به نام خدا

دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

گروه مهندسی نرم افزار

Mini Project 2

استاد:

دکتر ناصر قدیری مدرس

نویسنده:

علی کتیرایی

تیرماه ۹۸

سوال1.1

متد DESCR

```python
import sklearn.datasets as datasets
data=datasets.load_breast_cancer()
print(data.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 3 is Mean Radius, field
        13 is Radius SE, field 23 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign
```

:Summary Statistics:

	Min	Max
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252
concave points (worst):	0.0	0.291
symmetry (worst):	0.156	0.664
fractal dimension (worst):	0.055	0.208

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

```
ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/
```

```
.. topic:: References

  - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
    for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
    Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
    San Jose, CA, 1993.
  - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
    prognosis via linear programming. Operations Research, 43(4), pages 570-577,
    July-August 1995.
  - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
    to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)
    163-171.
```

سوال 1.2

متد های keys(): feature_names: data

```python
print(data.keys())
print(data.feature_names)
print(data.data)
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

سوال 1.3

Pandas DataFrame

.

```python
import pandas as pd
df=pd.DataFrame(data.data,columns=data.feature_names)
df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	wors smoothnes
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.162
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.123
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.144
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.209
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.137

5 rows × 30 columns

سوال 1.4

```
df.describe()
```

```
1 df.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst textu
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.00000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	16.269190	25.67722
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	4.833242	6.14625
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	7.930000	12.02000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	13.010000	21.08000
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	14.970000	25.41000
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	18.790000	29.72000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	36.040000	49.54000

8 rows × 30 columns

سوال1.5

کلید Target را برای دیتافریم تعریف می کنیم

```
1 df["target"]=data.get("target")
2 df.set_index('target',inplace=True)
3 df.head()
```

target																
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0
0	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0
0	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0
0	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0
0	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0

5 rows × 30 columns

سوال 1.6

این دو دستور یکی تعداد داده ها را در هر کدام از target های malignant و bengin نشان می دهد و دیگری نام هر کدام از تارگت ها را نمایان می کند.

```
print(df.index.value_counts())
print(data.target_names)
```

```
1    357
0    212
Name: target, dtype: int64
['malignant' 'benign']
```

سوال 1.7

داده های train , test را با Xو Y می سازیم

```
import sklearn.model_selection as model
X=data.data
Y=data.target
x_train,x_test,y_train,y_test=model.train_test_split(X,Y)
```

سوال 1.8

مشاهده می کنیم که داده های train و test با نسبت دلخواه تقسیم شده اند.

```
print("Data Size: " + str(df.shape))
print("X_Train Size: " + str(x_train.shape))
print("X_Test Size: " + str(x_test.shape))
```

```
print("Y_Train Size: " + str(y_train.shape))
print("Y_Test Size: " + str(y_test.shape))
```

```
Data Size: (569, 30)
X_Train Size: (426, 30)
X_Test Size: (143, 30)
Y_Train Size: (426,)
Y_Test Size: (143,)
```

سوال 1.9

```
from sklearn.neighbors import KNeighborsClassifier
neighb=KNeighborsClassifier(n_neighbors=6)
neighb.fit(x_train,y_train)
neighb.score(x_test,y_test)
```

دقت دسته بندی برابر است با :

```
0.9370629370629371
```

سوال 1.10

با استفاده از متد predict داده ها را پیش بینی می کنیم و داخل Y_pred می ریزیم

```
y_pred=neighb.predict(x_test)
```

سوال1.11

توضیح Predict :

در حقیقت این تابع با استفاده از داده های train شده در مدل داده های test را پیش بینی می کند.

سوال 1.12

MinMaxScaler

```
from sklearn.preprocessing import MinMaxScaler
MinMax=MinMaxScaler()
MinMax.fit(X)
norm_x_train=MinMax.transform(x_train)
norm_x_test=MinMax.transform(x_test)
```

سوال 1.13

```
neighb=KNeighborsClassifier(n_neighbors=6)
neighb.fit(norm_x_train,y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=6, p=2,
           weights='uniform')
```

سوال 1.14

```
print("test :" + str(neighb.score(norm_x_test,y_test)))
print("train :" + str(neighb.score(norm_x_train,y_train)))
```

دقت دیتاست های trainو test به شرح زیر می باشد :

