Name: Adiba

Email id: adibak1919@gmail.com

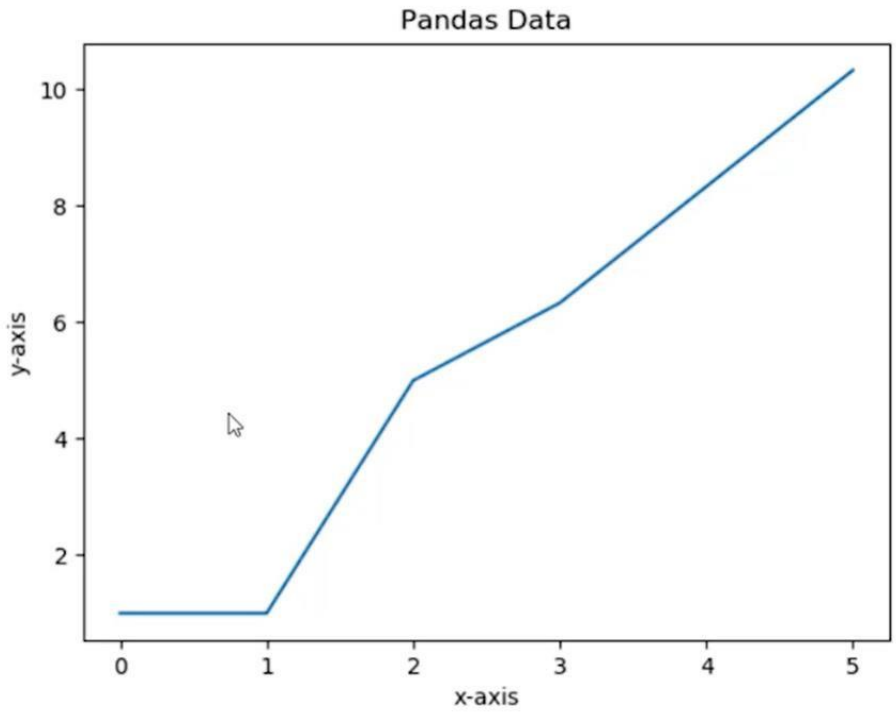
MACHINE LEARNING

Python for data visualization

Matplotlib (Part-1)

```
import matplotlib.pyplot as plt
import numpy as np  #fetch data from excel sheet
a,b,c=np.loadtxt('C:/Users/ADIBA/Addy/Desktop/pandas_data.csv',unpack=True,delimiter=',')
#importing excel sheet which is in csv format
# plt.plot([1,2,3,4],[5,5,6,7])
# plt.axis([0,6,0,20])
# plt.title('Demo')
# plt.xlabel('x-axis') #
plt.ylabel('y-axis') #
plt.show()
plt.plot(a,gs) #gs is for colour and shape
plt.title('Pandas Data') plt.xlabel('x-
axis') plt.ylabel('y-axis') plt.show()
```

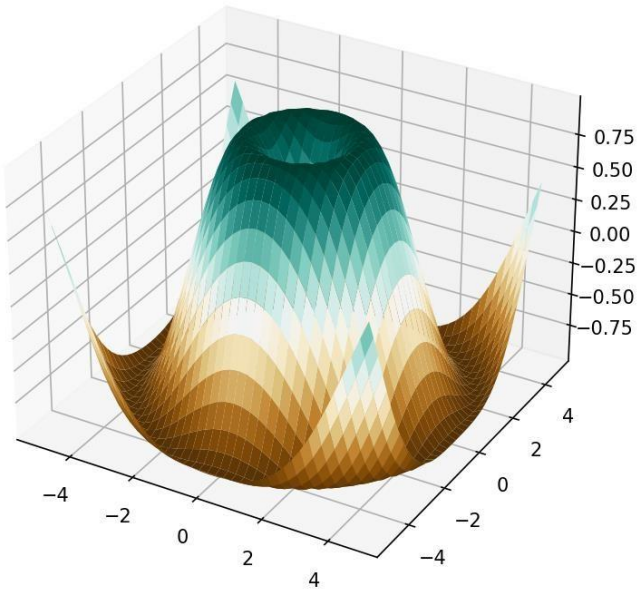
Output:



Matplotlib (Part-2)

```
#creating a 3d structure in matplotlib
import matplotlib.pyplot as plt import
numpy as np
from mpl_toolkits.mplot3d import Axes3D  #to handle 3d projection
from matplotlib import cm  #color or type of visualisation fig=plt.figure()
ax=fig.gca(projection='3d') #this gives types of projection
x=np.arange(-5,5,0.25) y=np.arange(-5,5,0.25)
x,y=np.meshgrid(x,y) #for preparing data r=np.sqrt(x**2+y**2)
z=np.sin(r)  #this will do sin function of all the values x and y
surf=ax.plot_surface(x,y,z,rstride=1,cstride=1,cmap=cm.BrBG) #for plotting cm=cm.coolwarm
plt.show()
```

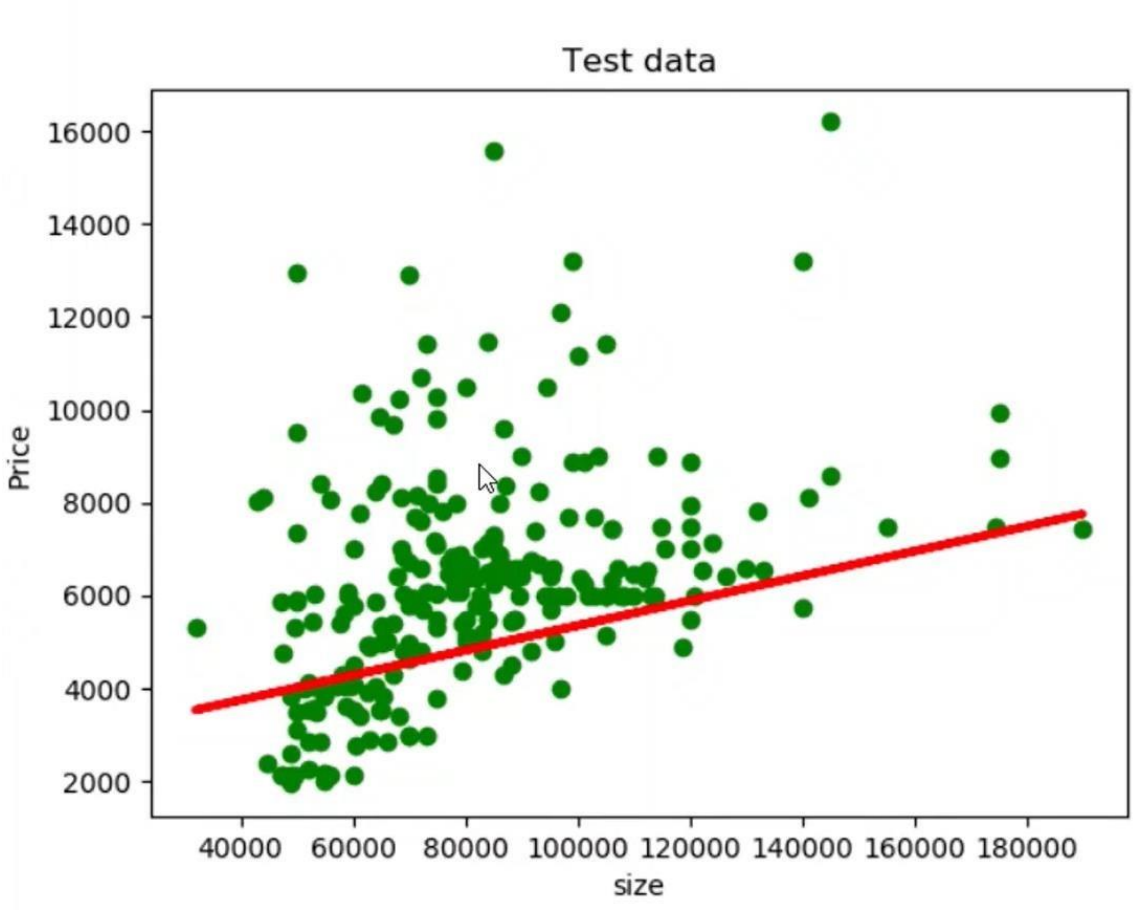
Output:



Linear Regression

```
import pandas as pd import
matplotlib .pyplot as plt import
numpy as np
from sklearn import linear_model
#Gathering of Data
data=pd.read_csv("C:/Users/ADIBA/Addy/Desktop/House_Data.csv")
x=np.array(data['Lotsize']) y=np.array(data['Price'])
x=x.reshape(len(x),1)
y=y.reshape(len(y),1)
```

```
#Preparing or Splitting the data
x_train=x[:-250] x_test=x[-
250:] y_train=y[:-250]
y_test=y[-250:] #Choosing
model
regr=linear_model.LinearRegression()
#Training model regr.fit(x_train,y_train)
plt.scatter(x_test,y_test,color='green')
plt.plot(x_test,regr.predict(x_test),color='red',linewidth=3)
plt.title('Test Data') plt.xlabel('Price') plt.ylabel('Size')
plt.show() Output:
```



