6688
9
Training set score: 0.9544
Test set score: 0.6623

<matplotlib.legend.Legend at 0x7f600e449550>

```

Accuracy

در نمودار زیر به خوبی مشاهده میشود که بهترین عمق ۵ برای مدل می باشد چرا که دقت مدل برای داده های تست در بهترین وضعیت قرار داد. از طرفی با افزایش عمق درخت ما شاهد افزایش دقت مدل برای داده های ترین هستیم و کاهش دقت برای داده های تست که نشان می دهد ما overfitting داریم.



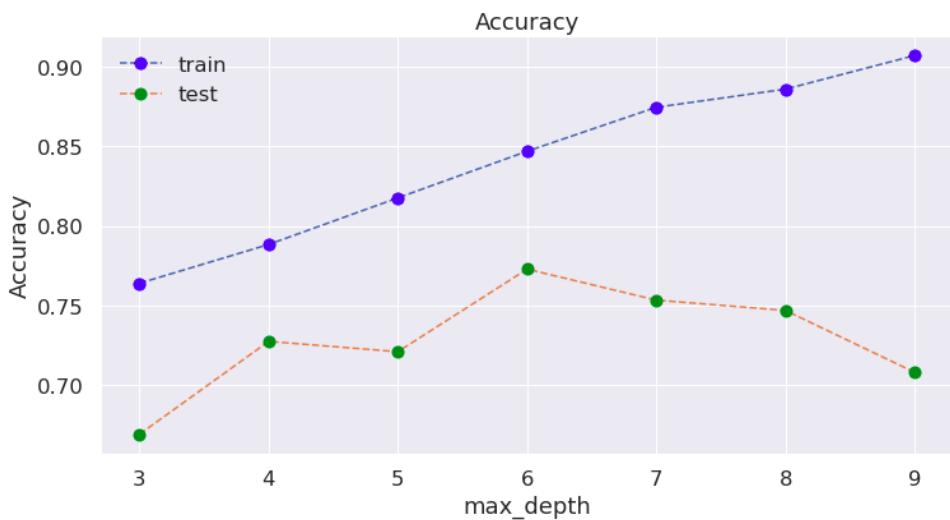
در قسمت بالا ما هر بار که می خواستیم مدل کنیم یک بار هم داده های ترین و تست را جداسازی می کردیم چون در خط ۵ کد ما (train_test_split) را استفاده کردیم که سبب می شود برای هر مدلسازی شانس (احتمال تصادفی) را هم دخیل کنیم و داشتن نتایج متفاوت معقول است اما در ادامه ما تصمیم گرفتیم برای این که تنها اثر عمق را بررسی کنیم این خط را از کد حذف کنیم و از جدا سازی قسمت ۲-۵ که برای مدل اصلی بود استفاده کنیم.

```

1 a=[]
2 b=[]
3 for i in range(3,10):
4     print(i)
5     classifier = DecisionTreeClassifier(criterion='entropy',max_depth=i)
6     classifier.fit(X_train, y_train)
7     y_pred = classifier.predict(X_test)
8     #print(classification_report(y_test, y_pred))
9     a.append(classifier.score(X_train, y_train))
10    b.append(classifier.score(X_test, y_test))
11    print('Training set score: {:.4f}'.format(classifier.score(X_train, y_train)))
12    print('Test set score: {:.4f}'.format(classifier.score(X_test, y_test)))
13 plt.figure(figsize=(12, 6))
14 plt.plot(range(3, 10), a, linestyle='dashed', marker='o',markerfacecolor='blue', markersize=10,label="train")
15 plt.plot(range(3, 10), b, linestyle='dashed', marker='o',markerfacecolor='green', markersize=10,label="test")
16 plt.xlabel('max_depth')
17 plt.ylabel('Accuracy')
18 plt.legend()
19 plt.show()

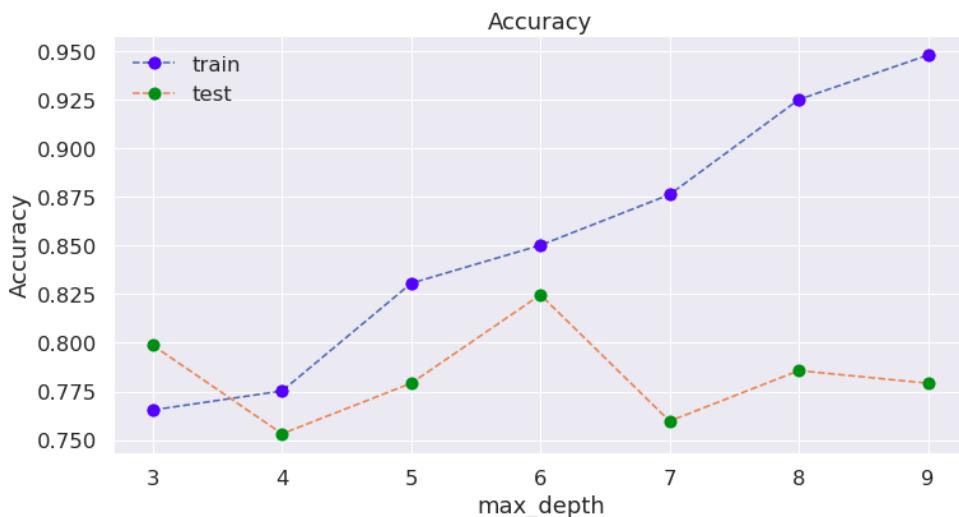
3
Training set score: 0.7638
Test set score: 0.6688
4
Training set score: 0.7883
Test set score: 0.7273
5
Training set score: 0.8176
Test set score: 0.7208
6
Training set score: 0.8469
Test set score: 0.7727
7
Training set score: 0.8746
Test set score: 0.7532
8
Training set score: 0.8860
Test set score: 0.7468
9
Training set score: 0.9072
Test set score: 0.7078
<matplotlib.legend.Legend at 0x7f0536c41b50>

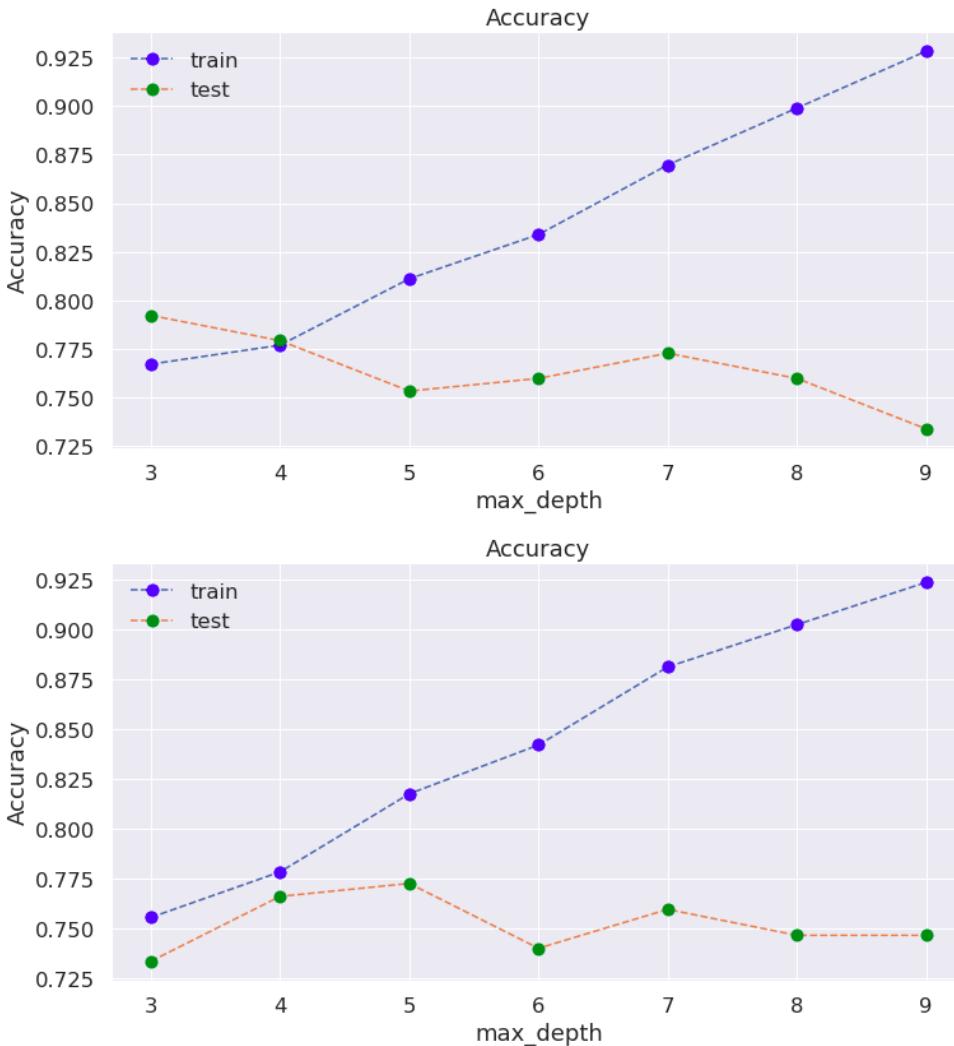
```



در دیاگرام بالا مشاهده می کنیم که نتایج مربوط به عمق ۶ بهتر می باشد و بعد از آن ما شاهد overfitting هستیم. نکته قابل توجه این است که train_test_split بر روی مدلسازی اثر انکار ناپذیری می گذارد و اگر یک بار دیگر جداسازی را انجام دهیم و نتایج را بررسی کنیم ممکن است شگفت زده شویم چرا که ممکن است جواب های بدست آمده به جواب فعلی حتی نزدیک هم نباشند. بنابراین توصیه ما این است که در صورت انجام train_test_split باری دیگر دقت بررسی شود و عمق بهتر بدست بیاید.

در ادامه نتایج را برای train_test_split جدید می بینیم:





مشاهده می شود که در نمودار های بالا عمق مناسب متفاوت است اما مشاهده شد که عمدتاً عمق مناسب ۵ یا ۶ می باشد.

۲.۹ در نمودار مربوط متده `feature_importances` را مشاهده می کنیم که نشان دهنده اهمیت و تاثیر تک پارامتر ها بر روی مدل می باشد.

عوامل مهم در مدل با همبستگی پارامتر ها با تابع هدف اولویت متفاوتی دارند. برای مثال دیابت حاملگی یکی از مواردی است که در مدل تاثیر خوبی داشته با وجود این که همبستگی با تابع هدف در آن مقداری کمتر از تاثیر سن می باشد!

نام متغیر	همبستگی	feature_importances
glucose	۰.۴۹	۰.۲۶۲۰۵۲۱۱
BMI	۰.۳۱	۰.۲۱۵۶۵۶۲۶
Diabetes pedigree function	۰.۱۷	۰.۱۴۹۴۲۰۵۸

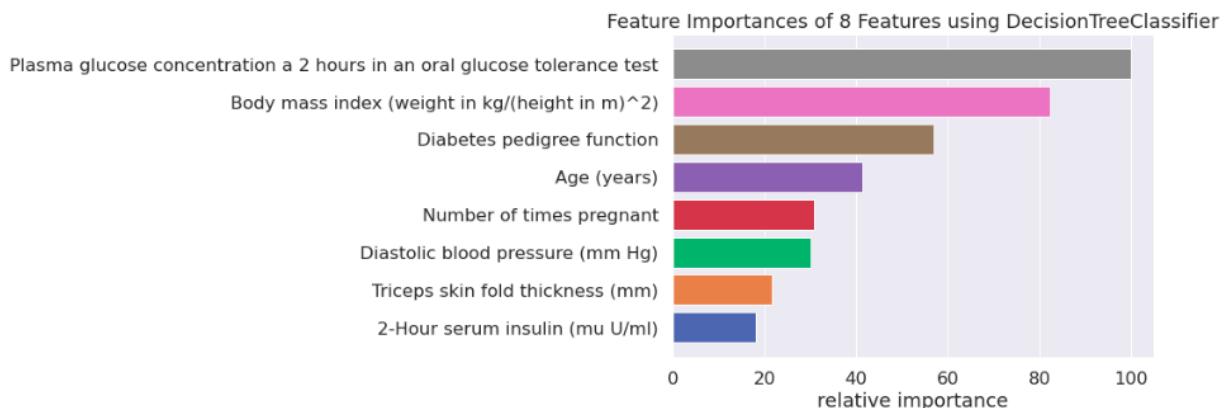
.10841826	.24	Age
.08105699	.22	No. pregnant
.07910373	.17	blood pressure
.05665945	.19	Triceps skin
.04763261	.15	2-Hour serum insulin

```

1 from yellowbrick.model_selection import FeatureImportances
2 viz = FeatureImportances(classifier)
3 viz.fit(X_train, y_train)
4 viz.show()
5 classifier.feature_importances_

```

/home/nick/anaconda3/lib/python3.7/site-packages/sklearn/base.py:213: FutureWarning: From version 0.24, get_params will raise an AttributeError if a parameter cannot be retrieved as an instance attribute. Previously it would return None.
FutureWarning)
/home/nick/anaconda3/lib/python3.7/site-packages/yellowbrick/model_selection/importances.py:291: UserWarning: Tight layout not applied. The left and right margins cannot be made large enough to accommodate all axes decorations.
plt.tight_layout()



```
array([0.08105699, 0.26205211, 0.07910373, 0.05665945, 0.04763261,
       0.21565626, 0.14942058, 0.10841826])
```

ما برای بهبود مدل انسولین را در مدل حذف می کنیم تا ببینیم خطا تست کاهش می یابد . مشاهده می شود که نتایج برای داده های تست بهتر عمل کرد ولی توجه شود که این جا ما باز چون ترن و تست را انجام می دهیم ممکن است نتایج متفاوتی را شاهد باشیم.

