**Scope Of Project:**

To Build a machine learning model to predict the results accurately.for bulding machine learning model we need to follow following steps

1.Data Preparation

2.Data visualisaton

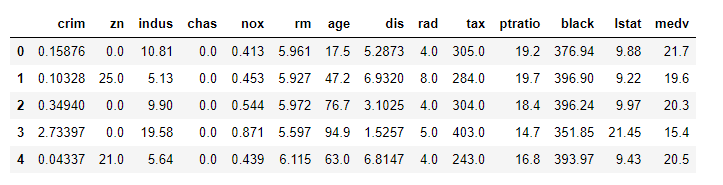
3.Data Preprocessing

4.Building model

5.Evaluating the results

**Undestanding Botson Dataset:**

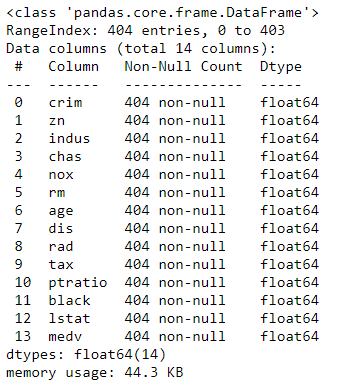
1. import numpy as np
2. import pandas as pd
3. import seaborn as sns
4. import matplotlib.pyplot as plt
5. from sklearn.linear\_model import LinearRegression
6. from sklearn.model\_selection import train\_test\_split
7. from sklearn.metrics import mean\_absolute\_error
8. dataset=pd.read\_csv('boston\_data.csv')
9. dataset.head()



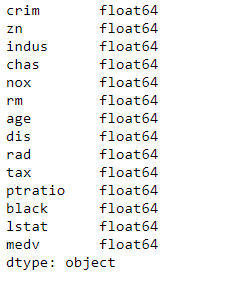
1. dataset.shape



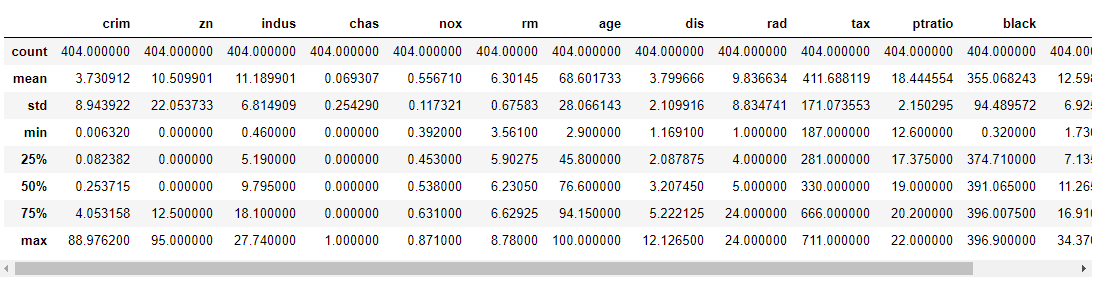
1. dataset.info()



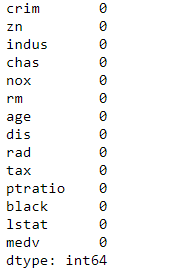
1. dataset.dtypes



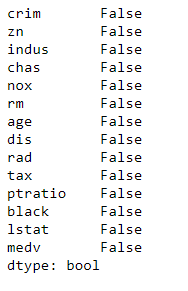
1. dataset.describe()



1. dataset.isnull().sum()



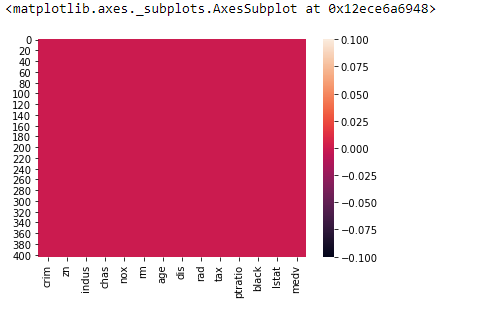
1. dataset.isnull().any()