K Nearest Neighbors

from sklearn.datasets import load_iris from

sklearn.neighbors import KNeighborsClassifier

iris=load_iris() knn=KNeighborsClassifier(n_neighbors=3) #bydefault 5

but we set 3 here knn.fit(iris.data,iris.target) a=knn.predict([[4,6,8,10],])

print(iris.target_names[a])

Output:

['virginica']

iris.feature_names

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

iris.target array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 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, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])

iris.target_names array(['setosa', 'versicolor',

'virginica'], dtype='<U10')

Comparing different classification models

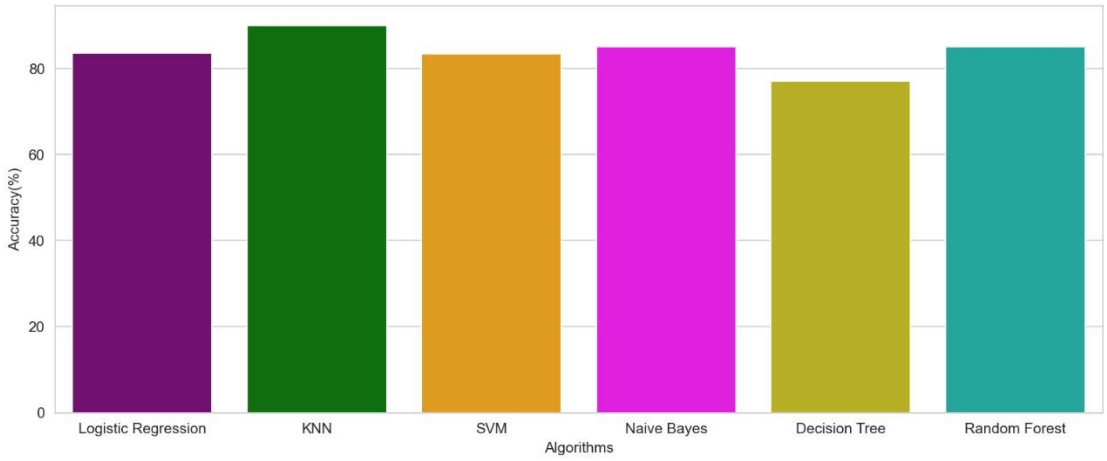
```
import pandas as pd, numpy as np from
sklearn.model_selection import train_test_split from
sklearn.linear_model import LogisticRegression from
sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
from sklearn.svm import SVC from
sklearn.naive_bayes import GaussianNB from
sklearn.tree import DecisionTreeClassifier from
sklearn.ensemble import RandomForestClassifier
import seaborn as sb

#gathering of data
data=pd.read_csv("C:/Users/ADIBA/Addy/Desktop/heart.csv") #import all data

y=data.target.values
x_data=data.drop(['target'],axis=1)
x=(x_data-np.min(x_data))/(np.max(x_data)-np.min(x_data)).values
```

```
#preparing of data (split the data)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

#choosing a model #Logistic
regression
log=LogisticRegression()
log.fit(x_train,y_train) #fit train our model
print('Test accuracy of Logistic regression: {}'.format(log.score(x_test,y_test)*100)) #tell us the score and
accuracy of model
#we multiply it to 100 because value coming from test will be from 0 to 1

#KNN
knn=KNeighborsClassifier(n_neighbors=2)  #create model with neighbor 3 or any number you want to
set up to 5
knn.fit(x_train,y_train) #train the model
prediction=knn.predict(x_test) #predicted values for knn
# print('Test accuracy of KNN: {}'.format(knn.score(x_test,y_test)*100))


#for checking the accuracy

scorelist=[] #store our all values here or accuracy inside list for
i in range(1,20):
   knn=KNeighborsClassifier(i)
knn.fit(x_train,y_train)
prediction=knn.predict(x_test)
   scorelist.append(knn.score(x_test,y_test)*100) #adding values inside list print('Test
accuracy of KNN: {}'.format(max(scorelist)))
#Now visualize curve so that we get at what position we get max accuracy
#with what no. of neighbors we get the max accuracy so for this import matplotlib
# plt.plot(range(1,20),scorelist) #plot a graph
# plt.xlabel('K value')
# plt.ylabel('Score')
# plt.show()

#support vector machine to train our model and further predict hte values
svm=SVC(random_state=1) svm.fit(x_train,y_train)
print('Test accuracy of support vector machine: {}'.format(svm.score(x_test,y_test)*100))

#Naive Bayes Algorithm nb=GaussianNB()
nb.fit(x_train,y_train)
print('Test accuracy of Naive Bayes: {}'.format(nb.score(x_test,y_test)*100))

#Decision tree algorithm
dtc=DecisionTreeClassifier() dtc.fit(x_train,y_train)
print('Test accuracy of Decision Tree: {}'.format(dtc.score(x_test,y_test)*100))

#Random Forest Algorithm
```

```
rf=RandomForestClassifier(n_estimators=1000,random_state=1) rf.fit(x_train,y_train)
print('Test accuracy of Random Forest: {}'.format(rf.score(x_test,y_test)*100))

#compare all values and accordingly display it to graph to directly visualize it
methods=['Logistic Regression','KNN','SVM','Naive Bayes','Decision Tree','Random Forest']
accuracy=[83.606,90.16,83.60,85.24,77.049,85.24]
colors=['purple','green','orange','magenta','#CFC60E','#0FBBAE']
sb.set_style('whitegrid') #using sklearn plt.figure(figsize=(16,5))
plt.xlabel('Algorithms') plt.ylabel('Accuracy(%)')
sb.barplot(x=methods,y=accuracy,palette=colors) plt.show()
```