```

1 X = B[['Plasma glucose concentration a 2 hours in an oral glucose tolerance test','Body mass index (weight in kg/m^2)', 'Age (years)', 'Skin fold thickness (mm)', 'Serum insulin (mu U/ml)', 'Diabetes pedigree function', 'Triceps skin fold thickness (mm)', '2-Hour serum glucose (mmol/L)']]
2 y = B['Class variable (0 or 1)']
3 from sklearn.model_selection import train_test_split
4 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
5 from sklearn import tree
6 classifier = DecisionTreeClassifier(criterion='entropy',max_features=3,max_depth=5)
7 classifier.fit(X_train, y_train)
8 y_pred = classifier.predict(X_test)
9 from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
10 print(confusion_matrix(y_test, y_pred))
11 print(classification_report(y_test, y_pred))
12 print('Testing-set accuracy score: {:.4f}'.format(accuracy_score(y_test, y_pred)))
13 print('Training set score: {:.4f}'.format(classifier.score(X_train, y_train)))
14 print('Test set score: {:.4f}'.format(classifier.score(X_test, y_test)))

```

```

[[92 14]
[21 27]]
      precision    recall   f1-score   support
0       0.81     0.87     0.84     106
1       0.66     0.56     0.61      48

   accuracy         0.77
  macro avg     0.74     0.72     0.72     154
weighted avg     0.77     0.77     0.77     154

```

Testing-set accuracy score: 0.7727

Training set score: 0.8094

Test set score: 0.7727

در زیر نتایج حاصل از یک ترین و تست برای مدل اصلی و مدل فعلی آورده ایم تا مقایسه درستی باشد:

در مدل اول همه متغیرها آمده و در مدل دوم انسولین حذف شده در اینجا مدل بهبود پیدا کرده است (با این حال خطا و دقت

مدل بطور میانگین تغییر نکرد)

```

1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.tree import tree
4 classifier = DecisionTreeClassifier(criterion='entropy',max_features=4,max_depth=5)
5 classifier.fit(X_train, y_train)
6 y_pred = classifier.predict(X_test)
7 from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
8 print(confusion_matrix(y_test, y_pred))
9 print(classification_report(y_test, y_pred))
10 print('Testing-set accuracy score: {:.0:0.4f}'.format(accuracy_score(y_test, y_pred)))
11 print('Training set score: {:.4f}'.format(classifier.score(X_train, y_train)))
12 print('Test set score: {:.4f}'.format(classifier.score(X_test, y_test)))

[[81 17]
 [22 34]]
      precision    recall   f1-score   support
          0       0.79      0.83      0.81      98
          1       0.67      0.61      0.64      56

   accuracy                           0.75      154
  macro avg       0.73      0.72      0.72      154
weighted avg       0.74      0.75      0.74      154

Testing-set accuracy score: 0.7468
Training set score: 0.8078
Test set score: 0.7468

[[83 15]
 [21 35]]
      precision    recall   f1-score   support
          0       0.80      0.85      0.82      98
          1       0.70      0.62      0.66      56

   accuracy                           0.77      154
  macro avg       0.75      0.74      0.74      154
weighted avg       0.76      0.77      0.76      154

Testing-set accuracy score: 0.7662
Training set score: 0.8078
Test set score: 0.7662

```

۲،۱۰. در این سوال ما مدلی که در پایان سوال قبل ترین کردیم را استفاده می کنیم:

فایل جواب به صورت پی دی اف به همراه کد این سوال ارسال شده است. نام پی دی اف classifierFinal می باشد.

```

1 from sklearn import tree
2 #classifier = DecisionTreeClassifier(criterion='entropy',max_features=3,max_depth=5)
3 #classifier.fit(X_train, y_train)
4 print(tree.export_graphviz(classifier))

graph Tree {
node [shape=box];
0 [label="X[1] <= 27.35\nentropy = 0.93\nsamples = 614\nvalue = [402, 212]"] ;
1 [label="X[1] <= 22.8\nentropy = 0.465\nsamples = 142\nvalue = [128, 14]"] ;
0 -> 1 [label=distance=2.5, labelangle=45, headlabel="True"];
2 [label="entropy = 0.0\nsamples = 34\nvalue = [34, 0]"] ;
1 -> 2 ;
3 [label="X[0] <= 147.5\nentropy = 0.556\nsamples = 108\nvalue = [94, 14]"] ;
1 -> 3 ;
4 [label="X[0] <= 106.5\nentropy = 0.297\nsamples = 95\nvalue = [90, 5]"] ;
3 -> 4 ;
5 [label="entropy = 0.0\nsamples = 44\nvalue = [44, 0]"] ;
4 -> 5 ;
6 [label="X[1] <= 22.95\nentropy = 0.463\nsamples = 51\nvalue = [46, 5]"] ;
4 -> 6 ;
7 [label="entropy = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
6 -> 7 ;
8 [label="entropy = 0.402\nsamples = 50\nvalue = [46, 4]"] ;
6 -> 8 ;
9 [label="X[3] <= 25.5\nentropy = 0.89\nsamples = 13\nvalue = [4, 9]"] ;
7 -> 9 ;
}

1 import os
2 os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin/'

1 import graphviz
2 dot_data = tree.export_graphviz(classifier, out_file=None)
3 graph = graphviz.Source(dot_data)
4 graph.render("classifierFinal")

'classifierFinal.pdf'

```

.۲،۱۱ گراف مربوط به مدل در قسمت ۹-۲ را اینجا مشاهده می کنیم:

```

1 from IPython.display import Image
2 dot_data = tree.export_graphviz(classifier, out_file=None,
3                                 feature_names=["Plasma glucose concentration a 2 hours in an oral glucose tolerance test",
4                                 "Class variable (0 or 1)",
5                                 filled=True, rounded=True,
6                                 special_characters=True)
7 graph = graphviz.Source(dot_data)
8 graph.render("classifierFinal")

'classifierFinal.pdf'

1 import pydotplus
2 graph = pydotplus.graph_from_dot_data(dot_data)
3 Image(graph.create_png())

```

جواب این سوال علاوه بر این که در خود ژوپیتر قابل نمایش است در classifierFinal.png نیز ذخیره شده و همراه کد ارسال می شود.

۳. خوشبندی با K-means

```

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns;
5 from sklearn.cluster import KMeans

```

لود دیتاست .۳.۱

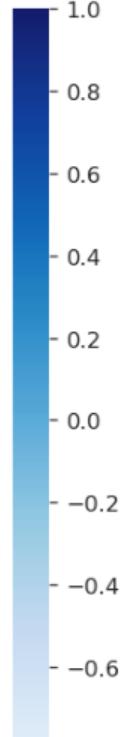
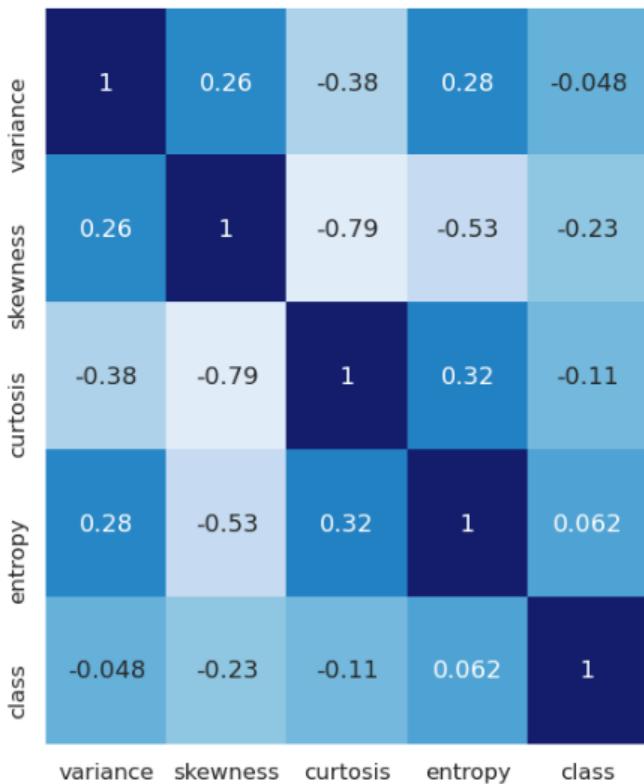
```

1 df = pd.read_csv("Banknote.csv", header=None, names=["variance", "skewness", "curtosis", "entropy", "class"])
2 df.head()

```

	variance	skewness	curtosis	entropy	class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

ساخت مدل .۳.۲



از دیاگرام heatmap متوجه می شویم که همبستگی قابل توجه بین هیچ کدام از داده ها قابل مشاهده نیست. از نتایج تجزیه و تحلیل که دو ویژگی واریانس و کجی در مجموعه داده بیشترین واریانس را دارند، این نشان می دهد که اطلاعات بالقوه برای آموختن در این دو پارامتر وجود دارد.

```

1 from sklearn.cluster import KMeans
2 #df1=df.drop(["class"],axis=1)
3 KM=KMeans(n_clusters=2).fit(df)
4 y_kmeans = KM.predict(df)

