:[28] In

```
import numpy as np
iris_data = np.genfromtxt('F:\\mydataset.data',
delimiter=",", skip_header=1)
print(iris_data)
```

```
[nan 0.4 1.7 3.9 5.4]
[nan 0.3 1.4 3.4 4.6]
[nan 0.2 1.5 3.4  .5]
[nan 0.2 1.4 2.9 4.4]
[nan 0.1 1.5 3.1 4.9]
[nan 0.2 1.5 3.7 5.4]
[nan 0.2 1.6 3.4 4.8]
[nan 0.1 1.4  .3 4.8]
[nan 0.1 1.1  .3 4.3]
[nan 0.2 1.2  .4 5.8]
[nan 0.4 1.5 4.4 5.7]
[nan 0.4 1.3 3.9 5.4]
[nan 0.3 1.4 3.5 5.1]
[nan 0.3 1.7 3.8 5.7]
[nan 0.3 1.5 3.8 5.1]
[nan 0.2 1.7 3.4 5.4]
[nan 0.4 1.5 3.7 5.1]
[nan 0.2  .1 3.6 4.6]
[nan 0.5 1.7 3.3 5.1]
[nan 0.2 1.9 3.4 4.8]
```

:[30] In

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(color_codes=True)
```

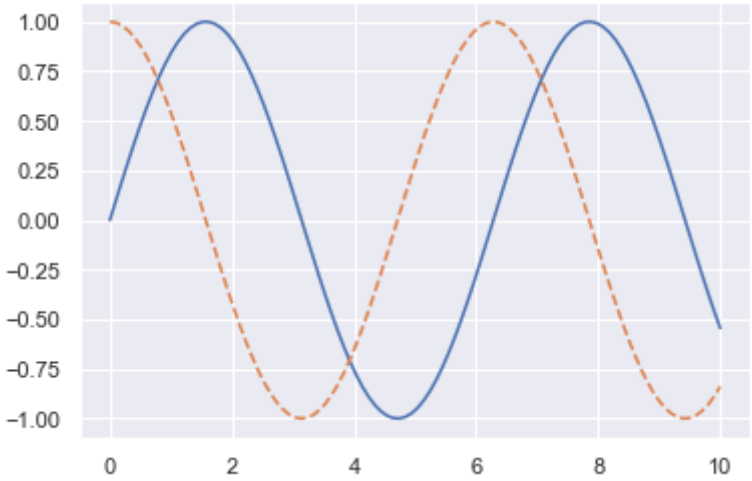
In [34]:

```
x = np.linspace(0, 10, 100)

fig = plt.figure()
plt.plot(x, np.sin(x), '-')
plt.plot(x, np.cos(x), '--')
```

Out[34]:

```
[<matplotlib.lines.Line2D at 0x28660395fc8>]
```



In [43]:

```
from sklearn import datasets
iris = datasets.load_iris()
print(iris.filename)
df=pd.read_csv(mydataset.filename)
df.head(5)
```

```
E:\Users\HP\Anaconda3\lib\site-packages\sklearn\datasets\data\iris.csv
```

Out[43]:

virginica	versicolor	setosa	4	150	
0	0.2	1.4	3.5	5.1	0
0	0.2	1.4	3.0	4.9	1
0	0.2	1.3	3.2	4.7	2
0	0.2	1.5	3.1	4.6	3
0	0.2	1.4	3.6	5.0	4

:[41] In

```
df=pd.read_csv(mydataset.filename).info()
```

```
<'class 'pandas.core.frame.DataFrame>
RangeIndex: 150 entries, 0 to 149
:(Data columns (total 5 columns
non-null float64 150           150
non-null float64 150             4
setosa        150 non-null float64
versicolor    150 non-null float64
virginica     150 non-null int64
(dtypes: float64(4), int64(1
memory usage: 6.0 KB
```

:[44] In

```
df.describe()
```

Out[44]:

	virginica	versicolor	setosa	4	150
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	1.000000	1.199333	3.758000	3.057333	5.843333
std	0.819232	0.762238	1.765298	0.435866	0.828066
min	0.000000	0.100000	1.000000	2.000000	4.300000
25%	0.000000	0.300000	1.600000	2.800000	5.100000
50%	1.000000	1.300000	4.350000	3.000000	5.800000
75%	2.000000	1.800000	5.100000	3.300000	6.400000
max	2.000000	2.500000	6.900000	4.400000	7.900000

:[47] In

```
import pandas as pd

df = pd.read_csv('F:\\mydataset.data')
df.describe()
```

Out[47]:

	0.2	1.4	3.5	5.1	
149.000000	149.000000	149.000000	149.000000	count
1.205369	3.774497	3.051007	5.848322	mean
0.761292	1.759651	0.433499	0.828594	std
0.100000	1.000000	2.000000	4.300000	min
0.300000	1.600000	2.800000	5.100000	25%
1.300000	4.400000	3.000000	5.800000	50%
1.800000	5.100000	3.300000	6.400000	75%
2.500000	6.900000	4.400000	7.900000	max

:[49] In

```
print(df.describe())
```

```
              0.2         1.4         3.5         5.1
count  149.000000  149.000000  149.000000  149.000000
mean     5.848322    3.051007    3.774497    1.205369
std      0.828594    0.433499    1.759651    0.761292
min      4.300000    2.000000    1.000000    0.100000
0.300000    1.600000    2.800000    5.100000      25%
1.300000    4.400000    3.000000    5.800000      50%
1.800000    5.100000    3.300000    6.400000      75%
max      7.900000    4.400000    6.900000    2.500000
```

:[50] In