```
test :0.9300699300699301
train :0.9812206572769953
```

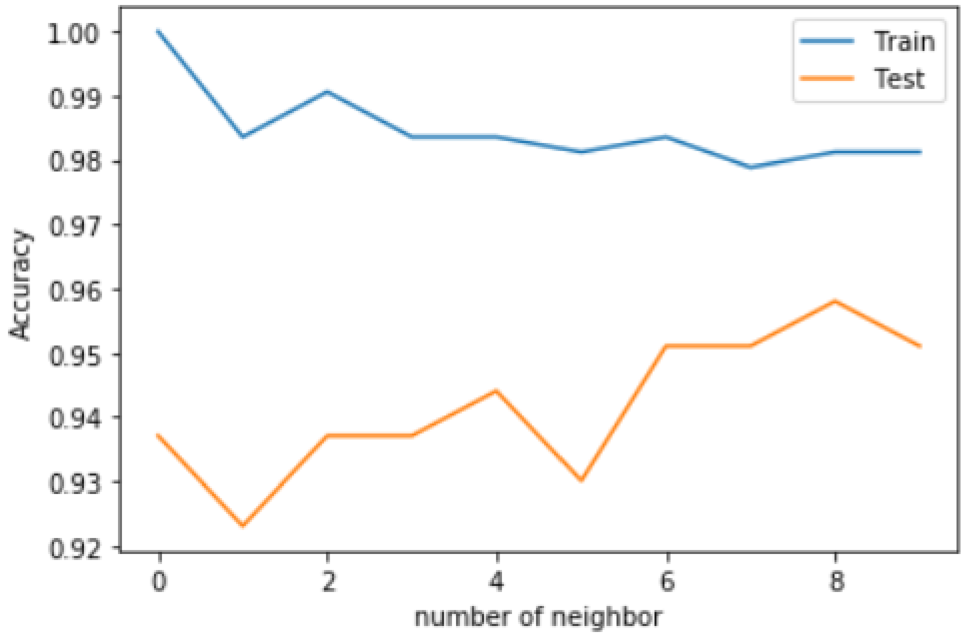
سوال 1.15

```python
train_accuracy=[]
test_accuracy=[]
for i in range(1,11):
    neighb=KNeighborsClassifier(n_neighbors=i)
    neighb.fit(norm_x_train,y_train)
    train_accuracy.insert(i,neighb.score(norm_x_train,y_train))
    test_accuracy.insert(i,neighb.score(norm_x_test,y_test))

print(train_accuracy)
print(test_accuracy)
```

```
[1.0, 0.9835680751173709, 0.9906103286384976, 0.9835680751173709, 0.9835680751173709, 0.9812206572769953, 0.9835680751173709,
0.9788732394366197, 0.9812206572769953, 0.9812206572769953]
[0.9370629370629371, 0.9230769230769231, 0.9370629370629371, 0.9370629370629371, 0.9440559440559441, 0.9300699300699301, 0.951
048951048951, 0.951048951048951, 0.958041958041958, 0.951048951048951]
```

سوال 1.16

```python
import matplotlib.pyplot as plt
plt.plot(train_accuracy,label="Train")
plt.plot(test_accuracy, label="Test")
plt.ylabel('Accuracy')
plt.xlabel('number of neighbor')
plt.legend(loc='upper right')
plt.show()
```



سوال 1.17

همانطور که مشاهده می کنید در داده های train با افزایش تعداد همسایگی دقت کاهش پیدا می کند چرا که مدل دید جامع تری دارد و مجبور است همسایگان بیشتری را در بر بگیرد. ولی در مدل تست تقریبا برعکس است و با افزایش همسایگی دقت مدل افزایش چشم گیری را خواهد داشت. طبیعی است چرا که داده ای که در مدل train نشده است با دیدن همسایگان بیشتر با احتمال بیشتری دقیق پیشبینی می شود ولی داده ای که قبلا در مدل وجود داشته با افزایش همسایگی جوانب را نیز افزایش دادیم به همین جهت دقت کمتری را خواهیم داشت.

سوال 2.1-2.2-