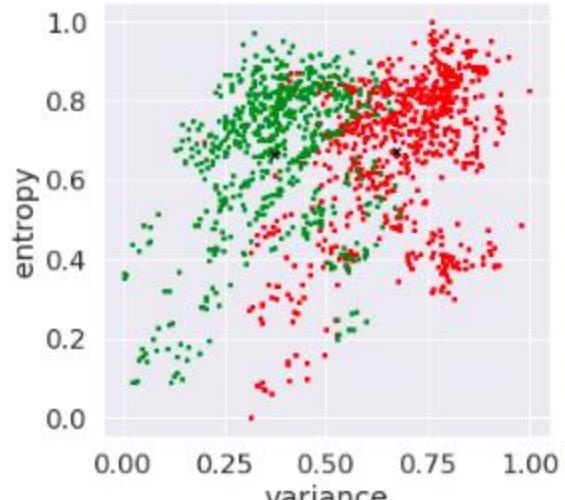
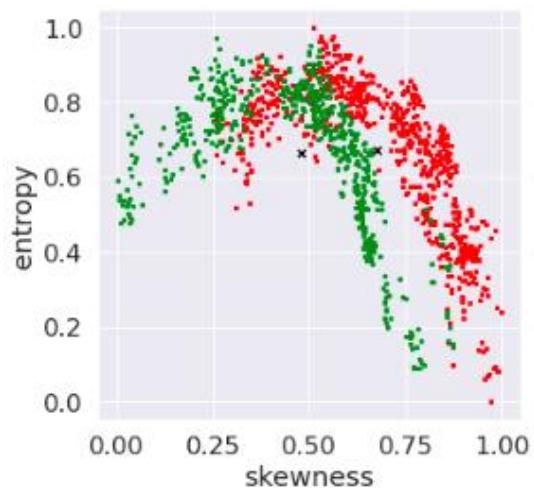
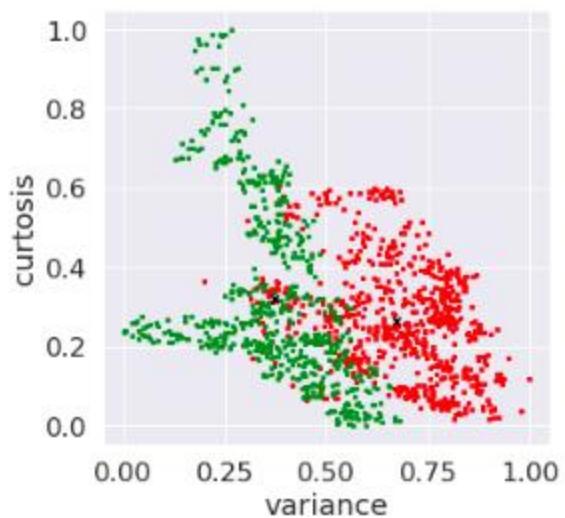
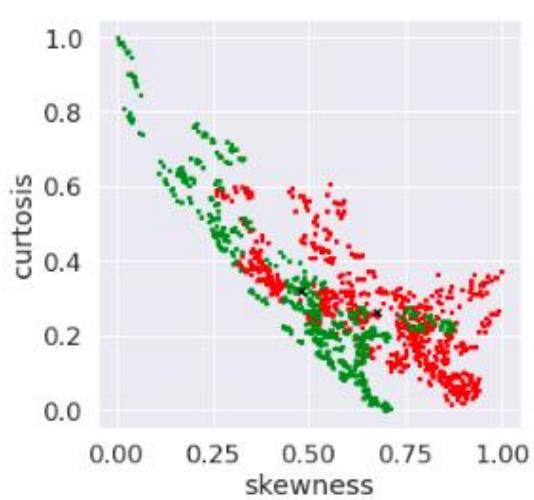
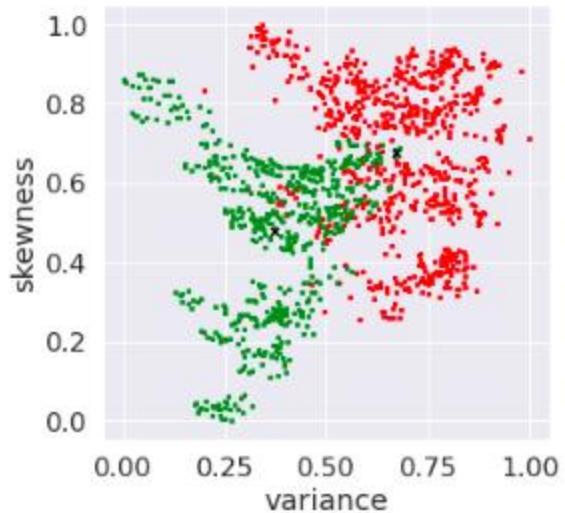
```

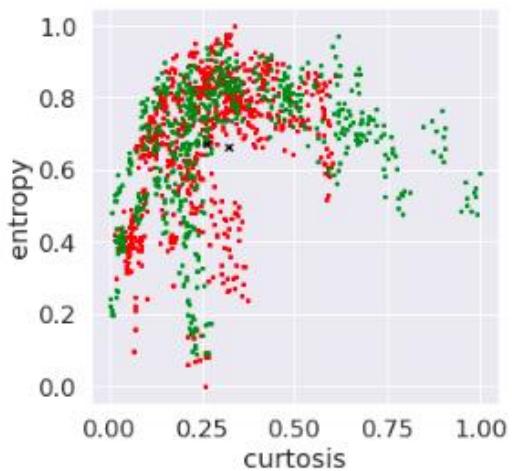
.۳،۳ مراکز خوشه ها

```
1 centroids = KM.cluster_centers_
2 centroids
array([[6.72016529e-01, 6.74646570e-01, 2.62037951e-01, 6.72918858e-01,
       6.66133815e-16],
       [3.73094018e-01, 4.78191481e-01, 3.20260668e-01, 6.63916933e-01,
       1.00000000e+00]])
```

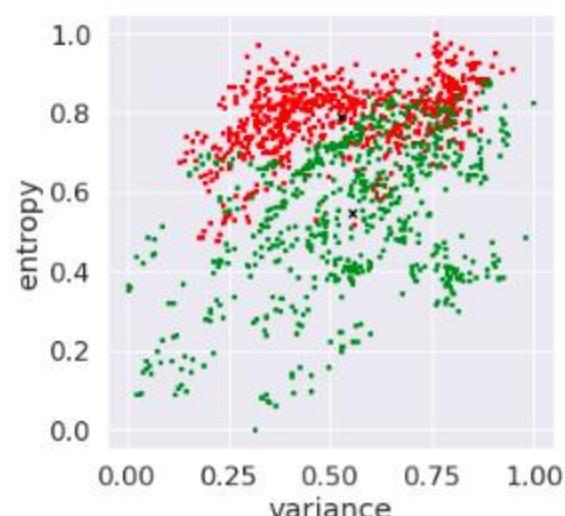
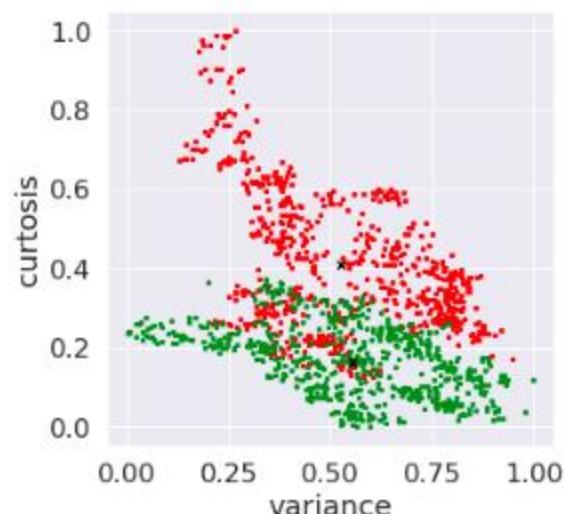
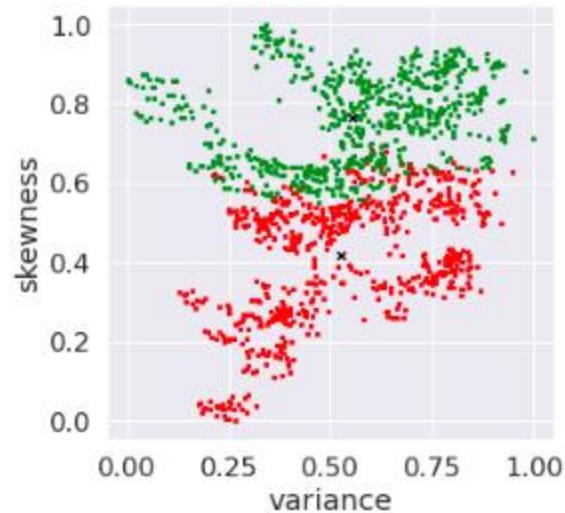
.۴،۳ نمودار مربوط به خوشه ها

```
1 l=["variance","skewness","curtosis","entropy"]
2 for a in range(4):
3     for b in range(4):
4         if a !=b :
5             x=df[l[a]]
6             y=df[l[b]]
7             x0=[]
8             y0=[]
9             x1=[]
10            y1=[]
11
12            for i in range(0,x.size):
13                if(KM.labels_[i] == 0):
14                    x0.append(x[i])
15                    y0.append(y[i])
16                else:
17                    x1.append(x[i])
18                    y1.append(y[i])
19
20
21 #Plotting the values and cluster centres
22 plt.figure(figsize=(5,5))
23 plt.scatter(x0,y0,c="red",s=5)
24 plt.scatter(x1,y1,c="green",s=5)
25 plt.scatter(centroids[:,a],centroids[:,b],c="black",s=25,marker="x")
26 plt.xlabel(l[a])
27 plt.ylabel(l[b])
```





اگر متغیر کلاس را حذف کنیم نتایج بهتری می بینم در واقع خوشه ها تمیز تر خواهند بود:



.۳،۵ اینرسی مدل

within-cluster sum-of-squares criterion یا اینرسی برای مدل بالا

هر چه کمتر باشد نشان می دهد که عناصر داخل یک کلاستر به یکدیگر شبیه ترند.

```
1 i=KM.inertia_
2 i
3
4 176.51602058157988
```

۳.۶. اینرسی برای خوش بندی با تعداد خوش های متغیر

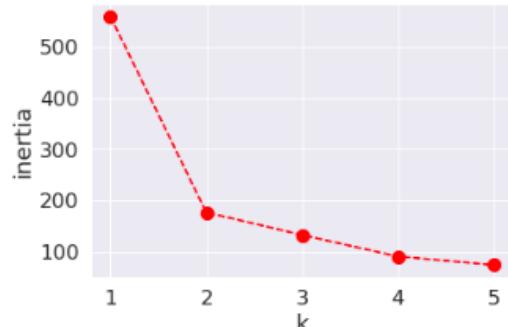
در این قسمت مدلی را که برای همه ویژگی ها بررسی شده در نظر گرفتیم.

```
1 inertias=[]
2 for i in range(1,6):
3     k=KMeans(n_clusters=i).fit(df)
4     inertias.append(k.inertia_)
5 print(k.inertia_,"\\t",i)
```

559.8299653599079	1
176.51602058157988	2
132.26375303529656	3
90.34854920855065	4
73.48336997849296	5

۳.۷. نمودار قسمت قبلی سوال و تفسیر آن

```
1 plt.plot(range(1, 6), inertias, color='red', linestyle='dashed', marker='o', markerfacecolor='red', markersize=10
2 plt.xlabel("K")
3 plt.ylabel("inertia")
4 plt.show()
```



$$WCSS = \sum_{i=1}^N \sum_{j=1}^k W^{(i,j)} \|x^{(i)} - \mu^{(j)}\|^2 \quad (1)$$

از نمودار نشان داده شده در بالا ، می توان متوجه شد که در خطای مجموع مربعات (1) () inertia یا within-cluster sum-of-squares criterion WCSS سقوط دارد، در شکل ، سقوط شدید در مقدار خطا ، هنگامی که تعداد خوش ها بین ۱ تا ۲ باشد ، نشان می دهد که تقریباً دو خوش بندی دارد. این مطابق با این واقعیت است که در مجموعه داده دو کلاس وجود دارد.

*** از نتایج تجزیه و تحلیل که دو ویژگی واریانس و کجی در مجموعه داده بیشترین واریانس را دارند، این نشان می دهد که اطلاعات بالقوه برای آموختن در این دو پارامتر وجود دارد.

```
1 from sklearn.decomposition import PCA
2 pca = PCA(n_components=4)
3 X_r = pca.fit(df)
4 print(X_r.explained_variance_ratio_)
```

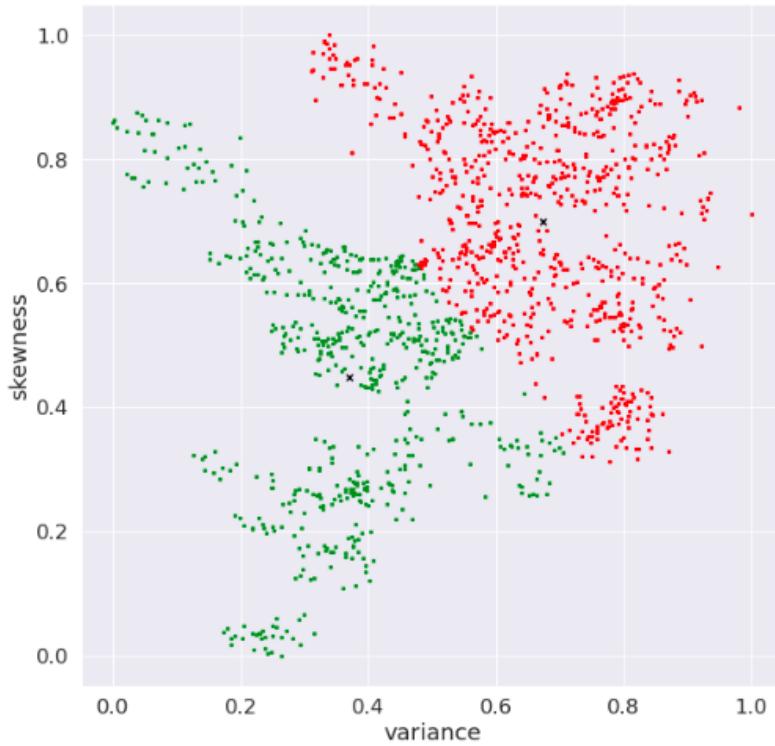
[0.69555915 0.18953598 0.08398447 0.02646894]

مدل آموزش داده شده را فقط برای دو ویژگی که در قسمت دو مدل کردیم در نظر بگیریم باز صحبت بالا درست است

```
1 from sklearn.cluster import KMeans
2
3 dfl=df.drop(["class","curtosis","entropy"],axis=1)
4 KM=KMeans(n_clusters=2).fit(dfl)
5 y_kmeans = KM.predict(dfl)

1 centroids = KM.cluster_centers_
2 centroids
array([[0.67423251, 0.6984855 ],
       [0.37032586, 0.44841236]])