```
import pandas as pd
from sklearn import datasets
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(color_codes=True)
```

:[54] In

```
dataset=mydataset.data()

data=pd.DataFrame(dataset['data'], columns=['petal length','petal width','sepal length','se
data['species']=dataset['target']
data['species']=data['species'].apply(lambda x: dataset['target_names'][x])
```

```
---------------------------------------------------------------------------
(TypeError                                 Traceback (most recent call last
<ipython-input-54-6ebe9b8539fe> in <module>
()dataset=mydataset.data 1 <----
 2
data=pd.DataFrame(dataset['data'], columns=['petal length','petal wi 3
(['dth','sepal length','sepal width
['data['species']=dataset['target 4
data['species']=data['species'].apply(lambda x: dataset['target_name 5
([s'][x

TypeError: 'numpy.ndarray' object is not callable
```

:[52] In

```
data.head(10)
```

Out[52]:

species	sepal width	sepal length	petal width	petal length	
setosa	0.2	1.4	3.5	5.1	0
setosa	0.2	1.4	3.0	4.9	1
setosa	0.2	1.3	3.2	4.7	2
setosa	0.2	1.5	3.1	4.6	3
setosa	0.2	1.4	3.6	5.0	4
setosa	0.4	1.7	3.9	5.4	5
setosa	0.3	1.4	3.4	4.6	6
setosa	0.2	1.5	3.4	5.0	7
setosa	0.2	1.4	2.9	4.4	8
setosa	0.1	1.5	3.1	4.9	9

:[55] In

```
data.isnull().sum() #there is no missing value.
```

Out[55]:

```
petal length    0
petal width     0
sepal length    0
sepal width     0
species         0
dtype: int64
```

:[56] In

```
modData = data.append({'petal length' : np.nan , 'petal width' : 3.6, 'sepal length' : 0,
                       'sepal width': 0.2, 'species' : 'setosa'} ,ignore_index=True)
modData.describe()
```

Out[56]:

	sepal width	sepal length	petal width	petal length
count	151.000000	151.000000	151.000000	150.000000
mean	1.192715	3.733113	3.060927	5.843333
std	0.764033	1.785785	0.436650	0.828066
min	0.100000	0.000000	2.000000	4.300000
25%	0.300000	1.550000	2.800000	5.100000
50%	1.300000	4.300000	3.000000	5.800000
75%	1.800000	5.100000	3.350000	6.400000
max	2.500000	6.900000	4.400000	7.900000

:[57] In

```
print('Columns with missing values')
print(modData.isnull().sum())
print('\n Columns with zero values')
print((modData[['petal length','petal width','sepal length','sepal width','species']]==0).s
```

```
Columns with missing values
petal length    1
petal width     0
sepal length    0
sepal width     0
species         0
dtype: int64

Columns with zero values
petal length    0
petal width     0
sepal length    1
sepal width     0
species         0
dtype: int64
```

:[58] In

```
modData.fillna(modData.mean(), inplace=True)
print(modData.isnull().sum())
```

```
petal length    0
petal width     0
sepal length    0
sepal width     0
species         0
dtype: int64
```

:[62] In

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
PCA_df = pd.read_csv(url, names=['sepal length','sepal width','petal length','petal width',
```

:[63] In

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
features = ['sepal length', 'sepal width', 'petal length', 'petal width']
# Separating out the features
x = PCA_df.loc[:, features].values
# Separating out the target
y = PCA_df.loc[:,['target']].values
# Standardizing the features
x = StandardScaler().fit_transform(x)
print (x)
```

```
[9.00681170e-01  1.03205722e+00 -1.34127240e+00 -1.31297673e+00-]]
[1.14301691e+00 -1.24957601e-01 -1.34127240e+00 -1.31297673e+00-]
[1.38535265e+00  3.37848329e-01 -1.39813811e+00 -1.31297673e+00-]
[1.50652052e+00  1.06445364e-01 -1.28440670e+00 -1.31297673e+00-]
[1.02184904e+00  1.26346019e+00 -1.34127240e+00 -1.31297673e+00-]
[5.37177559e-01  1.95766909e+00 -1.17067529e+00 -1.05003079e+00-]
[1.50652052e+00  8.00654259e-01 -1.34127240e+00 -1.18150376e+00-]
[1.02184904e+00  8.00654259e-01 -1.28440670e+00 -1.31297673e+00-]
[1.74885626e+00 -3.56360566e-01 -1.34127240e+00 -1.31297673e+00-]
[1.14301691e+00  1.06445364e-01 -1.28440670e+00 -1.44444970e+00-]
[5.37177559e-01  1.49486315e+00 -1.28440670e+00 -1.31297673e+00-]
[1.26418478e+00  8.00654259e-01 -1.22754100e+00 -1.31297673e+00-]
[1.26418478e+00 -1.24957601e-01 -1.34127240e+00 -1.44444970e+00-]
[1.87002413e+00 -1.24957601e-01 -1.51186952e+00 -1.44444970e+00-]
[5.25060772e-02  2.18907205e+00 -1.45500381e+00 -1.31297673e+00-]
[1.73673948e-01  3.11468391e+00 -1.28440670e+00 -1.05003079e+00-]
[5.37177559e-01  1.95766909e+00 -1.39813811e+00 -1.05003079e+00-]
[9.00681170e-01  1.03205722e+00 -1.34127240e+00 -1.18150376e+00-]
[1.73673948e-01  1.72626612e+00 -1.17067529e+00 -1.18150376e+00-]
```

:[64] In

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
                           , columns = ['principal component 1', 'principal component 2'])
```