```
import numpy as np
import pandas as pd
df=pd.read_csv('vehicle.csv')
df.head()
```

	COMPACTNESS	CIRCULARITY	DISTANCE_CIRCULARITY	RADIUS_RATIO	PR.AXIS_ASPECT_RATIO	MAX.LENGTH_ASPECT_RATIO	SCATTER_RATIO	ELONG.
0	95	48	83	178	72	10	162	
1	91	41	84	141	57	9	149	
2	104	50	106	209	66	10	207	
3	93	41	82	159	63	9	144	
4	85	44	70	205	103	52	149	

سوال 2.3

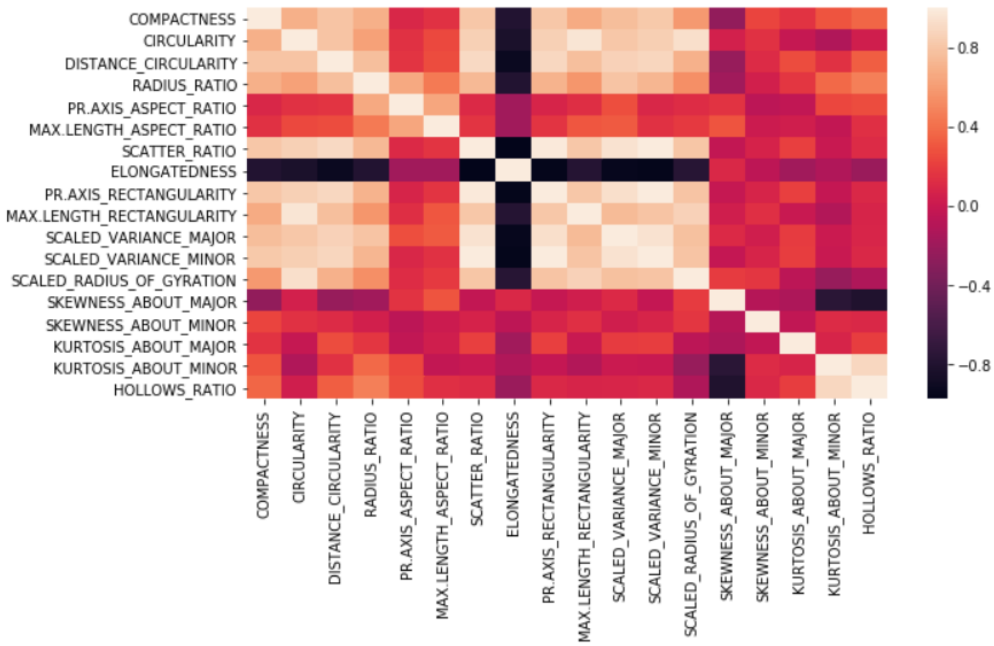
```
print(pd.unique(df["Class"]))
```

```
['van' 'saab' 'bus' 'opel']
```

سوال2.4

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
sns.heatmap(df.corr())
```

<matplotlib.axes._subplots.AxesSubplot at 0x28c347289e8>



سوال 2.5

```
X=df.loc[:,df.columns!="Class"]
Y=df["Class"]
```

سوال 2.6

```
import sklearn.model_selection as model
x_train,x_test,y_train,y_test=model.train_test_split(X,Y,test_size=0.2)
```

سوال 2.7

```
from sklearn.tree import DecisionTreeClassifier
decTree=DecisionTreeClassifier(max_depth = 5, max_features=4,
criterion='entropy')
```

```
decTree.fit(x_train,y_train)
```

.

سوال 2.8

```
import random
from scipy.stats import randint
Params = {"max_depth":[3 , None],
          "max_feature": randint(1,9),
          "min_samples_leaf": randint(1,9)}
print(Params)
```

```
{'max_depth': [3, None], 'max_feature': <scipy.stats._distn_infrastructure.rv_frozen object at 0x000002D0D0438A58>, 'min_sampl
es_leaf': <scipy.stats._distn_infrastructure.rv_frozen object at 0x000002D0D0438B70>}
```

سوال 2.9

```
from sklearn.model_selection import RandomizedSearchCV
tree = DecisionTreeClassifier()

tree_cv = RandomizedSearchCV(tree, Params, cv=5)

tree_cv.fit(X, Y)
```

```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
          estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
          fit_params=None, iid='warn', n_iter=10, n_jobs=None,
          param_distributions={'max_depth': [3, None], 'max_features': <scipy.stats._distn_infrastructure.rv_frozen object at 0
x000001DB90FC6470>, 'min_samples_leaf': <scipy.stats._distn_infrastructure.rv_frozen object at 0x000001DB90FC6588>},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring=None, verbose=0)
```

سوال 2.10

```python
print("Tuned Decision Tree Parameters: {}".format(tree_cv.best_params_))
print("Best score is {}".format(tree_cv.best_score_))
```

```
Tuned Decision Tree Parameters: {'max_depth': None, 'max_features': 4, 'min_samples_leaf': 7}
Best score is 0.6855791962174941
```

سوال 2.11

```python
clf_entropy = DecisionTreeClassifier(criterion = "entropy", max_depth = None, max_features=6, min_samples_leaf=8)
clf_entropy.fit(x_train, y_train)

y_pred = clf_entropy.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy is ", accuracy_score(y_test,y_pred)*100)
```

دقت بدست آمده برابر است با : 62.35294117647059

سوال 2.12

با افزایش مقدار cv دقت مدل نیز افزایش پیدا میکند

سوال 2.13

این تابع جهت مشخص کردن اهمیت feature ها برای انتخاب جهت Decision Node را تعیین می کند.