1 l=[ "variance", "skewness"]
2 for a in range(1):
3     for b in range(2):
4         if l[a] !=l[b] :
5             x=dfl[l[a]]
6             y=dfl[l[b]]
7             x0=[]
8             y0=[]
9             x1=[]
10            y1=[]
11
12            for i in range(0,x.size):
13                if(KM.labels_[i] == 0):
14                    x0.append(x[i])
15                    y0.append(y[i])
16                else:
17                    x1.append(x[i])
18                    y1.append(y[i])
19
20
21 #Plotting the values and cluster centres
22 plt.figure(figsize=(10,10))
23 plt.scatter(x0,y0,c="red",s=5)
24 plt.scatter(x1,y1,c="green",s=5)
25 plt.scatter(centroids[:,a],centroids[:,b],c="black",s=25,marker="x")
26 plt.xlabel(l[a])
27 plt.ylabel(l[b])
```



در بلا تصویر دو خوشی که بهتر جدا شده اند را نشان می دهد.

و در ادامه نیز می بینیم که در این مدل خطای هم کمتر است.

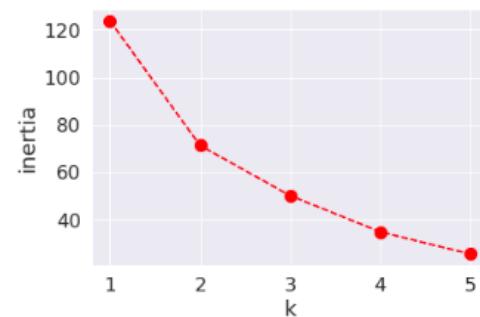
```
1 i=KM.inertia_
2 i
```

```
71.26245003881944
```

```
1 inertias=[]
2 for i in range(1,6):
3     k=KMeans(n_clusters=i).fit(dfl)
4     inertias.append(k.inertia_)
5     print(k.inertia_,"\\t",i)
```

```
123.74036629854533      1
71.26404884745804      2
50.14086348923321      3
34.9028417036666       4
25.540099447578893     5
```

```
1 plt.plot(range(1, 6), inertias, color='red', linestyle='dashed', marker='o',markerfacecolor='red', markersize=10
2 plt.xlabel("k")
3 plt.ylabel("inertia")
4 plt.show()
```



باز هم افت شدید را در دو خوشی می بینیم که مطالب قبلی را باز تأکید می کند.

۴. خوشه بندی سلسله مراتبی

```

1 from IPython.display import Image
2 import pandas as pd
3 from sklearn import datasets
4 from scipy.cluster.hierarchy import dendrogram, linkage ,fcluster
5 from scipy.spatial.distance import pdist
6 from matplotlib import pyplot as plt
7 iris = datasets.load_iris()
8 a=iris.data
9 df= pd.DataFrame(a,columns=iris.feature_names)
10 df

```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

۱. لینکج

```

1 distanceMatrix = pdist(iris.data,)
2 Z=linkage(distanceMatrix,'complete')
3 Z

```

array([[1.0100000e+02, 1.4200000e+02, 0.0000000e+00, 2.0000000e+00],	[7.0000000e+00, 3.9000000e+01, 1.0000000e-01, 2.0000000e+00],	[0.0000000e+00, 1.7000000e+01, 1.0000000e-01, 2.0000000e+00],	[9.0000000e+00, 3.4000000e+01, 1.0000000e-01, 2.0000000e+00],	[1.2800000e+02, 1.3200000e+02, 1.0000000e-01, 2.0000000e+00],	[1.0000000e+01, 4.8000000e+01, 1.0000000e-01, 2.0000000e+00],	[4.0000000e+00, 3.7000000e+01, 1.41421356e-01, 2.0000000e+00],	[1.9000000e+01, 2.1000000e+01, 1.41421356e-01, 2.0000000e+00],	[2.9000000e+01, 3.0000000e+01, 1.41421356e-01, 2.0000000e+00],	[5.7000000e+01, 9.3000000e+01, 1.41421356e-01, 2.0000000e+00],	[8.0000000e+01, 8.1000000e+01, 1.41421356e-01, 2.0000000e+00],	[1.1600000e+02, 1.3700000e+02, 1.41421356e-01, 2.0000000e+00],	[8.0000000e+00, 3.8000000e+01, 1.41421356e-01, 2.0000000e+00],	[3.0000000e+00, 4.7000000e+01, 1.41421356e-01, 2.0000000e+00],	[2.7000000e+01, 2.8000000e+01, 1.41421356e-01, 2.0000000e+00],	[8.2000000e+01, 9.2000000e+01, 1.41421356e-01, 2.0000000e+00],	[9.5000000e+01, 9.6000000e+01, 1.41421356e-01, 2.0000000e+00],	[1.2700000e+02, 1.3800000e+02, 1.41421356e-01, 2.0000000e+00],	[1.0000000e+00, 4.5000000e+01, 1.41421356e-01, 2.0000000e+00],	[6.3000000e+01, 9.1000000e+01, 1.41421356e-01, 2.0000000e+00]
--	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	---

سه نوع لینکج در درس معرفی (single , complete , average) شد که نوع کامپلیت باعث ایجاد خوشه های توزیع شده و دایره شکل می شود.

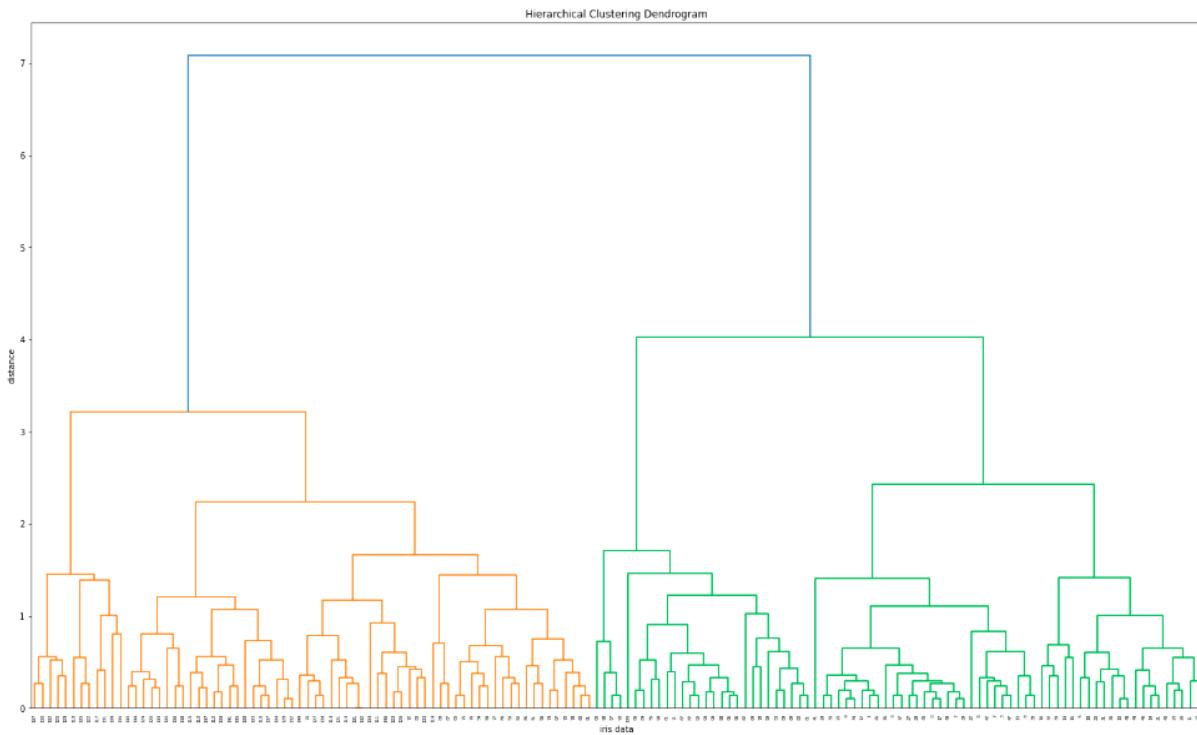
- method='complete' assigns

$$d(u, v) = \max(\text{dist}(u[i], v[j]))$$

در تصویر بالا متد complete در بالا تعریف شده است در واقع در این متد برای ساخت کلاستر ها از دورترین نقاط استفاده می کند.

۴.۲ مربوط به سوال قبلی dendrogram

```
1 plt.figure(figsize=(25, 15),edgecolor='blue')
2 plt.title('Hierarchical Clustering Dendrogram')
3 plt.xlabel('iris data')
4 plt.ylabel('distance')
5 dendrogram(Z)
6 plt.show()
```



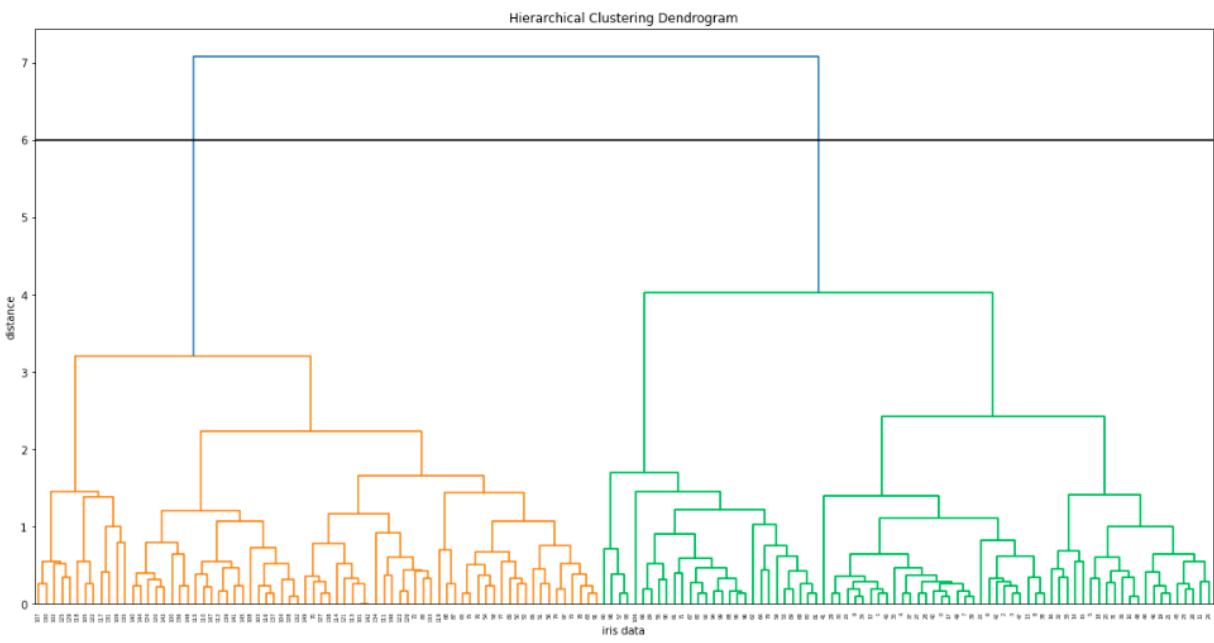
۴.۳ همانطور که گفته شد در سوال هرچه نمودار را بالاتر قطع کنیم تعداد کلاسترها کمتر خواهد بود در اگر

قطع کنیم داریم دو تا خوش بیشتر نداریم ۱ و ۲

```

1 plt.figure(figsize=(20, 10))
2 plt.title('Hierarchical Clustering Dendrogram')
3 plt.xlabel('iris data')
4 plt.ylabel('distance')
5 dendrogram(Z)
6 plt.axhline(y=6, color='black')
7 plt.show()
8 fcluster(Z, t=6, criterion='distance')
9

```



```

array([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, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 2, 2, 2, 1, 2, 1, 1, 1,
       2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 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], dtype=int32)

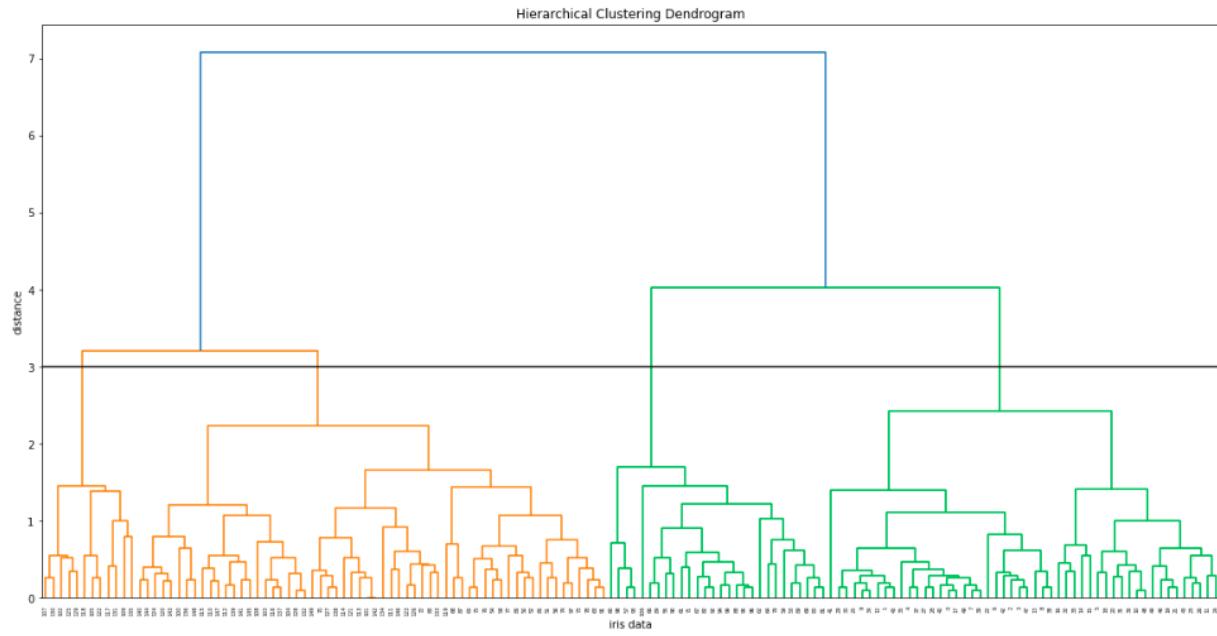
```

برای کلاستر با لول ۳ (۴ خوشه داریم).