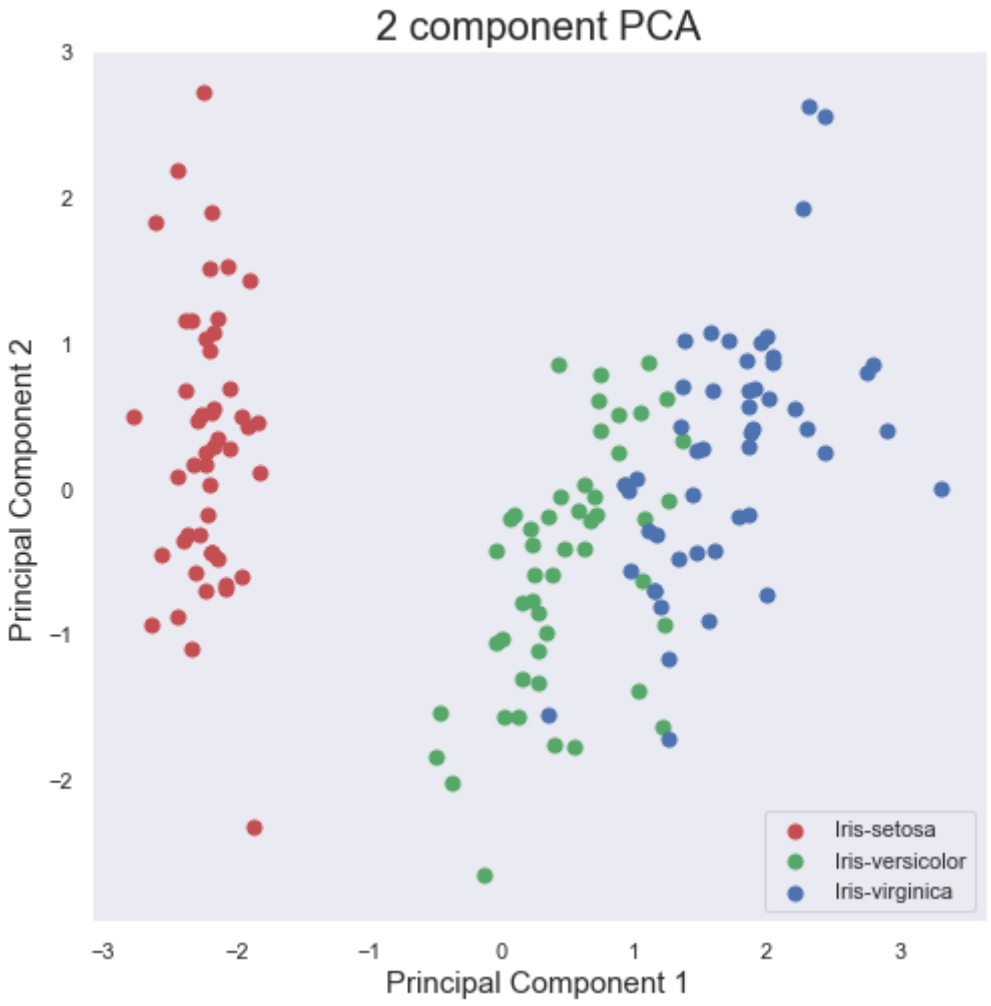
:[65] In

```
finalDf = pd.concat([principalDf, PCA_df[['target']]], axis = 1)
```

:[66] In

```
import matplotlib.pyplot as plt
%matplotlib inline
#sns.set(color_codes=True)

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```



:[67] In

```
print(pca.explained_variance_ratio_)
```

```
[0.23030523 0.72770452]
```

:[68] In

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
```

:[69] In

```
from sklearn import datasets
iris= datasets.load_iris()
df= pd.read_csv(iris.filename)
```

:[70] In

```
x = df.iloc[:, [0,1,2,3]].values
```

:[71] In

```
kmeans5 = KMeans(n_clusters=3)
y_kmeans5 = kmeans5.fit_predict(x)
print(y_kmeans5)
kmeans5.cluster_centers_
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 2 1 1 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2
1 2 1 1 1 2 1 1 1 2 1 1 1 1 2 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 2 1 1 1 1 2 2 1 1
[2 1
```

Out[71]:

```
,[      array([[5.006      , 3.428      , 1.462      , 0.246
,[2.07105263 ,5.74210526 ,3.07368421 ,       6.85]
([[1.43387097 ,4.39354839 , 2.7483871 , 5.9016129]
```

:[72] In

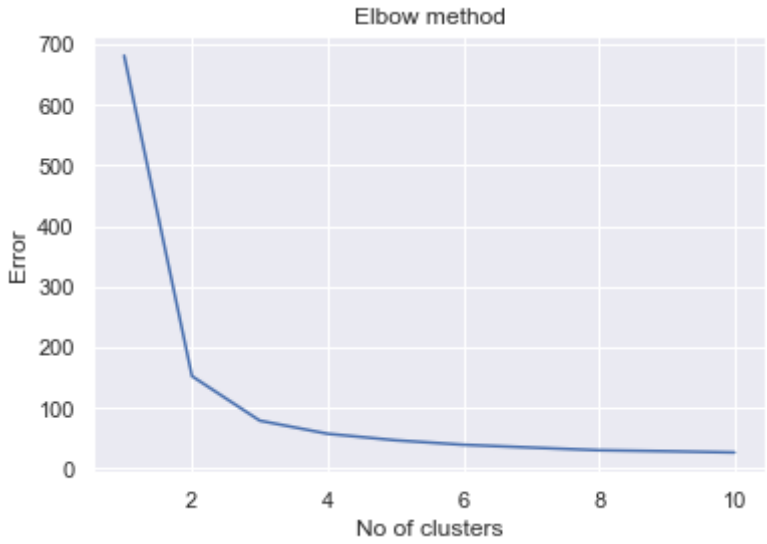
```
kmeans5.cluster_centers_
```

Out[72]:

```
,[      array([[5.006      , 3.428      , 1.462      , 0.246
,[2.07105263 ,5.74210526 ,3.07368421 ,       6.85]
([[1.43387097 ,4.39354839 , 2.7483871 , 5.9016129]
```