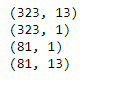
1. dataset.plot(subplots=True)



1. sns.heatmap(dataset.isnull())



1. x\_train=dataset[['crim','zn','indus','chas','nox','rm','age','dis','rad','tax','ptratio','black','lstat']]
2. x\_train=np.array(x\_train)
3. y\_train=dataset[['medv']]
4. y\_train=np.array(y\_train)
5. X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x\_train,y\_train,test\_size=0.2)
6. print(X\_train.shape)
7. print(Y\_train.shape)
8. print(Y\_test.shape)
9. print(X\_test.shape)



1. model=LinearRegression()
2. model.fit(X\_train,Y\_train)



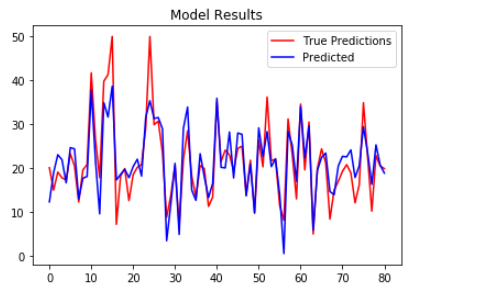
1. score=model.score(X\_test,Y\_test)
2. print(score)



1. predictions=model.predict(X\_test)
2. error=mean\_absolute\_error(Y\_test,predictions)
3. error



1. plt.plot(Y\_test,label='True Predictions',color='r')
2. plt.plot(predictions,label='Predicted',color='b')
3. plt.title('Model Results')
4. plt.legend()
5. plt.show()

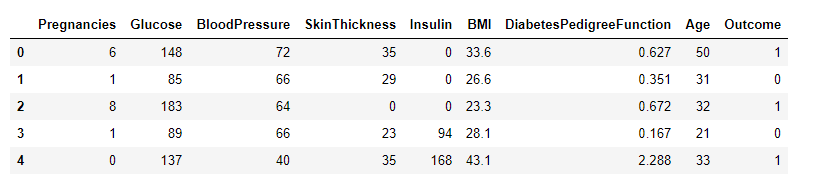


**Understanding Diabetic Dataset:**

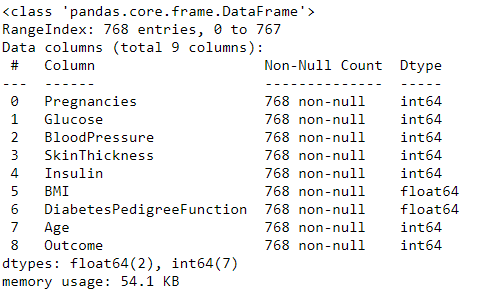
**Undersatanding Diabetic Dataset:**

1. import numpy as np
2. import pandas as pd
3. import seaborn as sns
4. import matplotlib.pyplot as plt
5. from sklearn.linear\_model import LogisticRegression
6. from sklearn.model\_selection import train\_test\_split
7. from sklearn.metrics import accuracy\_score,confusion\_matrix,plot\_confusion\_matrix,classification\_report

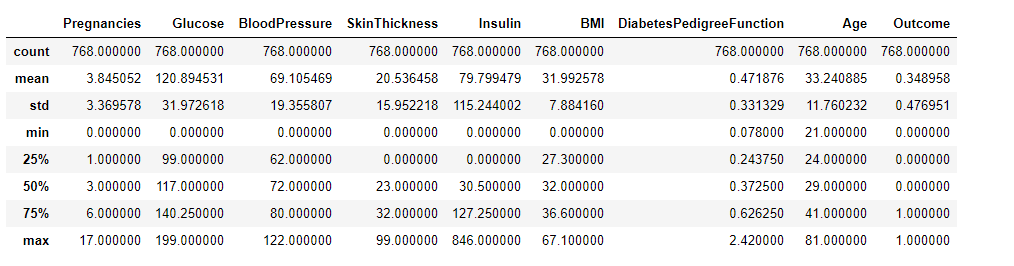
1. dataset=pd.read\_csv('diabetes.csv')
2. dataset.head()