```

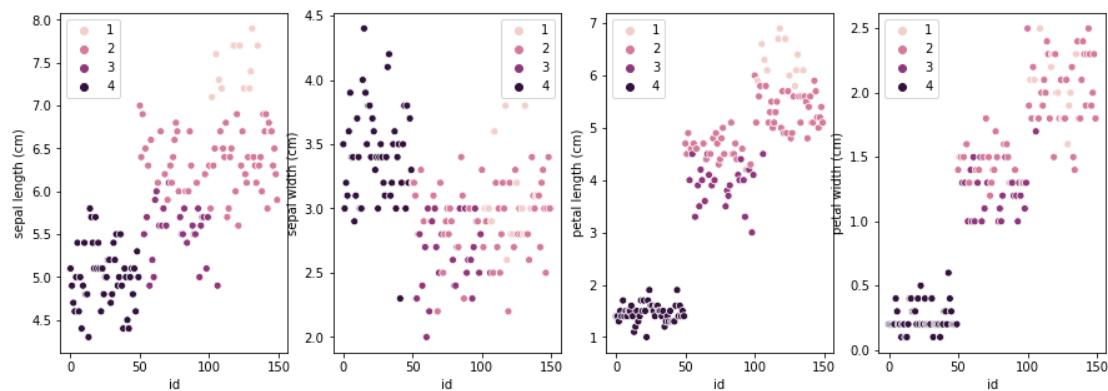
1 plt.figure(figsize=(20, 10))
2 plt.title('Hierarchical Clustering Dendrogram')
3 plt.xlabel('iris data')
4 plt.ylabel('distance')
5 dendrogram(Z)
6 plt.axhline(y=3, color='black')
7 plt.show()
8 fcluster(Z, t=3, criterion='distance')
9

```



```
array([4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 2, 2, 2, 3, 2, 3, 2, 3, 2, 3, 3, 3, 2, 3, 2, 3, 2,
       3, 3, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 2, 3, 2, 2, 2, 2,
       3, 3, 3, 2, 3, 3, 3, 3, 2, 3, 3, 2, 2, 1, 2, 2, 1, 3, 1, 2, 1,
       2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 1, 1,
       2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2], dtype=int32)
```

```
1 df1=df
2 df1['Species']=iris.target
3 df1 = df.reset_index()
4 df1['id'] = df1.index
5
6 f, axes = plt.subplots(1,4,figsize=(15,5))
7 y =iris.feature_names
8 ss=fcluster(Z, t=3, criterion='distance')
9 ax = 0
10 for y in y:
11     sns.scatterplot(x = 'id', y =y, hue = ss, data = df1, ax = axes[ax])
12     ax = ax + 1
13
```



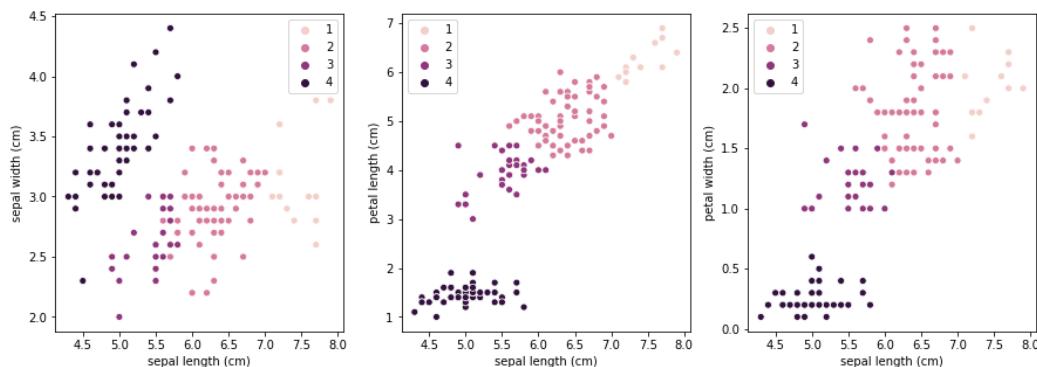
در بالا بر اساس id هر مورد scatter plot همه خوشه ها بر اساس هر متغیر رسم شده در نمودارهای بالا مرز بین خوشه ها ۴ و بقیه بهتر مشخص شده است و نمودار اول و دوم از سمت چپ هم مرز بین خوشه ها به خوبی مشخص شده است که این احتمالا از یک نوع ترتیب خاص در دیتاست صحبت می کند.

نمودار را برای تک تک ویژگی ها بر حسب یکدیگر رسم کردیم و خوش بندی را مشخص کردیم. برای مثال خوشه ۴ تقریبا همه جا مرز مشخصی از دیگر دسته ها دارد که این نشان می دهد خوش بندی خیلی خوبی انجام شده و همه بعد ها هم نقش خوبی در آن داشتند و بقیه موارد نیز مرز هایی می توان رسم کرد اما نه در همه دیاگرام ها \Leftarrow می توان بر اساس خوش بندی ها دسته های خوبی استخراج کرد. برای نمونه آن هایی که sepal length (cm) کمتر از ۶ هست و petal width (cm) شان کمتر 0.6 در خوشه ۴ قرار گرفته اند(شکل ۱ از سمت راست). توجه شود که برخی نمودار ها تکراری هستند و فقط x و y آن ها عوض شده است.

```

1 df1=df
2 df1['Species']=iris.target
3 df1 = df.reset_index()
4 df1['id'] = df1.index
5
6 f, axes = plt.subplots(1,3,figsize=(15,5))
7 y=['sepal width (cm)',
8 'petal length (cm)', 
9 'petal width (cm)']
10 ss=fcluster(Z, t=3, criterion='distance')
11 ax = 0
12 for y in y:
13     sns.scatterplot(x = 'sepal length (cm)', y = y, hue = ss, data = df1, ax = axes[ax])
14     ax = ax + 1

```

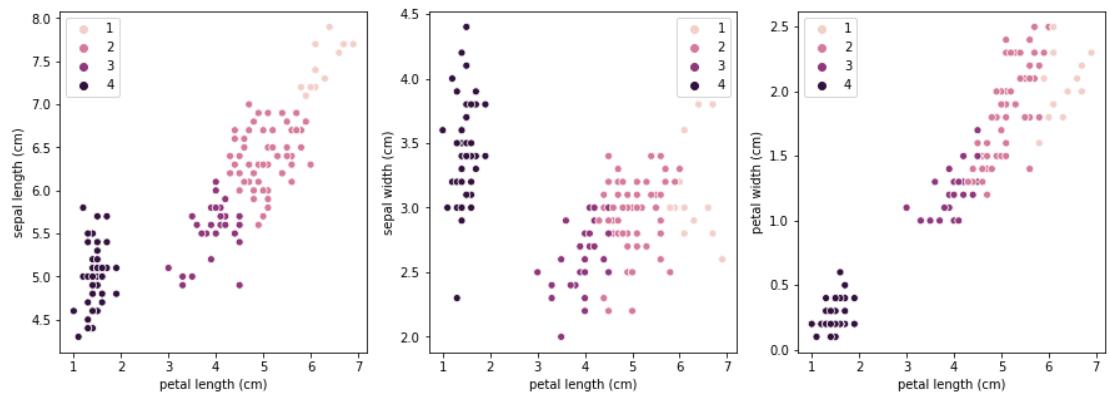


خوشه ها با معنای زیادی در هر کدام از نمودار ها می توانیم پیدا کنیم.

```

1 df1=df
2 df1['Species']=iris.target
3 df1 = df.reset_index()
4 df1['id'] = df1.index
5
6 f, axes = plt.subplots(1,3,figsize=(15,5))
7 y =['sepal length (cm)',
8 'sepal width (cm)',
9 'petal width (cm)']
10 ss=fcluster(Z, t=3, criterion='distance')
11 ax = 0
12 for y in y:
13     sns.scatterplot(x = 'petal length (cm)', y = y, hue = ss, data = df1, ax = axes[ax])
14     ax = ax + 1

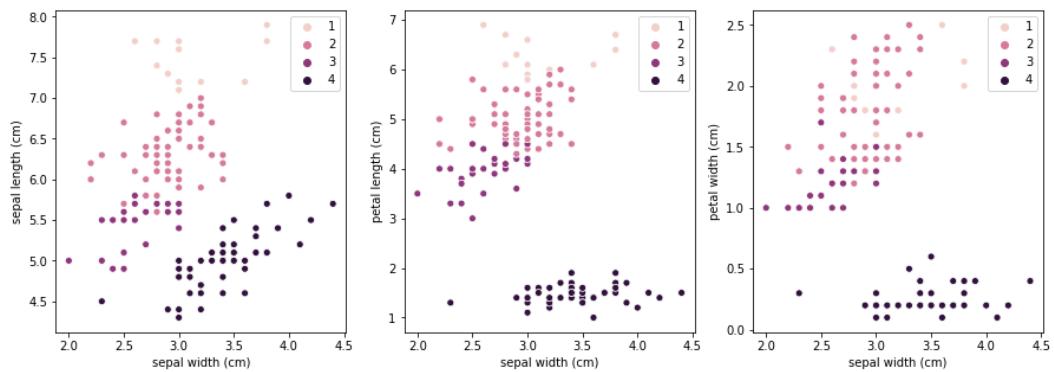
```



```

1 df1=df
2 df1['Species']=iris.target
3 df1 = df.reset_index()
4 df1['id'] = df1.index
5
6 f, axes = plt.subplots(1,3,figsize=(15,5))
7 y =['sepal length (cm)',
8 'petal length (cm)',
9 'petal width (cm)']
10 ss=fcluster(Z, t=3, criterion='distance')
11 ax = 0
12 for y in y:
13     sns.scatterplot(x = 'sepal width (cm)', y = y, hue = ss, data = df1, ax = axes[ax])
14     ax = ax + 1

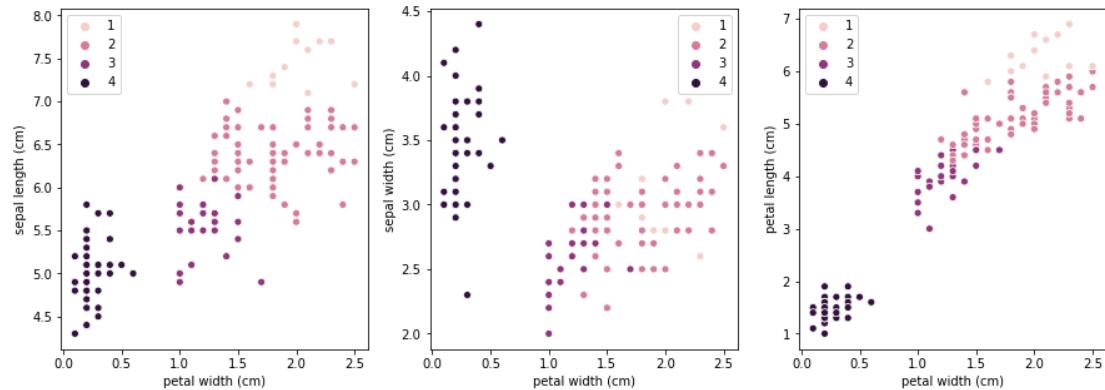
```



```

1 df1=df
2 df1['Species']=iris.target
3 df1 = df.reset_index()
4 df1['id'] = df1.index
5
6 f, axes = plt.subplots(1,3,figsize=(15,5))
7 y =['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)']
8 ss=fcluster(Z, t=3, criterion='distance')
9 ax = 0
10 for y in y:
11     sns.scatterplot(x = 'petal width (cm)', y = y, hue = ss, data = df1, ax = axes[ax])
12     ax = ax + 1

```



۵. رگرسیون

در زیر ما در قسمت keys آنچه که در دیتاست هست را می بینیم و همچنین نام ویژگی ها را می توانیم ببینیم . ما در این سوال ابتدا آمدیم و استاندارد سازی انجام دادیم چون رفع متغیر ها متفاوت است.(و نتایج در اوردر خطأ تفاوت داشتند).