Output:

```
Test accuracy of Logistic regression: 83.60655737704919
Test accuracy of KNN: 90.1639344262295
Test accuracy of support vector machine: 83.60655737704919
Test accuracy of Naive Bayes: 85.24590163934425
Test accuracy of Decision Tree: 73.77049180327869
Test accuracy of Random Forest: 85.24590163934425
```



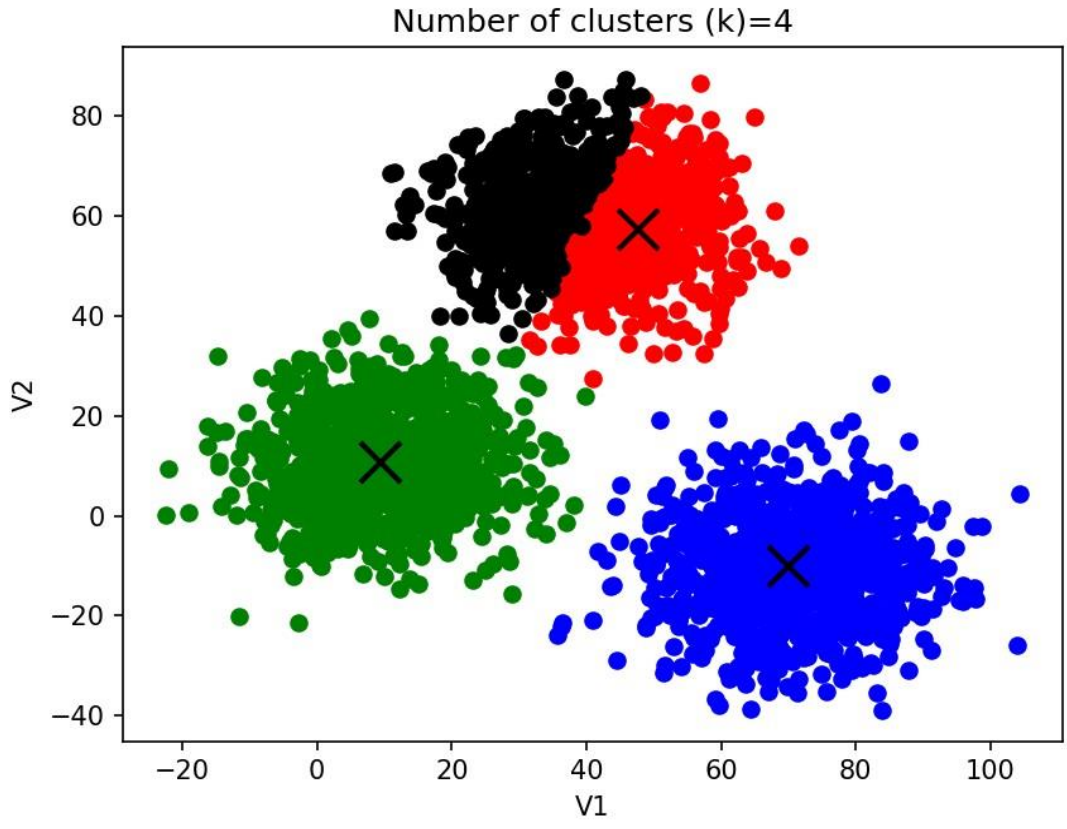
K means Clustering

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt from
sklearn.cluster import KMeans
data=pd.read_csv('C:/Users/ADIBA/Addy/Desktop/xclara.csv')
print(data.head()) k=4 #no.of clusters or categories
kmean=KMeans(n_clusters=k)
#next step is to train a model this we we need not to prepare our data #because in our data we have only input values
kmean=kmean.fit(data) #(data) will take all the input values
labels=kmean.labels_ #labels is an array which contains the cluster numbers
centroids=kmean.cluster_centers_
#next step is to prepare the data to test
# Testing data
x_test=[[49.6,67],[27.88,60],[94.65,98],[31.4,-56],[-1.33,5.6],[14.555,-1.22]] #we take 2 2 values because we have only v1 and v2 in our data prediction=kmean.predict(x_test) print(prediction)
```

```
colors=['blue','red','green','black']
y=0

for x in labels:  #inside labels we have many values and these values are cluster numbers we use loop so that we can plot the data according to that
   plt.scatter(data.iloc[y,0],data.iloc[y,1],color=colors[x]) #[y,0] means we are accessing only v1
   #[y,1] is for v2 #color scatter data for v1 and v2 and take color acc to that
y+=1
for x in range(k): #k=2 that means loop will run 2 times
lines=plt.plot(centroids[x,0],centroids[x,1],'kx')
   # kx will differentiate data into 2 different categories in (0,1)
plt.setp(lines,ms=15.0)   #to make centroid larger     plt.setp(lines,mew=2.0)
title=('Number of clusters (k)={}').format (k)
plt.title(title) plt.xlabel('V1') plt.ylabel('V2')
plt.show()
```

Output:

```
       V1        V2
0   2.072345  -3.241693
1   17.936710  15.784810
2   1.083576   7.319176
3   11.120670  14.406780
4   23.711550   2.557729

[1 3 1 0 2 2]
```



Apriori Algorithm

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

#we use transactionencoder because we want to convert data acc to dataset #and to create