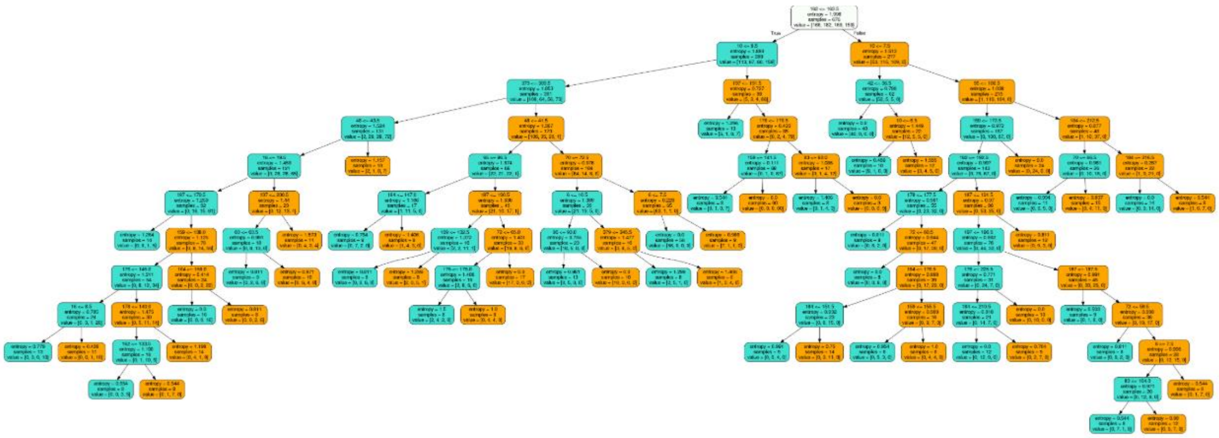
```python
fimportant = dict(zip(df.columns, clf_entropy.feature_importances_))
for key,val in fimportant.items():
    print(key, "=>", val)
```

```
COMPACTNESS => 0.057243604187455936
CIRCULARITY => 0.021646551795873187
DISTANCE_CIRCULARITY => 0.0331075292040589
RADIUS_RATIO => 0.0
PR.AXIS_ASPECT_RATIO => 0.06292942492213029
MAX.LENGTH_ASPECT_RATIO => 0.11923060972148906
SCATTER_RATIO => 0.021075929214674386
ELONGATEDNESS => 0.19069324997836132
PR.AXIS_RECTANGULARITY => 0.0
MAX.LENGTH_RECTANGULARITY => 0.08826263261284722
SCALED_VARIANCE_MAJOR => 0.0
SCALED_VARIANCE_MINOR => 0.19412643509730335
SCALED_RADIUS_OF_GYRATION => 0.014039747171418277
SKEWNESS_ABOUT_MAJOR => 0.07560526810188511
SKEWNESS_ABOUT_MINOR => 0.0019039046049978466
KURTOSIS_ABOUT_MAJOR => 0.026054267154103403
KURTOSIS_ABOUT_MINOR => 0.07847339011861136
HOLLOWS_RATIO => 0.015607456114790403
```

سوال 2.14 و 2.15 و 2.16

```
from sklearn import tree
import pydotplus
import collections
from sklearn.tree import export_graphviz
from IPython.display import Image
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
matplotlib.style.use('ggplot')

dot_data = tree.export_graphviz(clf_entropy,
                                feature_names=df.iloc[0,0:-1],
                                out_file=None,
                                filled=True,
                                rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)

colors = ('turquoise', 'orange')
edges = collections.defaultdict(list)

for edge in graph.get_edge_list():
    edges[edge.get_source()].append(int(edge.get_destination()))

for edge in edges:
    edges[edge].sort()
    for i in range(2):
        dest = graph.get_node(str(edges[edge][i]))[0]
        dest.set_fillcolor(colors[i])

```

```
graph.write_png('dot_data.png')
Image('dot_data.png')
```



سوال 3.1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
iris=datasets.load_iris()
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

	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

سوال 3.2

```
from sklearn.cluster import KMeans
```

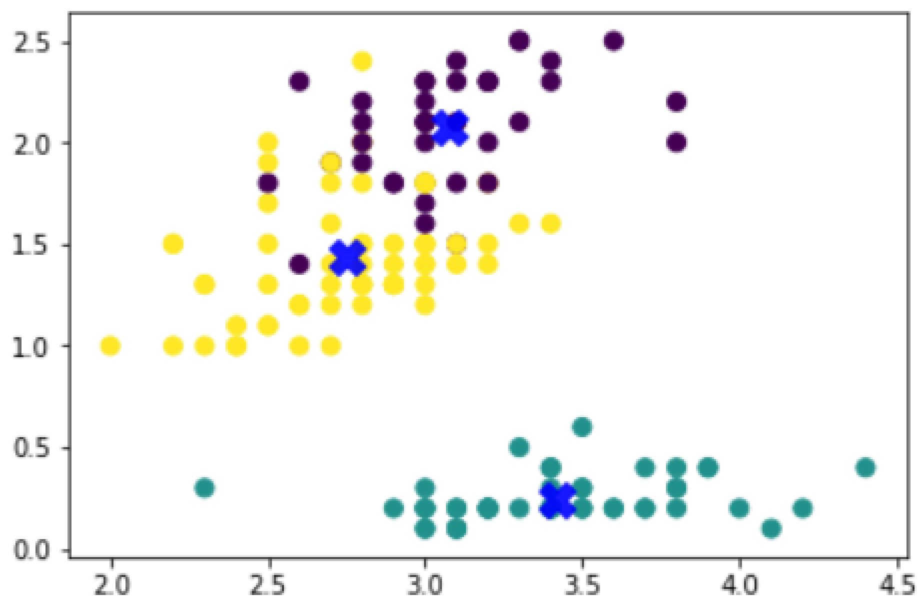
```
X = iris.data
y = iris.target
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

سوال 3.3