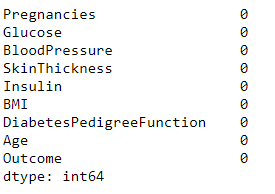
1. dataset.info()



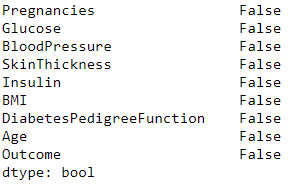
1. dataset.describe()



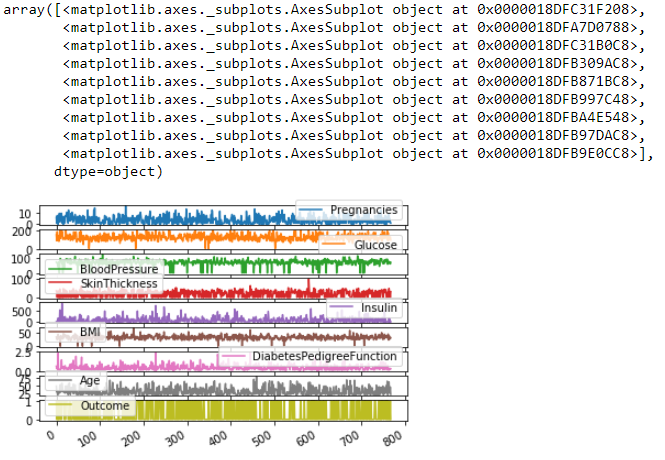
1. dataset.isnull().sum()



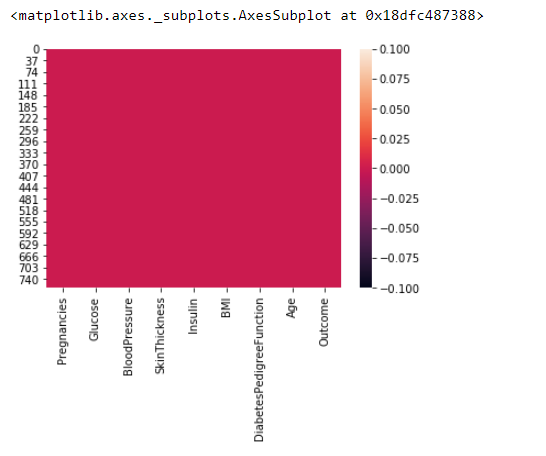
1. dataset.isnull().any()



1. dataset.plot(subplots=True)



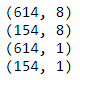
1. sns.heatmap(dataset.isnull())



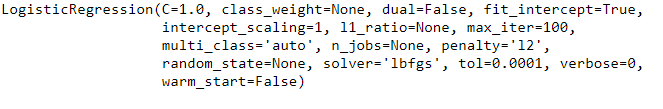
1. data\_in=dataset[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']]
2. data\_out=dataset[['Outcome']]
3. data\_in=np.array(data\_in)
4. data\_out=np.array(data\_out)
5. print(data\_in.shape)
6. print(data\_out.shape)



1. X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(data\_in,data\_out,test\_size=0.2)
2. print(X\_train.shape)
3. print(X\_test.shape)
4. print(Y\_train.shape)
5. print(Y\_test.shape)



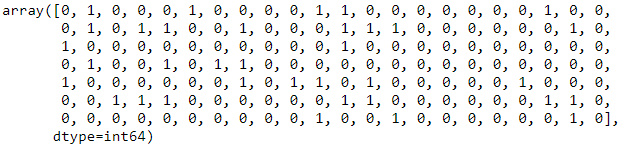
1. model=LogisticRegression()
2. model.fit(X\_train,Y\_train)



1. score=model.score(X\_test,Y\_test)
2. print(score)