```

1 from sklearn.datasets import load_boston
2 import numpy as np
3 import pandas as pd
4 import pandas as pd
5 boston = load_boston()

```

```

1 print("keys: ",boston.keys(),"\n")
2 print("feature_names: ",boston.feature_names)

keys: dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])

feature_names: ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
'B' 'LSTAT']

```

۱.۵. لود کردن دیتا

توضیحاتی در مورد دیتا سنت مورد نظر به همراه نام و تعریفی از ویژگی هایش را در ادامه می بینیم. نشان داده شده که داده گمشده ندارد.

```

1 | print(boston.DESCR)
.. _boston_dataset:
Boston house prices dataset
-----
**Data Set Characteristics:**

:Number of Instances: 506
:Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.
:Attribute Information (in order):
- CRIM per capita crime rate by town
- ZN proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS proportion of non-retail business acres per town
- CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX nitric oxides concentration (parts per 10 million)
- RM average number of rooms per dwelling
- AGE proportion of owner-occupied units built prior to 1940
- DIS weighted distances to five Boston employment centres
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per $10,000
- PTRATIO pupil-teacher ratio by town
- B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT % lower status of the population
- MEDV Median value of owner-occupied homes in $1000's

:Missing Attribute Values: None
:Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/

```

This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

ساخت دیتا فریم .۵،۲

```

1 | X = pd.DataFrame(boston.data, columns = boston.feature_names)
2 | X.head()

```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

اضافه کردن متغیر هدف به دیتا فریم و ذخیره دیتاست .۵،۳

```

1 | df['Price']=boston.target
2 | df.head()

```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```

1 | df.to_csv('Boston.csv')

```

```

1 b = df.values
2 from sklearn.preprocessing import StandardScaler
3 scaler = StandardScaler().fit(df)
4 standardized = scaler.transform(df)
5 df= pd.DataFrame(standardized,columns=["CRIM", 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO',
6 df

```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140214	-0.982843	-0.666608	-1.459000	0.441052	-1.075562	0.159686
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557160	-0.867883	-0.987329	-0.303094	0.441052	-0.492439	-0.101524
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557160	-0.867883	-0.987329	-0.303094	0.396427	-1.208727	1.324247
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077737	-0.752922	-1.106115	0.113032	0.416163	-1.361517	1.182758
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077737	-0.752922	-1.106115	0.113032	0.441052	-1.026501	1.487503
...
501	-0.413229	-0.487722	0.115738	-0.272599	0.158124	0.439316	0.018673	-0.625796	-0.982843	-0.803212	1.176466	0.387217	-0.418147	-0.014454
502	-0.415249	-0.487722	0.115738	-0.272599	0.158124	-0.234548	0.288933	-0.716639	-0.982843	-0.803212	1.176466	0.441052	-0.500850	-0.210362
503	-0.413447	-0.487722	0.115738	-0.272599	0.158124	0.984960	0.797449	-0.773684	-0.982843	-0.803212	1.176466	0.441052	-0.983048	0.148802
504	-0.407764	-0.487722	0.115738	-0.272599	0.158124	0.725672	0.736996	-0.668437	-0.982843	-0.803212	1.176466	0.403225	-0.865302	-0.057989
505	-0.415000	-0.487722	0.115738	-0.272599	0.158124	-0.362767	0.434732	-0.613246	-0.982843	-0.803212	1.176466	0.441052	-0.669058	-1.157248

506 rows × 14 columns

۴. جدا کردن داده های train و test .

```

1 X=df[boston.feature_names]
2 y=df['Price']
3 from sklearn.model_selection import train_test_split
4 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

```

۵. مدلسازی با استفاده از روش رگرسیون .

در مدل سازی با توجه به این که همبستگی بین TAX و RAD بسیار بالاست شاید بهتر می باشد حذف کنیم یکی از این دو را ← RAD چون با قیمت همبستگی کمتری دارد گزینه مناسب تری است برای حذف ← اما نهایتا باید بررسی شود که آیا دقت مدل بیشتر می شود یا خیر؟!

از طرفی دیگر برخی از پارامترها همبستگی خیلی کمی با هدف دارند (مثل DIS و CHAS) ← باید بررسی شوند چون در این سوال مدل رگرسیون است و همبستگی بین داده ها با هدف کمک کننده است.

```

1 from sklearn.linear_model import LinearRegression
2 reg = LinearRegression()
3 reg.fit(X_train, y_train)
4 pred_y = reg.predict(X_test)

```

۶. MSE .

تابع Mean_absolute_error میانگین خطای مطلق را محاسبه می کند ، که یک معیار خطر متناسب با

مقدار مورد انتظار از خطای مطلق یا از $\| \cdot \|_1$ -norm است و از فرمول زیر بدست می آید:

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|$$

تابع Mean_squared_error میانگین خطای مربع را محاسبه می کند ، یک معیار خطری متناسب با مقدار مورد

انتظار خطای مربعات (درجه دوم) است و از فرمول زیر بدست می آید:

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

۲R اندازه گیری آماری نزدیک داده ها به خط رگرسیون برازش شده میباشد. به R^2 ضریب تعیین یا ضریب تشخیص نیز گفته می شود. تعریف ضریب تعیین (R^2) نسبتاً ساده است: ضریب تعیین (R^2) نشان میدهد که چند درصد تغییرات متغیر وابسته به وسیله متغیر مستقل تبیین می شود یا به عبارت دیگر ضریب تعیین نشان دهنده این است که چه مقدار از تغییرات متغیر وابسته تحت تاثیر متغیر مستقل مربوطه بوده و مابقی تغییرات متغیر وابسته مربوط به سایر عوامل میباشد." و از فرمول زیر بدست می آید:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

که در آن \hat{y}_i حد مقدار پیش‌بینی شده عنصر i ام و \bar{y} مقدار واقعی آن است.

خطا بد نیست و چندان زیاد نیست و قدار $2R$ هم پایین* نیست \Leftrightarrow مدل بدی نیست.
 R^2 * پایین هم همیشه به معنای بد بودن مدل نیست.

```
1 from sklearn import metrics
2 print("MSE: ",metrics.mean_squared_error(y_test, pred_y))
MSE:  0.3832295951048717

1 metrics.mean_absolute_error(y_test, pred_y)
0.39466718732579364

1 reg.score(X_test, y_test)
0.6423604634152575
```

۵. نتایج برای این قسمت با $5=CV$.

ساده ترین روش برای استفاده از اعتبار سنجی متقابل ، فراخوانی عملکرد یاور `cross_val_score` در برآوردگر و مجموعه داده است.

```
1 from sklearn.model_selection import cross_val_score
2 scores = cross_val_score(reg, boston.data, boston.target, cv=5)
3 print("scores: ",scores)
4 print("Accuracy: %.2f (+/- %.2f)" % (scores.mean(), scores.std() * 2))

scores: [ 0.63919994  0.71386698  0.58702344  0.07923081 -0.25294154]
Accuracy: 0.35 (+/- 0.75)
```

برای ارزیابی متریک چند ، مقدار بازده بر اساس کلیدهای زیر می باشد:

`['test_<scorer1_name>','test_<scorer2_name>','test_<scorer...>','fit_time','score_time']`
در اسکور محاسبه می شود و مشاهده می کنیم که 0.35 می باشد بطور میانگین!
Confusion Matrix و ROC .۶

```
1 import pandas as pd
2 from sklearn.datasets import load_breast_cancer
3 from sklearn.model_selection import train_test_split
4 from sklearn import svm
5 import numpy as np
```

۶.۱ لود کردن داده ها

```
1 cancer = load_breast_cancer()
```

۶.۲. جدا کردن داده های train و test و مدلسازی بردار ماشین پشتیبان

۶.۳ ایمپورت کردن دو متده گفته شده

```
1 from sklearn.metrics import classification_report, confusion_matrix
```

۶.۴ بررسی خروجی confusion_matrix

```
1 c=pd.DataFrame(confusion_matrix(y_test,y_pred) , index=['RealCancer','RealHealthy'],columns=['PredictionCancer'])  
2 c
```

	PredictionCancer	PredictionHealthy
RealCancer	46	1
RealHealthy	4	63

FN=1

TP-47

TN=۶۳

$$\text{FP} = \xi$$

در واقع در ردیف اول از ۴۷ بیماری که سرطان دارند ۴۶ بیمار را به درستی مدل پیش‌بینی کرده (TP) و یک مورد را به اشتباه سالم پیش‌بینی کرده (FN) است.

از ۶۷ فرد سالم ۴ نفر را به اشتباه برایشان سرطان پیشگیری کرده (FN) ولی بقیه درست اند (TN).
ما در مدل باید TP و TN بالایی داشته باشیم، و مواردی که غلط تشخیص داده می شوند بسته به مسیله هزینه متفاوتی به ما تحمیل می کنند. مثلا در این مورد مواردی که سرطان دارند ولی سالم تشخیص داده می شوند توسط مدل هزینه زیادی تحمیل می کنند.

classification_report

		Condition Phase (Worst Case)		
		Condition Positive/ Shaded	Condition Negative/ Unshaded	
Testing Phase (Best Case)	Test Positive/ Shaded	True positive shaded T_p (Correct)	False positive shaded F_p (Incorrect)	Precision/Positive Predictive Value (PPV) $\frac{T_p}{T_p + F_p} \times 100\%$
	Test Negative/ Unshaded	False negative unshaded F_n (Incorrect)	True negative unshaded T_n (Correct)	Negative Predictive Value (NPV) $\frac{T_n}{T_n + F_n} \times 100\%$
		Sensitivity/Recall Rate (RR) $\frac{T_p}{T_p + F_n} \times 100\%$	Specificity Rate (SR) $\frac{T_n}{T_n + F_p} \times 100\%$	

```
1 | print(classification_report(y_test,y_pred))
   precision    recall  f1-score   support
0           0.92      0.98      0.95       47
1           0.98      0.94      0.96       67
   accuracy                           0.96      114
  macro avg       0.95      0.96      0.96      114
weighted avg       0.96      0.96      0.96      114
```

۶.۶ نرمال سازی نتایج confusion_matrix

```
1 | from sklearn import preprocessing
2 | normalize=preprocessing.normalize(c, norm='l1', axis=1)
3 |
4 |
array([[ 0.9787234 ,  0.0212766 ],
       [ 0.05970149,  0.94029851]])
```

۶.۷ ذخیره نتایج به صورت دیتا فریم

```
1 | d=pd.DataFrame(normalize, index=['Malignant','Benign'],columns=['PredictionMalignant','PredictionBenign'])
2 | d
```

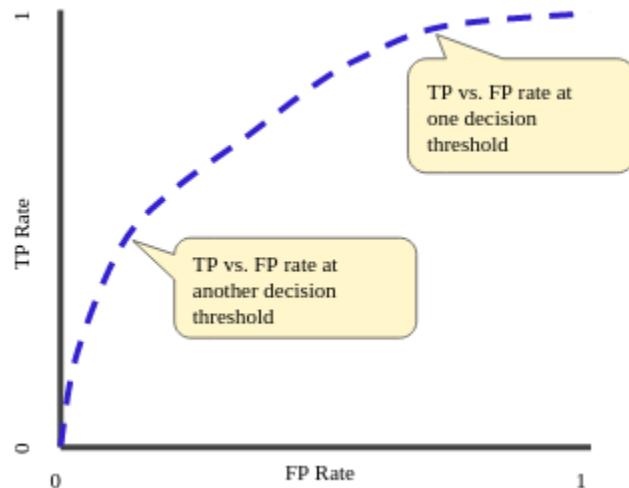
	PredictionMalignant	PredictionBenign
Malignant	0.978723	0.021277
Benign	0.059701	0.940299

۶.۸ ROC تحلیل نمودار های

منحنی ROC (منحنی مشخصه عامل گیرنده) یک نمودار است که عملکرد یک مدل طبقه بندی را در همه آستانه های طبقه بندی نشان می دهد. این منحنی دو پارامتر را ترسیم می کند:

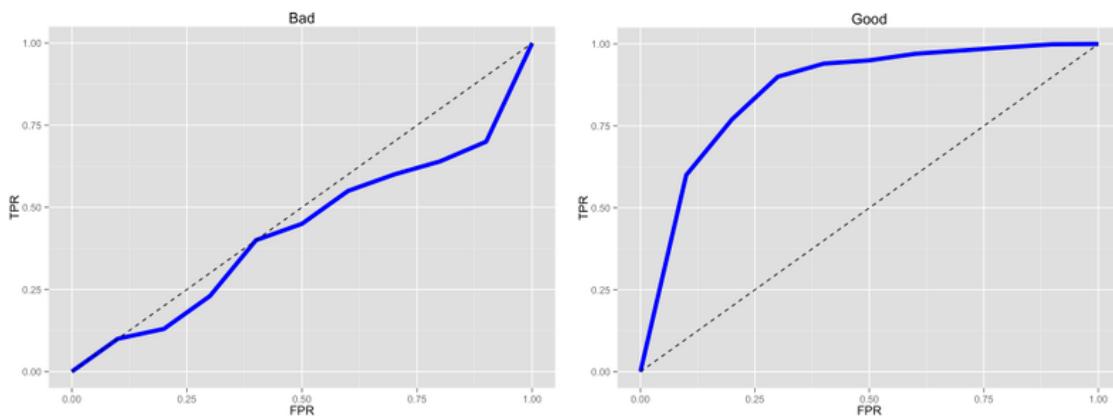
- True Positive Rate (TPR)
- False Positive Rate (FPR)

منحنی ROC، TPR و FPR را در آستانه های طبقه بندی مختلف ترسیم می کند. پایین آمدن آستانه طبقه بندی ، موارد بیشتری را به عنوان مثبت طبقه بندی می کند ، بنابراین باعث افزایش هر دو مثبت کاذب و مثبت می شود. شکل زیر منحنی ROC معمولی را نشان می دهد.