:[73] In

```
Error =[]
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i).fit(x)
    kmeans.fit(x)
    Error.append(kmeans.inertia_)
plt.plot(range(1, 11), Error)
plt.title('Elbow method')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show()
```

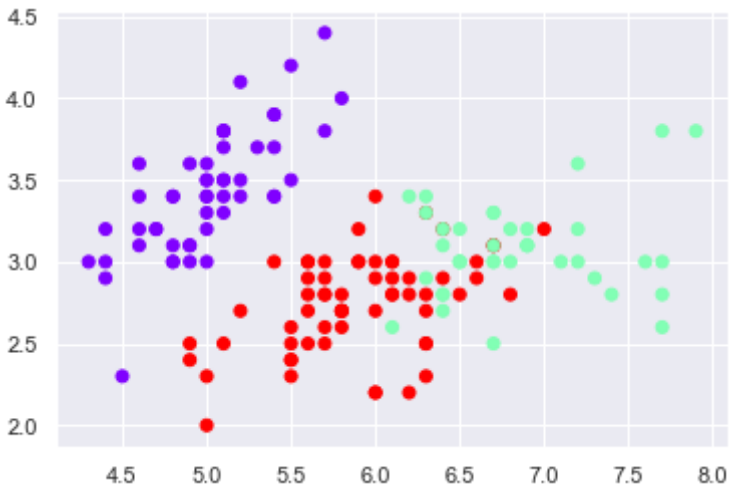


:[74] In

```
plt.scatter(x[:,0], x[:,1], c=y_kmeans5, cmap='rainbow')
```

Out[74]:

```
<matplotlib.collections.PathCollection at 0x28660f9f088>
```



:[75] In

```
pip install mlxtend
```

```
Requirement already satisfied: mlxtend in e:\users\hp\anaconda3\lib\site-pac
(kages (0.17.2
Requirement already satisfied: pandas>=0.24.2 in e:\users\hp\anaconda3\lib\s
(ite-packages (from mlxtend) (0.25.1
Requirement already satisfied: matplotlib>=3.0.0 in e:\users\hp\anaconda3\li
(b\site-packages (from mlxtend) (3.1.1
Requirement already satisfied: joblib>=0.13.2 in e:\users\hp\anaconda3\lib\s
(ite-packages (from mlxtend) (0.13.2
Requirement already satisfied: numpy>=1.16.2 in e:\users\hp\anaconda3\lib\si
(te-packages (from mlxtend) (1.16.5
Requirement already satisfied: setuptools in e:\users\hp\anaconda3\lib\site-
(packages (from mlxtend) (41.4.0
Requirement already satisfied: scikit-learn>=0.20.3 in e:\users\hp\anaconda3
(\lib\site-packages (from mlxtend) (0.21.3
Requirement already satisfied: scipy>=1.2.1 in e:\users\hp\anaconda3\lib\sit
(e-packages (from mlxtend) (1.3.1
Requirement already satisfied: python-dateutil>=2.6.1 in e:\users\hp\anacond
(a3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2.8.0
Requirement already satisfied: pytz>=2017.2 in e:\users\hp\anaconda3\lib\sit
(e-packages (from pandas>=0.24.2->mlxtend) (2019.3
Requirement already satisfied: cycler>=0.10 in e:\users\hp\anaconda3\lib\sit
(e-packages (from matplotlib>=3.0.0->mlxtend) (0.10.0
Requirement already satisfied: kiwisolver>=1.0.1 in e:\users\hp\anaconda3\li
(b\site-packages (from matplotlib>=3.0.0->mlxtend) (1.1.0
 Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
 e:\users\hp\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
((2.4.2
Requirement already satisfied: six>=1.5 in e:\users\hp\anaconda3\lib\site-pa
(ckages (from python-dateutil>=2.6.1->pandas>=0.24.2->mlxtend) (1.12.0
.Note: you may need to restart the kernel to use updated packages
```

:[76] In

```
from mlxtend.frequent_patterns import apriori
```

:[77] In

```
dataset
```

Out[77]:

```
,[data': array([[5.1, 3.5, 1.4, 0.2'}
,[0.2 ,1.4 , .3 ,4.9]
,[0.2 ,1.3 ,3.2 ,4.7]
,[0.2 ,1.5 ,3.1 ,4.6]
,[0.2 ,1.4 ,3.6 , .5]
,[0.4 ,1.7 ,3.9 ,5.4]
,[0.3 ,1.4 ,3.4 ,4.6]
,[0.2 ,1.5 ,3.4 , .5]
,[0.2 ,1.4 ,2.9 ,4.4]
,[0.1 ,1.5 ,3.1 ,4.9]
,[0.2 ,1.5 ,3.7 ,5.4]
,[0.2 ,1.6 ,3.4 ,4.8]
,[0.1 ,1.4 , .3 ,4.8]
,[0.1 ,1.1 , .3 ,4.3]
,[0.2 ,1.2 , .4 ,5.8]
,[0.4 ,1.5 ,4.4 ,5.7]
,[0.4 ,1.3 ,3.9 ,5.4]
```