1. predictions=model.predict(X\_test)
2. predictions



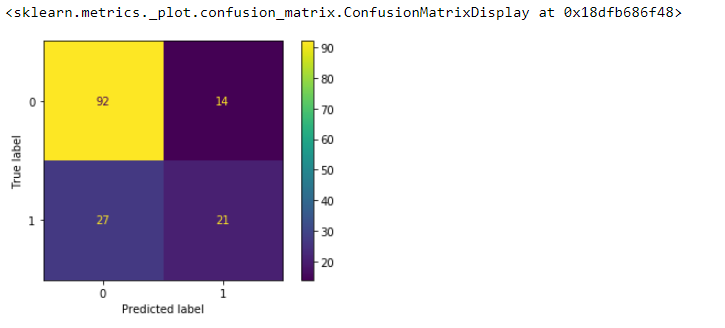
1. conf=confusion\_matrix(Y\_test,predictions)
2. conf



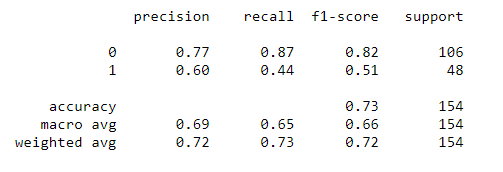
1. acc=accuracy\_score(Y\_test,predictions)
2. acc



1. plot\_confusion\_matrix(model,X\_test,Y\_test)



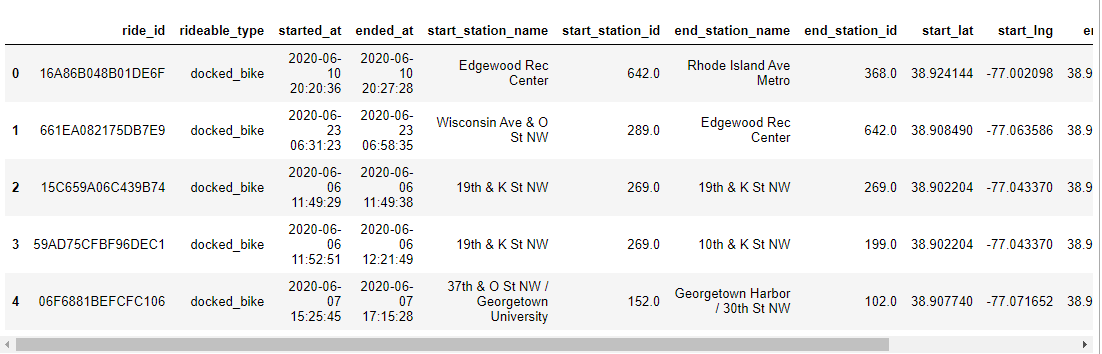
1. report=classification\_report(Y\_test,predictions)
2. print(report)



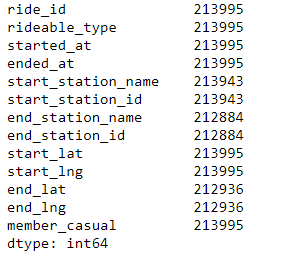
**UnderStanding BikeSharing Dataset:**

1. import numpy as np
2. import pandas as pd
3. import seaborn as sns
4. from sklearn.metrics import
5. accuracy\_score,confusion\_matrix,plot\_confusion\_matrix,classification\_report
6. from sklearn.tree import DecisionTreeClassifier
7. from sklearn.model\_selection import train\_test\_split
8. from sklearn.preprocessing import LabelEncoder

1. dataset=pd.read\_csv('202006-capitalbikeshare-tripdata.csv')
2. dataset.head()



1. dataset.count()



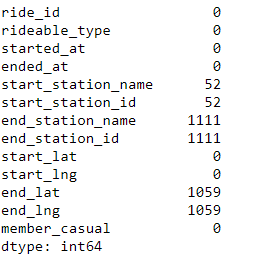
1. dataset['rideable\_type'].unique()