برای محاسبه نقاط در یک منحنی ROC ، می توانیم چندین بار مدل رگرسیون لجستیک را با آستانه طبقه بندی متفاوت ارزیابی کنیم ، اما این می تواند ناکارآمد باشد. خوشبختانه ، یک الگوریتم کارآمد به نام AUC وجود دارد که می تواند این اطلاعات را برای ما فراهم کند.

AUC مخفف "ناحیه زیر منحنی ROC" است. AUC از ۰ تا ۱ متغیر است. الگویی که پیش بینی های آنها ۱۰۰٪ اشتباه است ، دارای AUC ۰.۰ است. کسی که پیش بینی های آن صدرصد صحیح باشد دارای AUC ۱.۰ است.



با توجه به توضیحات بالا در مدل شکل سمت راست تعداد مواردی که به درستی پیش‌بینی می کند به نسبت مواردی که اشتباه پیش‌بینی می کند بسیار بالاتر از مدلی است که در سمت چپ نمودار ROC اش رسم شده است. ۶.۹ predict_proba استفاده از متده است.

```

1 y_predict_proba=SV.predict_proba(X_test)
2 y_predict_proba
array([[9.53794103e-01, 4.62058968e-02],
       [8.63246558e-02, 9.13675344e-01],
       [8.74896559e-03, 9.91251034e-01],
       [1.60582139e-01, 8.39417861e-01],
       [1.25918867e-06, 9.99998741e-01],
       [1.15416721e-02, 9.88458328e-01],
       [2.30054115e-02, 9.76994589e-01],
       [6.80154705e-03, 9.93198453e-01],
       [1.13094387e-02, 9.88690561e-01],
       [8.38870493e-07, 9.99999161e-01],
       [3.51920988e-01, 6.48079012e-01],
       [2.30233818e-01, 7.69766182e-01],
       [6.15948800e-03, 9.93840512e-01],
       [4.48409489e-01, 5.51590511e-01],
       [2.65102780e-01, 7.34897220e-01],
       [9.20327713e-01, 7.96722874e-02],
       [1.60902651e-02, 9.83909735e-01],
       [9.99966727e-01, 3.32734782e-05],
       [9.52183972e-01, 4.78160281e-02],
       [9.99999161e-01, 1.84035156e-07])

```

ROC نتایج .۶,۱۰

```

1 from sklearn.metrics import roc_curve ,auc
2 import matplotlib.pyplot as plt
3 fpr, tpr, thresholds=roc_curve(y_test,y_pred)
4 print('fpr=',fpr,'\\n','tpr=',tpr,'\\n','thresholds=' , thresholds)
5 roc_auc=auc(fpr,tpr)
6 print('ROC AUC: %0.2f' % roc_auc)

fpr= [0.          0.0212766 1.          ]
tpr= [0.          0.94029851 1.          ]
thresholds= [2 1 0]
ROC AUC: 0.96

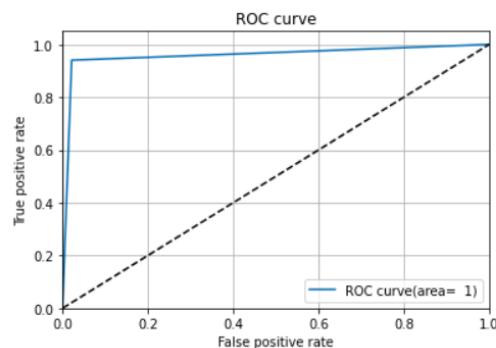
```

ROC نمایش .۶,۱۱

```

1 #plot of ROC curve for a specified class
2 plt.figure()
3 plt.plot(fpr,tpr,label='ROC curve(area= %2.f)' %roc_auc)
4 plt.plot([0,1],[0,1],'k--')
5 plt.xlim([0.0,1.0])
6 plt.ylim([0.0,1.05])
7 plt.xlabel('False positive rate')
8 plt.ylabel('True positive rate')
9 plt.title('ROC curve')
10 plt.legend(loc='lower right')
11 plt.grid()
12 plt.show()

```



مدل خیلی خوب عمل کرده است و نسبت پیش بینی های درست به اشتباهش ۰.۹۶ است که مقدار خوبی است.

قوانين وابستگی .۷

در این دیتاست هر کدام از ویژگی ها به شرح زیر اند:
 InvoiceNo: شماره فاکتور، یک عدد ۶ رقمی است که به طور اختصاصی به هر معامله اختصاص می یابد. اگر این کد با حرف "C" شروع شود، نشانگر لغو آن است.
 StockCode: کد محصول (مورد). اسمی، یک عدد ۵ رقمی است که به طور خاص به هر محصول متمایز اختصاص داده شده است.

Description: نام محصول (مورد) اسمی.

Quantity: مقدار هر محصول (مورد) برای هر تراکنش. عددی InvoiceDate: تاریخ و زمان Invice عددی، روز و زمان تولید هر تراکنش.
 UnitPrice: قیمت واحد. عددی، قیمت محصول در هر واحد در استرلینگ.
 CustomerID: شماره مشتری. اسمی، یک عدد انتگرالی ۵ رقمی است که به طور اختصاصی به هر مشتری اختصاص می یابد.
 Country: نام کشور اسمی، نام کشوری که هر مشتری در آن سکونت دارد.

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
```

۷.۱. خواندن دیتاست

```
1 df = pd.read_excel("Online Retail.xlsx")
2 df
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

541909 rows × 8 columns

۷.۲. ایمپورت های گفته شده

```
1 from mlxtend.frequent_patterns import apriori
2 from mlxtend.frequent_patterns import association_rules
```

۷.۳. حذف بلانک ها

strip(): فاصله ها را در ابتداء و انتهای رشته حذف می کند.
 برای این که بتوانیم از متاد بالا استفاده کنیم چون در دیتافریم ما object داریم لازم است آن ها را تبدیل به رشته کنیم.
 بنظر می رسد اصلا جای خالی ندارد.

```

1 df['Description'] = df['Description'].str.strip()
2 df

```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

541909 rows × 8 columns

حذف رکوردهای خالی InvoiceNO .۴،۷

```

1 df.dropna(subset=['InvoiceNo'], inplace=True)
2 df['InvoiceNo'] = df['InvoiceNo'].astype('str')
3 df

```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

541909 rows × 8 columns

حذف تراکنش های دارای حرف C (حذف تراکنش های لغو شده) InvoiceNO .۷،۵

```

1 df = df[~df['InvoiceNo'].str.contains('C')]
2 df

```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

532621 rows × 8 columns

۷.۶ توضیح دستور گفته شده (در این سوال ابتدا تراکنش ها را محدود به کشور فرانسه کرده و برای هر تراکنشی یک سبد خرید تعریف می کند تا بداند هر تراکنشی چه مواردی را و به چه تعداد (مقداری sum) خریداری کرده و با unstack می آید توضیحات محصول را بالا می برد و باعث می شود که هر تراکنش یک رکورد داشته باشد شامل تمام محصولات که خریده و نخرید و بار از نو دیگر ایندکس گذاری می کند و برای مقادیری که چیزی تعریف نشده (در یک تراکنشی اصلا یک محصول را نخریده ← مقدار تعریف نشده یا null است) مقدار را صفر قرار می دهد و نهایتاً مقدار ایندکس را شماره فاکتور قرار می دهد)

محور یک سطح از برچسب های شاخص (لزوماً سلسله مرتبی) است.

```

1 basket=df[df['Country']=="France"].groupby(["InvoiceNo", 'Description'])['Quantity'].sum().unstack().reset_index()
2 basket

```

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT	12 PENCILS TALL TUBE WOODLAND	WRAP VINTAGE PETALS DESIGN	YE FAS
InvoiceNo												
536370	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536852	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536974	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537065	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537463	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...
580986	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
581001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
581171	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
581279	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
581587	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

392 rows × 1563 columns

۷.۷ نوشتنتابع گفته شده (در واقع می آید وزن خرید را از بین می برد و فقط به این که یک کالا در یک تراکنش خریداری شده یا نشده بسته می کند)

```

1 def encode_units(x):
2     if x > 0:
3         return 1
4     else:
5         return 0
6
7 basket_sets = basket.applymap(encode_units)
8 basket_sets

```

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT	12 PENCILS TALL TUBE WOODLAND	... WRAP VINTAGE PETALS DESIGN	YE FA
InvoiceNo												
536370	0	0	0	0	0	0	0	0	0	0	0	0
536852	0	0	0	0	0	0	0	0	0	0	0	0
536974	0	0	0	0	0	0	0	0	0	0	0	0
537065	0	0	0	0	0	0	0	0	0	0	0	0
537463	0	0	0	0	0	0	0	0	0	0	0	0
...
580986	0	0	0	0	0	0	0	0	0	0	0	0
581001	0	0	0	0	0	0	0	0	0	0	0	0
581171	0	0	0	0	0	0	0	0	0	0	0	0
581279	0	0	0	0	0	0	0	0	0	0	0	0
581587	0	0	0	0	0	0	0	0	0	0	0	0

392 rows × 1563 columns

.۷،۸ POSTAGE حذف ستون

```

1 basket_sets.drop('POSTAGE', inplace=True, axis=1)
2 basket_sets

```

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT	12 PENCILS TALL TUBE WOODLAND	... WRAP VINTAGE PETALS DESIGN	YE FA
InvoiceNo												
536370	0	0	0	0	0	0	0	0	0	0	0	0
536852	0	0	0	0	0	0	0	0	0	0	0	0
536974	0	0	0	0	0	0	0	0	0	0	0	0
537065	0	0	0	0	0	0	0	0	0	0	0	0
537463	0	0	0	0	0	0	0	0	0	0	0	0
...
580986	0	0	0	0	0	0	0	0	0	0	0	0
581001	0	0	0	0	0	0	0	0	0	0	0	0
581171	0	0	0	0	0	0	0	0	0	0	0	0
581279	0	0	0	0	0	0	0	0	0	0	0	0
581587	0	0	0	0	0	0	0	0	0	0	0	0

392 rows × 1562 columns

.۷،۹ frequent item support با sets حداقل

منظور از support از ۷ درصدی این است که به میزان ۷ تکرار شده باشد در دیتابست.

```
1 frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
2 frequent_itemsets
```

	support	itemsets
0	0.071429	(4 TRADITIONAL SPINNING TOPS)
1	0.096939	(ALARM CLOCK BAKELIKE GREEN)
2	0.102041	(ALARM CLOCK BAKELIKE PINK)
3	0.094388	(ALARM CLOCK BAKELIKE RED)
4	0.081633	(BAKING SET 9 PIECE RETROSPOT)
5	0.071429	(CHILDRENS CUTLERY DOLLY GIRL)
6	0.099490	(DOLLY GIRL LUNCH BOX)
7	0.096939	(JUMBO BAG RED RETROSPOT)
8	0.076531	(JUMBO BAG WOODLAND ANIMALS)
9	0.125000	(LUNCH BAG APPLE DESIGN)
10	0.084184	(LUNCH BAG DOLLY GIRL DESIGN)
11	0.153061	(LUNCH BAG RED RETROSPOT)
12	0.119898	(LUNCH BAG SPACEBOY DESIGN)
13	0.117347	(LUNCH BAG WOODLAND)
14	0.142857	(LUNCH BOX WITH CUTLERY RETROSPOT)
15	0.104592	(MINI PAINT SET VINTAGE)
16	0.102041	(PACK OF 72 RETROSPOT CAKE CASES)
17	0.081633	(PAPER BUNTING RETROSPOT)
18	0.168367	(PLASTERS IN TIN CIRCUS PARADE)
19	0.137755	(PLASTERS IN TIN SPACEBOY)
20	0.081633	(PLASTERS IN TIN STRONGMAN)
21	0.170010	(PLASTERS IN TIN WOODLAND ANIMALS)