:[78] In

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

TranEncod = TransactionEncoder()
te_ary = TranEncod.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=TranEncod.columns_)
```

:[79] In

```
df
```

Out[79]:

m	l	i	g	f	e	d	a	_	S	R	E	D	C	
False	False	False	False	False	False	True	True	False	False	False	False	False	False	0
False	False	False	True	False	True	False	True	False	False	False	False	False	False	1
True	False	False	True	False	True	False	True	True	False	False	False	False	False	2
False	False	False	False	False	False	False	False	False	True	True	True	True	True	3
True	False	False	False	True	True	False	True	True	False	False	False	False	False	4
True	True	True	False	True	True	False	True	False	False	False	False	False	False	5

In [80]:

```
apriori(df, min_support=0.6)
```

Out[80]:

	itemsets	support
0	(6)	0.833333
1	(8)	0.666667
2	(17)	0.666667
3	(6 ,8)	0.666667
4	(6 ,17)	0.666667

In [81]:

```
apriori(df, min_support=0.6, use_colnames=True)
```

Out[81]:

	itemsets	support
0	(a)	0.833333
1	(e)	0.666667
2	(t)	0.666667
3	(a, e)	0.666667
4	(a, t)	0.666667

:[82] In

```
apriori(df, min_support=0.3, use_colnames=True)
```

Out[82]:

	itemsets	support
0	(_)	0.333333
1	(a)	0.833333
2	(e)	0.666667
3	(f)	0.333333
4	(g)	0.333333
...	...	...
282	(a, r, t, e, s, n ,_)	0.333333
283	(m, a, r, t, s, n ,_)	0.333333
284	(m, r, t, e, s, n ,_)	0.333333
285	(m, a, r, t, e, s, n)	0.333333
286	(m, a, r, t, e, s, n ,_)	0.333333

rows × 2 columns 287

In [83]:

```
frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

Out[83]:

	length	itemsets	support
0	1	(a)	0.833333
1	1	(e)	0.666667
2	1	(m)	0.500000
3	1	(n)	0.500000
4	1	(r)	0.500000
5	1	(t)	0.666667
6	2	(a, e)	0.666667
7	2	(m, a)	0.500000
8	2	(a, n)	0.500000
9	2	(a, r)	0.500000
10	2	(a, t)	0.666667
11	2	(m, e)	0.500000
12	2	(e, n)	0.500000
13	2	(e, r)	0.500000
14	2	(e, t)	0.500000
15	2	(m, n)	0.500000
16	2	(r, t)	0.500000
17	3	(m, a, e)	0.500000
18	3	(a, e, n)	0.500000
19	3	(a, e, r)	0.500000
20	3	(a, e, t)	0.500000
21	3	(m, a, n)	0.500000
22	3	(a, r, t)	0.500000
23	3	(m, e, n)	0.500000
24	3	(e, r, t)	0.500000
25	4	(m, a, e, n)	0.500000
26	4	(a, e, r, t)	0.500000

:[84] In

```
frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                   (frequent_itemsets['support'] >= 0.5) ]
```

Out[84]:

	length	itemsets	support	
	2	(a, e)	0.666667	6
	2	(m, a)	0.500000	7
	2	(a, n)	0.500000	8
	2	(a, r)	0.500000	9
	2	(a, t)	0.666667	10
	2	(m, e)	0.500000	11
	2	(e, n)	0.500000	12
	2	(e, r)	0.500000	13
	2	(e, t)	0.500000	14
	2	(m, n)	0.500000	15
	2	(r, t)	0.500000	16

:[85] In

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'Diaper', 'Drink'} ]
```

Out[85]:

	length	itemsets	support

:[86] In

```
pip install Graphviz
```

```
Requirement already satisfied: Graphviz in e:\users\hp\anaconda3\lib\site-pa
(ckages (0.13.2
.Note: you may need to restart the kernel to use updated packages
```

:[87] In

```
import pandas as pd
import graphviz
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Setting random seed.
seed = 10
```

:[88] In

```
from sklearn import datasets
iris= datasets.load_iris()
df=pd.read_csv('F:\\mydataset.data', delimiter=',', header= 0, names= ['sepal length (cm)',

df.head()
```

Out[88]:

	Variety	(petal width (cm	(petal length (cm	(sepal width (cm	(sepal length (cm	
	Iris-setosa	0.2	1.4	3.0	4.9	0
	Iris-setosa	0.2	1.3	3.2	4.7	1
	Iris-setosa	0.2	1.5	3.1	4.6	2
	Iris-setosa	0.2	1.4	3.6	5.0	3
	Iris-setosa	0.4	1.7	3.9	5.4	4

:[89] In

```
# Creating a LabelEncoder and fitting it to the dataset labels.
le = LabelEncoder()
le.fit(df['Variety'].values)
# Converting dataset str labels to int labels.
y = le.transform(df['Variety'].values)
# Extracting the instances data.
X = df.drop('Variety', axis=1).values
# Splitting into train and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.34, stratify=y, randc
```

:[90] In