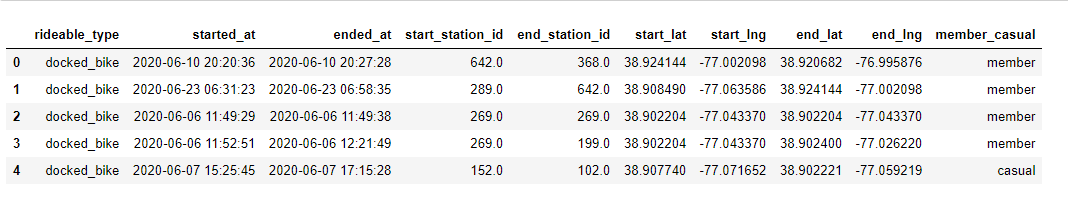
1. dataset['member\_casual'].unique()



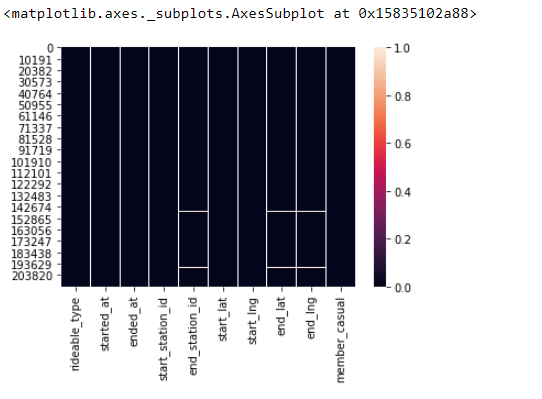
1. dataset.isnull().sum()



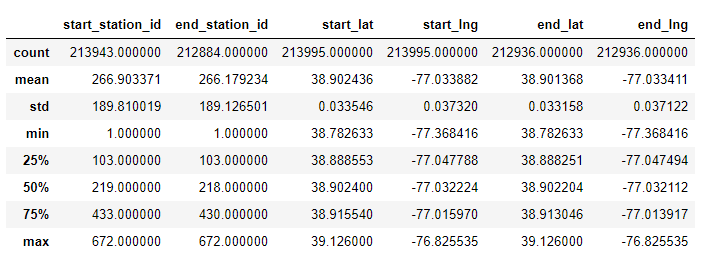
1. dataset=dataset.drop(['start\_station\_name','end\_station\_name','ride\_id'],axis=1)
2. dataset.head()



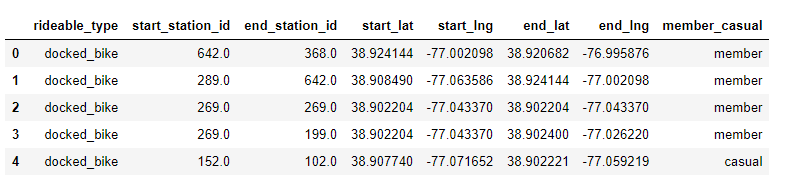
1. sns.heatmap(dataset.isnull())



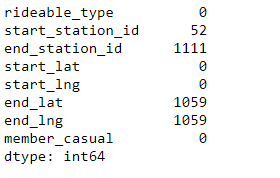
1. dataset.describe()



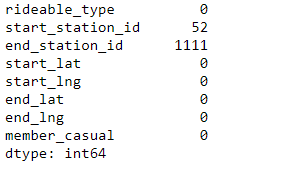
1. dataset=dataset.drop(['started\_at','ended\_at'],axis=1)
2. dataset.head()



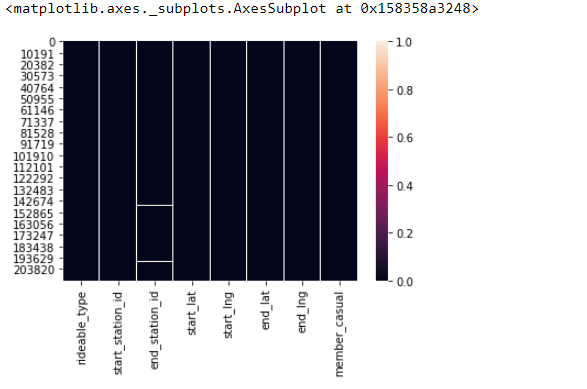
1. dataset.isnull().sum()