20	0.081633	(PLASTERS IN TIN STRONGMAN)
21	0.170918	(PLASTERS IN TIN WOODLAND ANIMALS)
22	0.188776	(RABBIT NIGHT LIGHT)
23	0.096939	(RED RETROSPOT CHARLOTTE BAG)
24	0.137755	(RED RETROSPOT MINI CASES)
25	0.071429	(RED RETROSPOT PICNIC BAG)
26	0.181122	(RED TOADSTOOL LED NIGHT LIGHT)
27	0.125000	(REGENCY CAKESTAND 3 TIER)
28	0.086735	(RETROSPOT TEA SET CERAMIC 11 PC)
29	0.107143	(ROUND SNACK BOXES SET OF 4 FRUITS)
30	0.158163	(ROUND SNACK BOXES SET OF 4 WOODLAND)
31	0.076531	(SET/10 RED POLKA DOT PARTY CANDLES)
32	0.132653	(SET/20 RED RETROSPOT PAPER NAPKINS)
33	0.137755	(SET/6 RED SPOTTY PAPER CUPS)
34	0.127551	(SET/6 RED SPOTTY PAPER PLATES)
35	0.071429	(SPACEBOY BIRTHDAY CARD)
36	0.125000	(SPACEBOY LUNCH BOX)
37	0.122449	(STRAWBERRY LUNCH BOX WITH CUTLERY)
38	0.094388	(TEA PARTY BIRTHDAY CARD)
39	0.073980	(WOODLAND CHARLOTTE BAG)
40	0.073980	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...
41	0.079082	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...
42	0.073980	(ALARM CLOCK BAKELIKE PINK, ALARM CLOCK BAKEL...
43	0.071429	(DOLLY GIRL LUNCH BOX, SPACEBOY LUNCH BOX)
44	0.089286	(PLASTERS IN TIN CIRCUS PARADE, PLASTERS IN TI...
45	0.102041	(PLASTERS IN TIN WOODLAND ANIMALS, PLASTERS IN...
46	0.104592	(PLASTERS IN TIN WOODLAND ANIMALS, PLASTERS IN...
47	0.102041	(SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...
48	0.102041	(SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...
49	0.122449	(SET/6 RED SPOTTY PAPER CUPS, SET/6 RED SPOTTY...
50	0.099490	(SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...

۷.۱۰ تولید قوانین وابستگی

می توانیم ببینیم که چند قانون کاملاً دارای ارزش بالابر بالا وجود دارد که با توجه به تعداد تراکنش ها و ترکیبات محصول ، بیشتر از آنچه انتظار می رود اتفاق می افتد. همچنین می توان چندین مورد را دید که conf در آنها زیاد است.

```

1 rules = association_rules(frequent_itemsets, metric="lift")
2 rules

```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE PINK)	0.096939	0.102041	0.073980	0.763158	7.478947	0.064088	3.791383
1	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE GREEN)	0.102041	0.096939	0.073980	0.725000	7.478947	0.064088	3.283859
2	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959	0.069932	4.916181
3	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959	0.069932	5.568878
4	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE RED)	0.102041	0.094388	0.073980	0.725000	7.681081	0.064348	3.293135
5	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE PINK)	0.094388	0.102041	0.073980	0.783784	7.681081	0.064348	4.153061
6	(DOLLY GIRL LUNCH BOX)	(SPACEBOY LUNCH BOX)	0.099490	0.125000	0.071429	0.717949	5.743590	0.058992	3.102273
7	(SPACEBOY LUNCH BOX)	(DOLLY GIRL LUNCH BOX)	0.125000	0.099490	0.071429	0.571429	5.743590	0.058992	2.101190
8	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN SPACEBOY)	0.168367	0.137755	0.089286	0.530303	3.849607	0.066092	1.835747
9	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN CIRCUS PARADE)	0.137755	0.168367	0.089286	0.648148	3.849607	0.066092	2.363588
10	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN CIRCUS PARADE)	0.170918	0.168367	0.102041	0.597015	3.545907	0.073264	2.063681
11	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.168367	0.170918	0.102041	0.606061	3.545907	0.073264	2.104592
12	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN SPACEBOY)	0.170918	0.137755	0.104592	0.611940	4.442233	0.081047	2.221939
13	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.137755	0.170918	0.104592	0.759259	4.442233	0.081047	3.443878
14	(SET/20 RED RETROSPOT PAPER NAPKINS)	(SET/6 RED SPOTTY PAPER CUPS)	0.132653	0.137755	0.102041	0.769231	5.584046	0.083767	3.736395
15	(SET/6 RED SPOTTY PAPER CUPS)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.137755	0.132653	0.102041	0.740741	5.584046	0.083767	3.345481
16	(SET/20 RED RETROSPOT PAPER NAPKINS)	(SET/6 RED SPOTTY PAPER PLATES)	0.132653	0.127551	0.102041	0.769231	6.030769	0.085121	3.780612
17	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.132653	0.102041	0.800000	6.030769	0.085121	4.336735
18	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.127551	0.122449	0.888889	6.968889	0.104878	7.852041
19	(SET/6 RED SPOTTY PAPER)	(SET/6 RED SPOTTY PAPER)	0.127551	0.137755	0.122449	0.960000	6.968889	0.104878	21.556122

۷،۱۱ فیلتر کردن قواعد

```

1 rules[ (rules['lift'] >= 6) & (rules['confidence'] >= 0.8) ]

```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
2	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959	0.069932	4.916181
3	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959	0.069932	5.568878
17	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.132653	0.102041	0.800000	6.030769	0.085121	4.336735
18	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.127551	0.122449	0.888889	6.968889	0.104878	7.852041
19	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.127551	0.137755	0.122449	0.960000	6.968889	0.104878	21.556122
20	(SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.127551	0.099490	0.975000	7.644000	0.086474	34.897959
21	(SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.099490	0.975000	7.077778	0.085433	34.489796
22	(SET/6 RED SPOTTY PAPER CUPS, SET/6 RED SPOTTY...	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.122449	0.132653	0.099490	0.812500	6.125000	0.083247	4.625850

۷،۱۲ تفسیر یکی از قوانین

با نگاهی به قوانین ، به نظر می رسد که ساعتهای زنگ سبز و قرمز با هم خریداری می شوند.
اگر بخواهیم یکی از قوانین را توصیف کنیم داریم:

If (ALARM CLOCK BAKELIKE GREEN) then (ALARM CLOCK BAKELIKE RED)

هر کس ساعت سبز بخرد به احتمال بالا ساعت قرمز هم می خرد. با توجه به تراکنش هایی که ساعت سبز می خرند $P(A=conf) = 0.81$
 $P(B|x) = 0.81$

و درصد تراکنش هایی که ساعت سبز و قرمز می خرند حدوداً ۸۱٪ درصد است.

در واقع ۸۱٪ است به معنی این که اگر کسی ساعت سبز خرید ۸۱٪ برابر بیشتر احتمال دارد که ساعت قرمز را هم بخرد.

Naive Bayes .۸

.۸.۱ مدل سازی برای پیش‌بینی متن.

.۸.۲ دیتاست ۲۰NewsGroups را لود می کنیم (و اهداف یا برچسب هایی که وجود دارد را مشاهده می کنیم)

```
1 from sklearn.datasets import fetch_20newsgroups
2 import pandas as pd
3 data=fetch_20newsgroups()
4 data.target_names

['alt.atheism',
'comp.graphics',
'comp.os.ms-windows.misc',
'comp.sys.ibm.pc.hardware',
'comp.sys.mac.hardware',
'comp.windows.x',
'misc.forsale',
'rec.autos',
'rec.motorcycles',
'rec.sport.baseball',
'rec.sport.hockey',
'sci.crypt',
'sci.electronics',
'sci.med',
'sci.space',
'soc.religion.christian',
'talk.politics.guns',
'talk.politics.mideast',
'talk.politics.misc',
'talk.religion.misc']
```

.۸.۳ تخصیص داده های ترین و تست با استفاده از آرایه گفته شده در سوال (یک تعداد خوبی از برچسب ها را حذف می کنیم)

```
1 categories = ['talk.religion.misc','soc.religion.christian','sci.space','comp.graphics']
2 train = fetch_20newsgroups(subset='train', categories=categories)
3 test = fetch_20newsgroups(subset='test', categories=categories)
```

.۸.۴ هر داده دریافت شده تبدیل به یک بردار می کنیم تا آن مدل سازی کنیم

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 from sklearn.naive_bayes import MultinomialNB
3 vectorizer = TfidfVectorizer()
4 Xtrain = vectorizer.fit_transform(train.data)
5 Xtest=vectorizer.transform(test.data)
6 ytrain=train.target
7 ytest=test.target
```

.۸.۵ تفاوت TF با TF-IDF

Tf مخفف کلمه Term frequency است به معنی فراوانی اصطلاح (منظور کلمه یا ترم است) که نشان می دهد در سند مورد نظر یک کلمه چند بار تکرار شده است و سپس بر همین اساس یک وزن می دهد به هر اصطلاح (منظور کلمه یا ترم است)

است منظور از "IDF" معکوس فراوانی سند Term frequency-Inverse document frequency است که وزن اصطلاحاتی را که در مجموعه سند بسیار تکرار می شوند را کاهش دهد و وزن اصطلاحاتی که به ندرت در سند ظاهر می شوند را افزایش دهد. چرا که کلماتی مثل the و امثال آن بسیار پر تکرار اند در یک سند و لی مفهوم خاصی را استخراج نمی کنند بنابراین بهتر است که وزن کمتری داشته باشند. به عبارت دیگر برای بررسی کیفیت یک اصطلاح می آند معکوس فراوانی(وزن) آن اصطلاح را بدست می آورند.

.۸,۶ ساخت مدل و پیش‌بینی برای مقادیر تست

```
1 clf = MultinomialNB().fit(Xtrain, ytrain)
2 y_pred = clf.predict(Xtest)
```

.۸,۷ Confusion matrix

در این آرایه که ما آن را تبدیل به یک دیتا فریم کردیم index ها مربوط به مقادیر واقعی هستند و مقادیر ستون ها مقدار پیش‌بینی شده را نشان می دهد.

```
1 from sklearn import metrics
2 from sklearn.metrics import confusion_matrix
3 from sklearn import preprocessing
4 normalize=preprocessing.normalize(c, norm='l1', axis=1)
5 normalize
6 c=pd.DataFrame(confusion_matrix(test.target, y_pred), index=['comp.graphics', 'sci.space', 'soc.religion.chris'
7 c
```

	Pcomp.graphics	Psci.space	Psoc.religion.christian	Ptalk.religion.misc
comp.graphics	344	13	32	0
sci.space	6	364	24	0
soc.religion.christian	1	5	392	0
talk.religion.misc	4	12	187	48

```
1 d=pd.DataFrame(normalize,index=['comp.graphics', 'sci.space', 'soc.religion.christian', 'talk.religion.misc'],c
2 d
```

	Pcomp.graphics	Psci.space	Psoc.religion.christian	Ptalk.religion.misc
comp.graphics	0.884319	0.033419	0.082262	0.000000
sci.space	0.015228	0.923858	0.060914	0.000000
soc.religion.christian	0.002513	0.012563	0.984925	0.000000
talk.religion.misc	0.015936	0.047809	0.745020	0.191235

Talk.religion میزان خطاش زیاد است.

.۸,۸ پیش‌بینی برای جمله ("she is also campaigning to remove Christmas ("programs.

خط لوله های یادگیری ماشین (ML) از چندین مرحله برای آموزش یک مدل تشکیل شده است. استفاده از پایپ لاین بسیار راحت است و بهترین دقت را بر می گرداند گزینه خوبی بود برای استفاده و ساخت مدل و پیش‌بینی جمله گفته شده است.

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 from sklearn.naive_bayes import MultinomialNB
3 from sklearn.pipeline import make_pipeline
4 import seaborn as sns;
5 md = make_pipeline(TfidfVectorizer(), MultinomialNB())

1 md.fit(train.data, train.target)
2 lb = md.predict(test.data)

1 a=md.predict(['she is also campaigning to remove Christmas programs'])
2 train.target_names[a[0]]

'soc.religion.christian'
```