```
# Creating a DecisionTreeClassifier.
# The criterion parameter indicates the measure used (possible values: 'gini' for the Gini
# 'entropy' for the information gain).
# The min_samples_leaf parameter indicates the minimum of objects required at a leaf node.
# The min_samples_split parameter indicates the minimum number of objects required to split
# The max_depth parameter controls the maximum tree depth. Setting this parameter to None w
# tree until all leaves are pure or until all leaves contain less than min_samples_split sa
tree = DecisionTreeClassifier(criterion='gini',
                              min_samples_leaf=5,
                              min_samples_split=5,
                              max_depth=None,
                              random_state=seed)
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print('DecisionTreeClassifier accuracy score: {}'.format(accuracy))
```

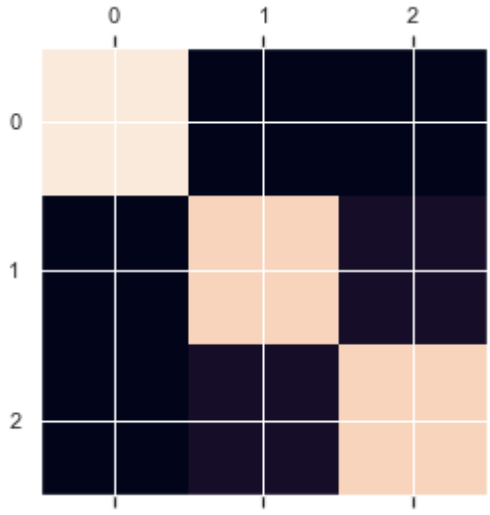
```
DecisionTreeClassifier accuracy score: 0.9607843137254902
```

:[91] In

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

print('Confusion Matrix is')
print(confusion_matrix(y_test, y_pred))
cm=confusion_matrix(y_test, y_pred)
plt.matshow(cm)
plt.show()
```

```
Confusion Matrix is
[0  0  17]]
[1  16 0 ]
[[16 1  0 ]
```



:[98] In

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

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

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

In [99]:

```
import pandas as pd
data= {'weather': ['Rainy','Rainy','Overcast','Sunny','Sunny','Sunny',
                   'Overcast','Rainy','Rainy','Sunny','Rainy','Overcast',
                   'Overcast','Sunny'],
       'temp': ['Hot','Hot','Hot','Mild','Cool','Cool','Cool',
                'Mild','Cool','Mild','Mild','Mild','Hot','Mild'],
       'play': ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','N

data= pd.DataFrame(data)
data
```

Out[99]:

	play	temp	weather
0	No	Hot	Rainy
1	No	Hot	Rainy
2	Yes	Hot	Overcast
3	Yes	Mild	Sunny
4	Yes	Cool	Sunny
5	No	Cool	Sunny
6	Yes	Cool	Overcast
7	No	Mild	Rainy
8	Yes	Cool	Rainy
9	Yes	Mild	Sunny
10	Yes	Mild	Rainy
11	Yes	Mild	Overcast
12	Yes	Hot	Overcast
13	No	Mild	Sunny

In [105]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [106]:

```
from sklearn import datasets

dataset= datasets.load_iris()
df= pd.DataFrame(dataset['data'], columns=['sepal length (cm)', 'sepal width (cm)', 'petal l
```

:[107] In

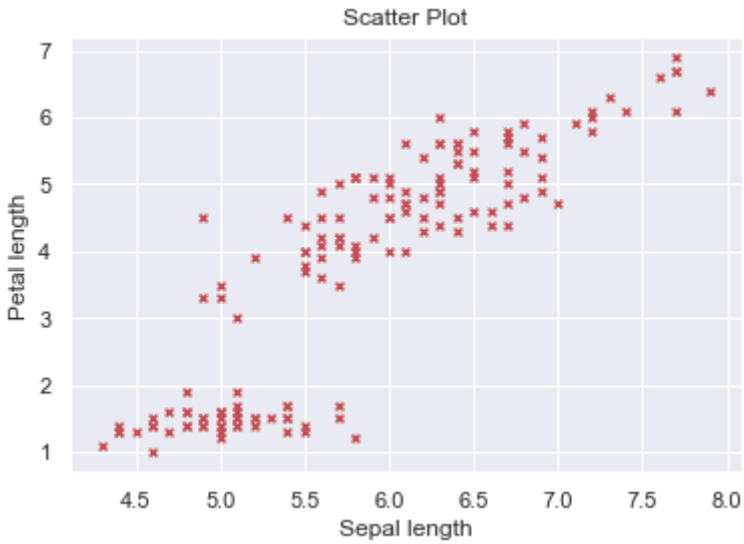
```
X= df['sepal length (cm)']
Y= df['petal length (cm)']
Slic_df = pd.DataFrame({'sepal length': X, 'petal length': Y})
print(Slic_df)
```

```
     sepal length  petal length
     1.4           5.1             0
     1.4           4.9             1
     1.3           4.7             2
     1.5           4.6             3
     1.4           5.0             4
     ...           ...            ..
     5.2           6.7           145
     5.0           6.3           146
     5.2           6.5           147
     5.4           6.2           148
     5.1           5.9           149

     [rows x 2 columns 150]
```

:[108] In

```
plt.scatter(Slic_df[['sepal length']], Slic_df[['petal length']], color = "r", marker = "x"
plt.xlabel('Sepal length')
plt.ylabel('Petal length')
plt.title('Scatter Plot')
plt.show()
```



:[109] In

```
from sklearn.linear_model import LinearRegression

#define the classifier
classifier = LinearRegression()

#train the classifier
model = classifier.fit(Slic_df[['sepal length']], Slic_df[['petal length']])
```

In [110]:

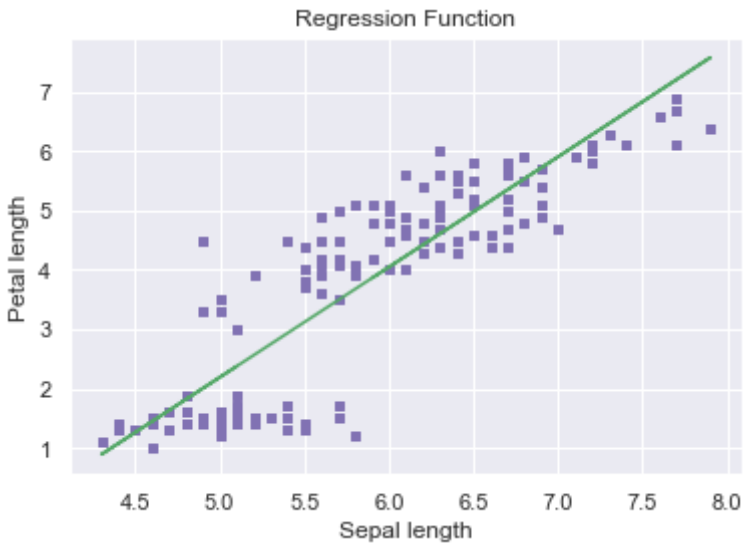
```
#use the trained classifier to make prediction
y_pred = classifier.predict(Slic_df[['sepal length']])
print(y_pred)

#print coefficient (a in y=ax+b) and intercept (the constant, b in y=ax+b)
print('Coefficients: \n', classifier.coef_)
print('Intercept: \n', classifier.intercept_)
```

```
[2.37656482]]
[2.00487822]
[1.63319163]
[1.44734833]
[2.19072152]
[2.93409471]
[1.44734833]
[2.19072152]
[1.07566173]
[2.00487822]
[2.93409471]
[1.81903493]
[1.81903493]
[0.88981844]
[ 3.6774679]
[3.49162461]
[2.93409471]
[2.37656482]
[3.49162461]
```

In [111]:

```
#visualize original data points
plt.scatter(Slic_df[['sepal length']], Slic_df[['petal length']], color = "m", marker = "s"
#visualize regression function
plt.plot(Slic_df['sepal length'], y_pred, color = "g")
plt.xlabel('Sepal length')
plt.ylabel('Petal length')
plt.title('Regression Function')
plt.show()
```



:[112] In

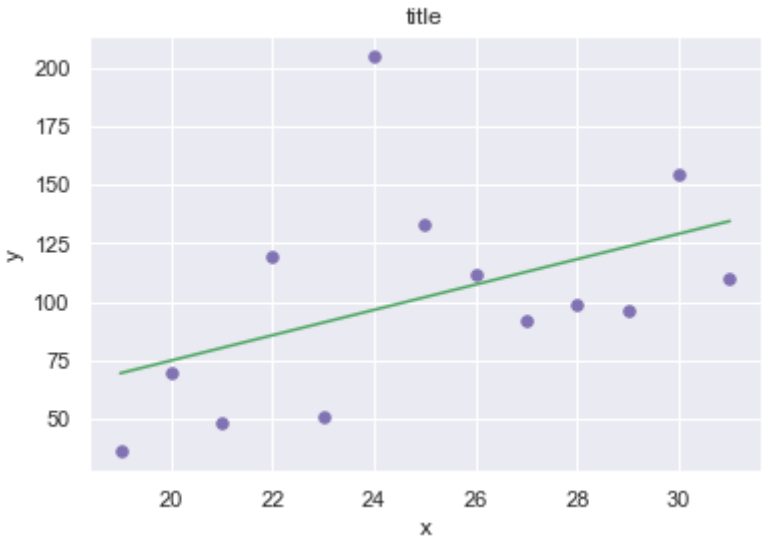
```
df2 = pd.DataFrame({'col1': [19, 20,21,22,23,24,25,26,27,28,29,30,31],
                    'col2': [36,70,48,119,51,205,133,112,92,99,96,154,110]})

#your turn - solution

#define the classifier
classifier = LinearRegression()
#train the classifier
classifier.fit(df2[['col1']],df2[['col2']])

#use the classifier to make prediction
y_predict = classifier.predict(df2[['col1']])

#visualize data points
plt.scatter(df2[['col1']], df2[['col2']], color = "m", marker = "o", s = 30)
#visualize regression function
plt.plot(df2[['col1']], y_predict, color = "g")
plt.xlabel('x')
plt.ylabel('y')
plt.title('title')
plt.show()
#print coefficient (a in y=ax+b) and intercept (the constant, b in y=ax+b)
print('Coefficients: \n', classifier.coef_)
print('Intercept: \n', classifier.intercept_)
```




```
 :Coefficients
[[5.41208791]]
 :Intercept
[33.37912088-]
```

:[113] In

```
Day_18 = classifier.intercept_+ classifier.coef_ * 18
print(Day_18)
```

```
[[64.03846154]]
```

:[114] In

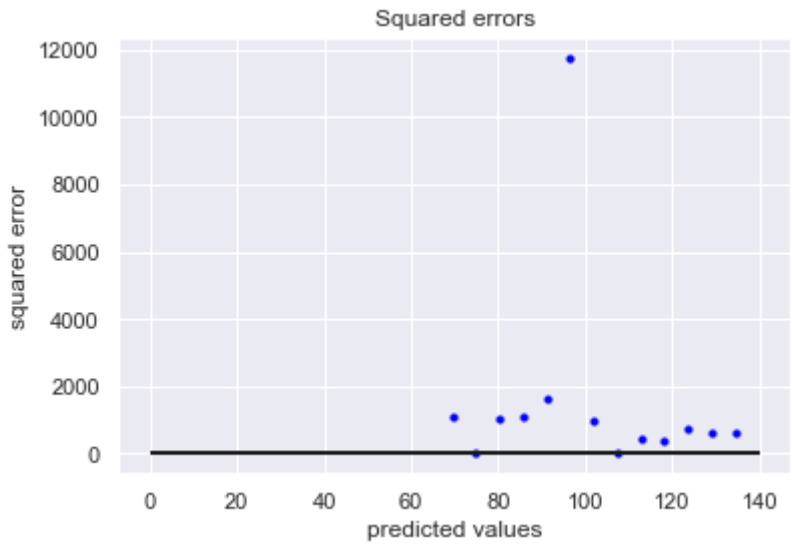
```
from sklearn.metrics import mean_squared_error, mean_absolute_error

print("Mean squared error:")
print(mean_squared_error(df2[['col2']], y_predict))
```

```
:Mean squared error
1572.1551141166526
```