```
centroids = kmeans.cluster_centers_
```

سوال 3.4

```
plt.scatter(X[:, 1], X[:, 3], c=y_kmeans, s=50, cmap='viridis')
plt.scatter(centroids[:, 1], centroids[:, 3], c='blue',marker="X", s=200,
alpha=0.9);
```



سوال 3.5

```
kmeans.inertia_
```

78.94084142614602

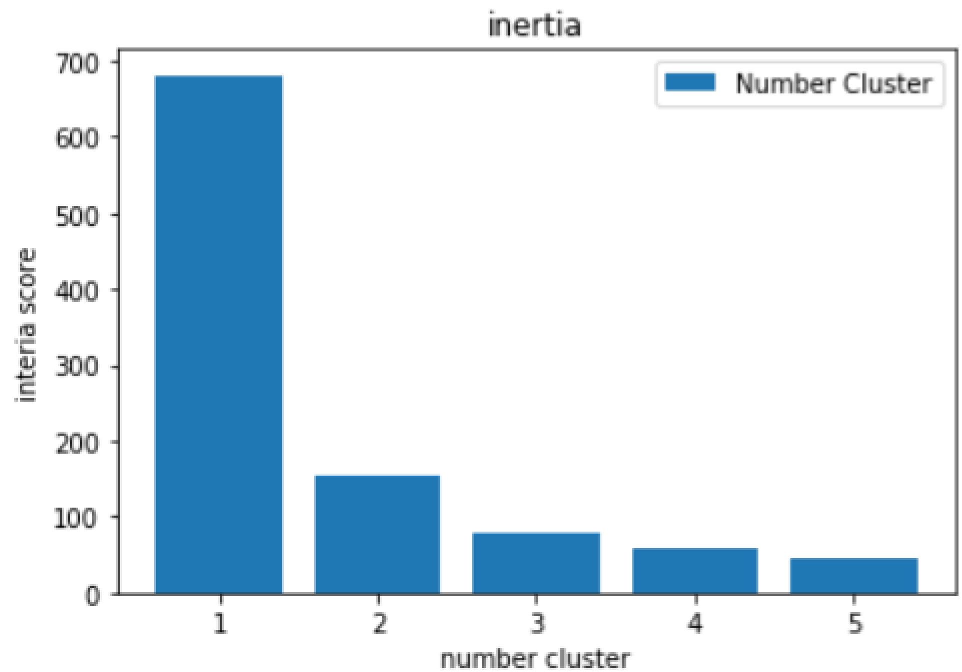
سوال 3.6

```
interia_list=[]
for i in range(5):
    kmeans = KMeans(n_clusters=i+1)
    kmeans.fit(X)
    y_kmeans = kmeans.predict(X)
    interia_list.append(kmeans.inertia_)

interia_list
```

**[680.8244,
152.36870647733906,
78.94084142614602,
57.31787321428571,
46.53558205128205]**

سوال 3.7