#dataframe acc to that dataset=[['Milk','Onion','Nutmeg','Kidney
Beans','Eggs','Yogurt'],
      ['Dill','Onion','Nutmeg','Kidney Beans','Eggs','Yogurt'],
      ['Milk','Apple','Kidney Beans','Eggs'],
      ['Milk','Unicorn','Corn','Kidney Beans','Yogurt'],
      ['Corn','Onion','Onion','Kidney Beans','Ice cream','Eggs']] te=TransactionEncoder()
#converting data into dataframe below Tran_array=te.fit(dataset).transform(dataset)
df=pd.DataFrame(Tran_array,columns=te.columns_)
# print(df)
#now we have dataframe next step is to use apriori alog to check the support for each values
ap=apriori(df,min_support=0.6,use_colnames=True)
# print(ap)
#next step is to add a new column for apriori model because in here we can see the lengths of item
we have but we here we dont know how the machine learning algo will predict these items
ap['Length']=ap['itemsets'].apply(lambda x:len(x)) print(ap)
#next step is to check all the values of itemsets out of which we want to figure out any two values or
any two items which we will sold together and having a minimum confidence level
# print(ap[(ap['Length']==2) & (ap['support']>=0.8)])       #output shows eggs and kidney beans can be
sold together
#we can change the length and support values
# print(ap[(ap['Length']==3) & (ap['support']>=0.7)])  #this time no condition satisfied
print(ap[(ap['Length']==2) & (ap['support']>=0.6)])
```

Output:

```
   support              itemsets  Length
0      0.8               (Eggs)       1
1      1.0         (Kidney Beans)       1
2      0.6               (Milk)       1
3      0.6              (Onion)       1
4      0.6             (Yogurt)       1
5      0.8      (Kidney Beans, Eggs)       2
6      0.6        (Onion, Eggs)       2
7      0.6      (Kidney Beans, Milk)       2
8      0.6     (Onion, Kidney Beans)       2
9      0.6    (Kidney Beans, Yogurt)       2 10     0.6  (Onion, Kidney
       Beans, Eggs)       3   support           itemsets  Length 5      0.8
       (Kidney Beans, Eggs)       2
6      0.6        (Onion, Eggs)       2
7      0.6    (Kidney Beans, Milk)       2
8      0.6   (Onion, Kidney Beans)       2
9      0.6  (Kdney Beans, Yogurt)       2
```