:[115] In

```
plt.scatter(y_predict, (df2[['col2']] - y_predict) ** 2, color = "blue", s = 10,)
plt.title("Squared errors")
## plotting line for zero error
plt.hlines(y = 0, xmin = 0, xmax = 140, linewidth = 2)
plt.xlabel('predicted values')
plt.ylabel('squared error')
plt.show()
```

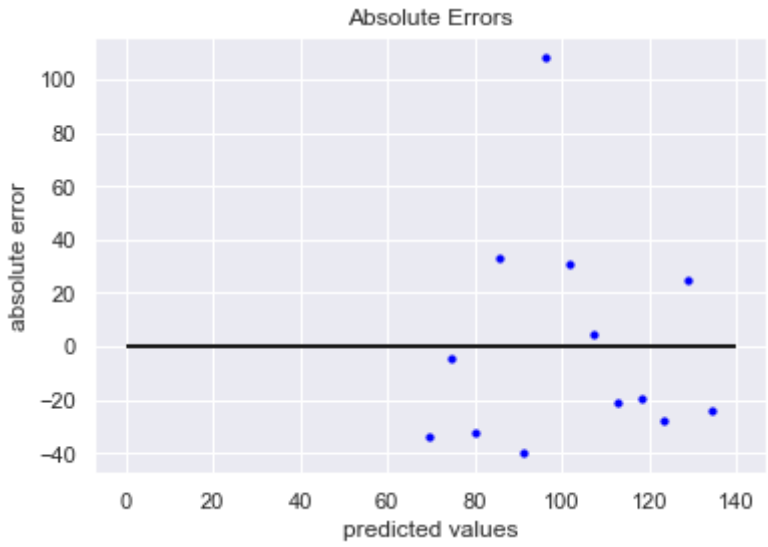


:[116] In

```
#Your turn
print("Mean absolute error: ")
print(mean_absolute_error(df2['col2'], y_predict))

#plot
plt.scatter(y_predict, (df2[['col2']] - y_predict), color = "blue", s = 10,)
plt.title("Absolute Errors")
plt.hlines(y = 0, xmin = 0, xmax = 140, linewidth = 2)
plt.xlabel('predicted values')
plt.ylabel('absolute error')
plt.show()
```

```
 :Mean absolute error
31.163144547759927
```



:[117] In

```
df['Species']= dataset['target']
df['Species']=df['Species'].apply(lambda x:dataset['target_names'][x])
df['Species']
```

Out[117]:

```
setosa         0
setosa         1
setosa         2
setosa         3
setosa         4
    ...
virginica    145
virginica    146
virginica    147
virginica    148
virginica    149
Name: Species, Length: 150, dtype: object
```

:[118] In

```
from sklearn.linear_model import LogisticRegression


X = Slic_df[['sepal length']] #we only use the first feature as descriptive feature
y = df['Species'] #use the species as target feature

#defining and training the logistic regression model
log_regression = LogisticRegression(solver = 'liblinear', multi_class = 'ovr')
log_regression.fit(X, y)

pred = log_regression.predict(X)

print('Score: \n', log_regression.score(X, y)) #the mean accuracy on the given test data an
print('Coefficients: \n', log_regression.coef_)
print('Intercept: \n', log_regression.intercept_)
```

```
 :Score
0.6933333333333334
 :Coefficients
[0.86959145-]]
[0.01223362 ]
[[0.57972675 ]
 :Intercept
[ 3.9921824- 0.74244291- 4.16186636 ]
```

:[119] In

```
#we use all descriptive features, but not the target feature
X = df.iloc[:, 0:4]
y=df['Species']
print('X: \n', X.head(), '\n')
print('y: \n', y.head(), '\n')

from sklearn.linear_model import LogisticRegression
classifier3 = LogisticRegression( solver = 'liblinear', multi_class = 'ovr')
classifier3.fit(X, y)

y_pred = classifier3.predict(X)

print('Score: \n', classifier3.score(X, y))
print('Coefficients: \n', classifier3.coef_)
print('Intercept: \n', classifier3.intercept_)
```

```
 :X
(sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm
0.2               1.4                3.5               5.1                0
0.2               1.4                3.0               4.9                1
0.2               1.3                3.2               4.7                2
0.2               1.5                3.1               4.6                3
 0.2               1.4                3.6               5.0                4

 :y
setosa    0
setosa    1
setosa    2
setosa    3
setosa    4
 Name: Species, dtype: object

 :Score
0.96
 :Coefficients
[1.02103509- 2.26003266- 1.46416217  0.41021713 ]]
[1.40617325-  0.5758173  1.61211605-  0.4275087 ]
[[2.55537041  2.47096755  1.53427768- 1.70751526-]
 :Intercept
[1.21470917- 1.09392467  0.26421853 ]
```

:[ ] In

```

```