```
plt.bar(list(range(1,6)),interia_list, label="Number Cluster")
plt.legend()
plt.xlabel('number cluster')
plt.ylabel('interia score')
plt.title('inertia')
plt.show()
```



با افزایش تعداد کلاستر مقدار اینرسی نیز کاهش می یابد چرا که مقادیر درون هر کلاستر منسجم تر می شود تا جایی که هر کلاستر خود یک نود شود. جهت فهمیدن تعداد کلاستر بهینه از دو روش معروف می توان استفاده کرد:

1. Elbow : در این روش با استفاده از معیار فاصله درون خوشه ای اقدام به پیدا کردن کمترین فاصله درون خوشه ای به ازای k می کنیم.
2. میانگین شاخص silhouette: در این روش نیز مانند قبل یک بازه ای از K ها را تست کرده و برای هر k یک میانگین silhouette بدست می آوریم و با رسم نمودار curve آنها بالا ترین یا بیشترین مقدار همان بهینه ترین است

سوال 4.1

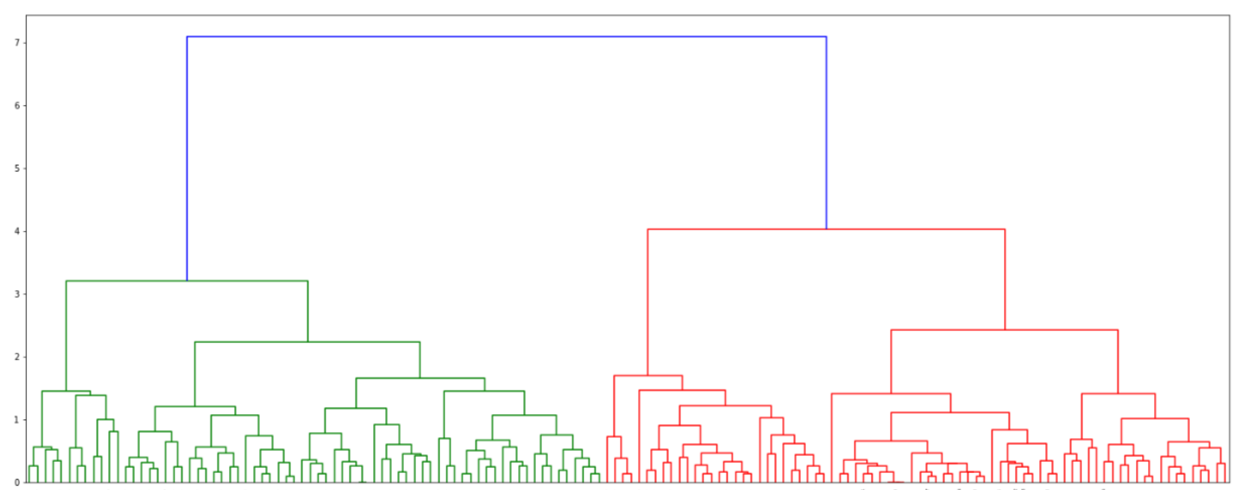
```
import numpy as np
import pandas as pd
from sklearn import datasets
iris=datasets.load_iris()
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

```python
from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib import pyplot as plt
X = df
Z = linkage(X, 'complete')
```

	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

سوال 4.2

```python
fig = plt.figure(figsize=(25, 10))
dn = dendrogram(Z)
```



سوال 4.3

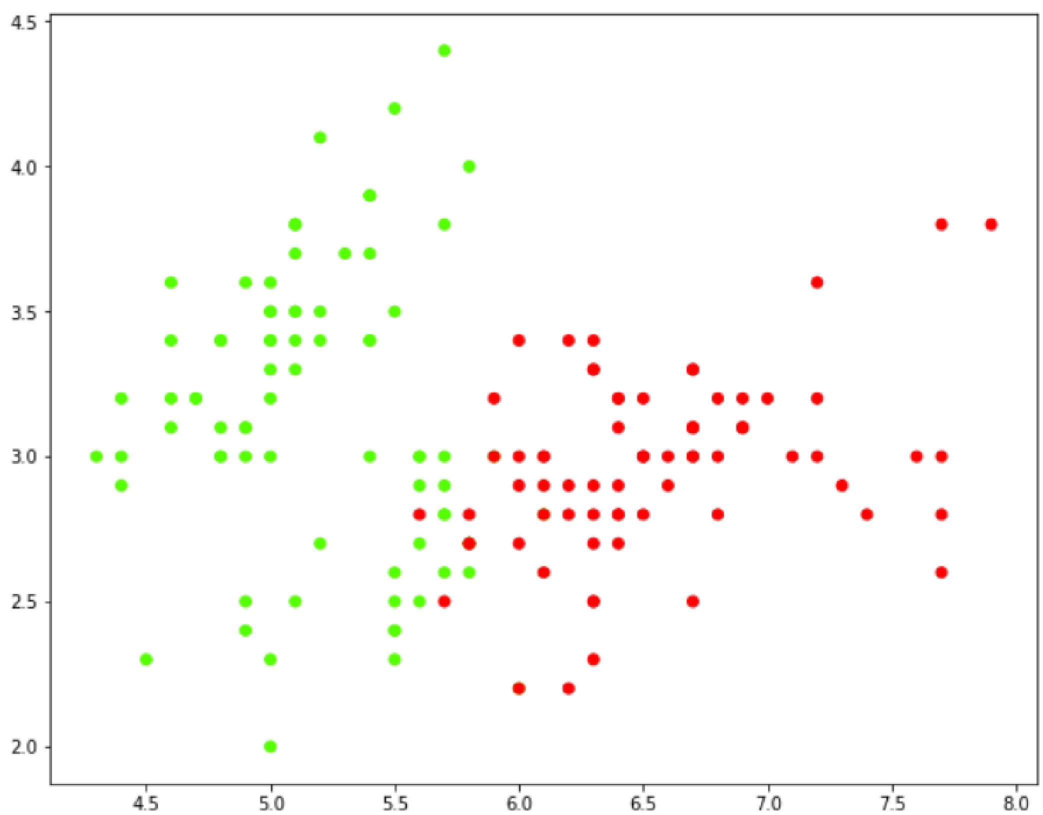
```python
from scipy.cluster.hierarchy import fcluster
```

```
fcluster(Z, t=6, criterion='distance')
```

```
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, 2, 2, 2, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 2, 1, 2, 1,
       2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 2, 1, 1, 1,
       2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 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], dtype=int32)
```

سوال 4.4

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 8))
plt.scatter(iris.data[:,0], iris.data[:,1], c=clusters, cmap='prism')
plt.show()
```



سوال 5.1

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
boston=datasets.load_boston()
df=pd.DataFrame(boston.data,columns=boston.feature_names)
df.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

سوال 5.2

```python
df["Price"]=boston.target
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

سوال 5.3

```python
X=df[["CRIM","ZN"]]
Y=df["Price"]
```

سوال 5.4

```python
import sklearn.model_selection as model
x_train,x_test,y_train,y_test=model.train_test_split(X,Y,test_size=0.3)
```

سوال 5.5

```python
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(x_train, y_train)
y_pred = lm.predict(x_test)
plt.scatter(y_test, y_pred)
plt.xlabel("Prices:")
plt.ylabel("Predicted prices")
plt.title("Prices vs Predicted prices")
```

```
Text(0.5,1,'Prices vs Predicted prices')
```

سوال 5.6

```python
from sklearn.metrics import mean_squared_error
print("MSE : " + str(mean_squared_error(y_test,y_pred)))
```

```
MSE : 65.20918707439772
```

سوال 5.7

```
X=df.loc[:,df.columns != "Price"]
Y=df["Price"]

x_train,x_test,y_train,y_test=model.train_test_split(X,Y,test_size=0.3)


lm = LinearRegression()
lm.fit(x_train, y_train)

y_pred = lm.predict(x_test)

plt.scatter(y_test, y_pred)
plt.xlabel("Prices:")
plt.ylabel("Predicted prices")
plt.title("Prices vs Predicted prices")ed)))
```

```
Text(0.5,1,'Prices vs Predicted prices')
```

سوال 5.8

```
print("MSE : " + str(mean_squared_error(y_test,y_pred)))
```

```
MSE : 20.381895735190685
```

مقدار MSE کاهش چشم گیری را داشته و طبیعی است که با افزایش تعدادfeature ها دقت افزایش و خطا نیز کاهش می یابد. چرا که جوانب بیشتری را در مدل کردن مد نظر دارد.

سوال 5.9

```
from sklearn.model_selection import cross_val_score
import statistics
cvs=cross_val_score(lm, X, Y, cv=5)
print(cvs)
print("Mean cvs: "+ str(statistics.mean(cvs)))
```

```
[ 0.63861069  0.71334432  0.58645134  0.07842495 -0.26312455]
Mean cvs: 0.3507413509325258
```

سوال 6.1

```
import numpy as np
import pandas as pd
import sklearn.datasets as datasets
bcancer=datasets.load_breast_cancer()
df=pd.DataFrame(bcancer.data,columns=bcancer.feature_names)
df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	wors smoothnes
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.162
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.123
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.144
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.209
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.137

5 rows × 30 columns

سوال 6.2

```python
import sklearn.model_selection as model
X=bcancer.data
Y=bcancer.target
x_train,x_test,y_train,y_test=model.train_test_split(X,Y,test_size=0.2)

from sklearn.neighbors import KNeighborsClassifier
neighb=KNeighborsClassifier(n_neighbors=8)
neighb.fit(x_train,y_train)
y_pred=neighb.predict(x_test)
```

سوال 6.3

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

سوال 6.4

```
print(confusion_matrix(y_test,y_pred))
```

```
[[41  3]
 [ 2 68]]
```

این چهار عدد نشان دهنده 4 حالت پیش بینی ما و درستی جواب می باشد که شامل: TN , TP , FN , FP

سوال 6.5

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        44
           1       0.96      0.97      0.96        70

   micro avg       0.96      0.96      0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

میزان دقت در دو دسته بندی را نشان می دهد که هر کدام از این 0 و 1 ها نمایان گر یک دسته می باشند.

سوال 6.6

```
from sklearn.preprocessing import normalize
normalize(confusion_matrix(y_test,y_pred),norm="l1")
```

```
array([[0.93181818, 0.06818182],
       [0.02857143, 0.97142857]])
```

سوال 6.7

```
dff = pd.DataFrame(Normal,
index=['benign','malignant'],columns=['benign','malignant'])
print (dff)
```

```
           benign  malignant
benign     0.958333   0.041667
malignant  0.106061   0.893939
```

سوال 6.8

هر چه در این نمونه از نمودار ها به محور x=y نزدیک تر باشد خطای بیشتری دارد به همین منظور می توان گفت هر چه حالت elbow داشته باشت خطای آن کم تر و در نتیجه جواب ما ایده آل تر خواهد بود

سوال 6.9

```
y_pred_prob=neighb.predict_proba(x_test)
print(y_pred_prob)
```

```
[[1.    0.   ]
 [1.    0.   ]
 [0.875 0.125]
 [0.    1.   ]
 [0.125 0.875]
 [0.5   0.5  ]
 [0.    1.   ]
 [0.    1.   ]
 [1.    0.   ]
 [0.    1.   ]
 [0.    1.   ]
 [0.625 0.375]
 [0.    1.   ]
 [0.875 0.125]
 [0.    1.   ]
 [1.    0.   ]
 [0.    1.   ]
 [1.    0.   ]
 [1.    0.   ]
 [0.25  0.75 ]
```

سوال 6.10

```python
from sklearn.metrics import roc_curve
roc_curve(y_pred,y_pred_prob)
```

مشکل در ورودی تابع داشتیم و نتونستیم شکل ROC را رسم کنیم ☹

سوال 7.1

```python
import pandas as pd
```

```python
import numpy as np
df=pd.read_excel("OnlineRetail.xlsx")
df.head()
```

	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

سوال 7.2

```python
! pip install mlxtend --upgrade --no-deps
```

سوال 7.3

```python
from mlxtend.frequent_patterns import apriori,association_rules
```

سوال 7.4

```python
df["Description"]=df["Description"].str.strip()
```

سوال 7.5

```python
print("Orginal Size : " + str(df.size))
df["InvoiceNo"].replace('', np.nan, inplace=True)
df.dropna(subset=['InvoiceNo'], inplace=True)
print("Reduced Size : " + str(df.size))

df["InvoiceNo"]=df["InvoiceNo"].astype("str")
```

```
Orginal Size : 4335272
Reduced Size : 4335272
```

سوال 7.6

```python
df=df[~df.InvoiceNo.str.contains("C")]
```

سوال 7.7

```python
basket = (df[df['Country'] =="France"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0).set_index('InvoiceNo'))
basket.head()
```

Description InvoiceNo	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 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	YELL(CO RA PAF FASHI
536370	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	
536974	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	
537463	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

5 rows × 1563 columns

سوال 7.8

```
basket=basket.applymap(lambda x: 1 if x > 0 else 0)
basket.head()
```

Description InvoiceNo	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 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	YELL(CO RA PAF FASHI
536370	0	0	0	0	0	0	0	0	0	0	...	0	
536852	0	0	0	0	0	0	0	0	0	0	...	0	
536974	0	0	0	0	0	0	0	0	0	0	...	0	
537065	0	0	0	0	0	0	0	0	0	0	...	0	
537463	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 1563 columns

سوال 7.9

```
basket=basket.drop("POSTAGE",axis=1)
```

سوال 7.10

```
frequent_itemsets=apriori(basket,min_support=0.07)
frequent_itemsets.head()
```

	support	itemsets
0	0.071429	(35)
1	0.096939	(61)
2	0.102041	(64)
3	0.094388	(65)
4	0.081633	(103)

سوال 7.11

```
rules = association_rules(frequent_itemsets, metric="lift",
min_threshold=1)
rules.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(64)	(61)	0.102041	0.096939	0.073980	0.725000	7.478947	0.064088	3.283859
1	(61)	(64)	0.096939	0.102041	0.073980	0.763158	7.478947	0.064088	3.791383
2	(65)	(61)	0.094388	0.096939	0.079082	0.837838	8.642959	0.069932	5.568878
3	(61)	(65)	0.096939	0.094388	0.079082	0.815789	8.642959	0.069932	4.916181
4	(64)	(65)	0.102041	0.094388	0.073980	0.725000	7.681081	0.064348	3.293135

سوال 7.12

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

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
2	(65)	(61)	0.094388	0.096939	0.079082	0.837838	8.642959	0.069932	5.568878
3	(61)	(65)	0.096939	0.094388	0.079082	0.815789	8.642959	0.069932	4.916181
16	(1267)	(1247)	0.127551	0.132653	0.102041	0.800000	6.030769	0.085121	4.336735
18	(1266)	(1267)	0.137755	0.127551	0.122449	0.888889	6.968889	0.104878	7.852041
19	(1267)	(1266)	0.127551	0.137755	0.122449	0.960000	6.968889	0.104878	21.556122
20	(1266, 1267)	(1247)	0.122449	0.132653	0.099490	0.812500	6.125000	0.083247	4.625850
21	(1266, 1247)	(1267)	0.102041	0.127551	0.099490	0.975000	7.644000	0.086474	34.897959
22	(1267, 1247)	(1266)	0.102041	0.137755	0.099490	0.975000	7.077778	0.085433	34.489796

سوال 7.13

به عنوان مثال آیتم 65 را اگر موجود باشد با دقت 0.83 امکان دارد آیتم 61 نیز باشد. یا به صورت دیگر اگر آیتم ها را مربوط به سوپرمارکت بدانیم می توان گفت اگر آیتم 1266 و 1267 با هم انتخاب شوند با confidence 0.97 می توان گفت آیتم 1267 نیز خریداری می شود.