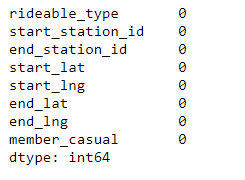
1. dataset['end\_lat']=dataset['end\_lat'].fillna(np.mean(dataset['end\_lat']))
2. dataset['end\_lng']=dataset['end\_lng'].fillna(np.mean(dataset['end\_lng']))
3. dataset.isnull().sum()



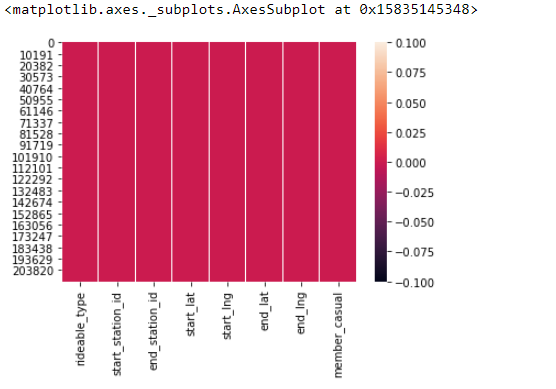
1. sns.heatmap(dataset.isnull())



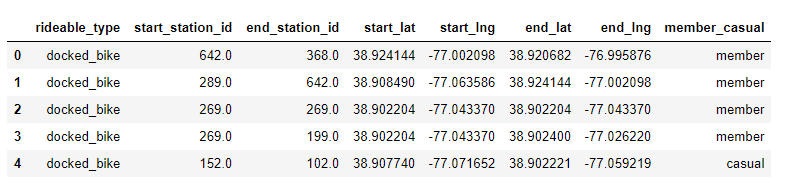
1. dataset['start\_station\_id']=dataset['start\_station\_id'].fillna(dataset['start\_station\_id'].value\_counts().index[0])
2. dataset['end\_station\_id']=dataset['end\_station\_id'].fillna(dataset['end\_station\_id'].value\_counts().index[0])
3. dataset.isnull().sum()



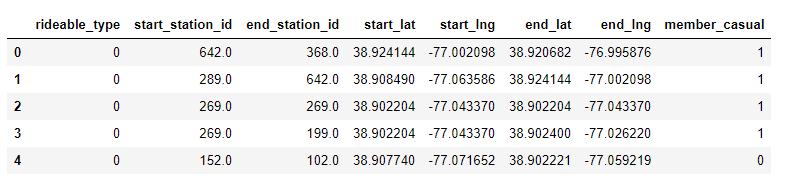
1. sns.heatmap(dataset.isnull())



1. dataset.head()



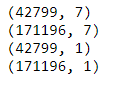
1. enc=LabelEncoder()
2. dataset['rideable\_type']=enc.fit\_transform(dataset['rideable\_type'])
3. dataset['member\_casual']=enc.fit\_transform(dataset['member\_casual'])
4. dataset.head()



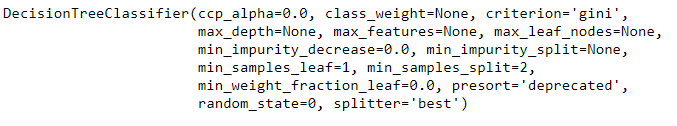
1. x\_data=dataset[['rideable\_type','start\_station\_id','end\_station\_id','start\_lat','start\_lng','end\_lat','end\_lng']]
2. y\_data=dataset[['member\_casual']]
3. x\_data=np.array(x\_data)
4. y\_data=np.array(y\_data)
5. print(x\_data.shape)
6. print(y\_data.shape)



1. X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x\_data,y\_data,test\_size=0.2)
2. print(X\_test.shape)
3. print(X\_train.shape)
4. print(Y\_test.shape)
5. print(Y\_train.shape)



1. model=DecisionTreeClassifier(random\_state=0)
2. model.fit(X\_train,Y\_train)



1. predictions=model.predict(X\_test)
2. predictions



1. label\_pred=enc.inverse\_transform(predictions)
2. label\_pred



1. score=model.score(X\_test,Y\_test)
2. print(score)



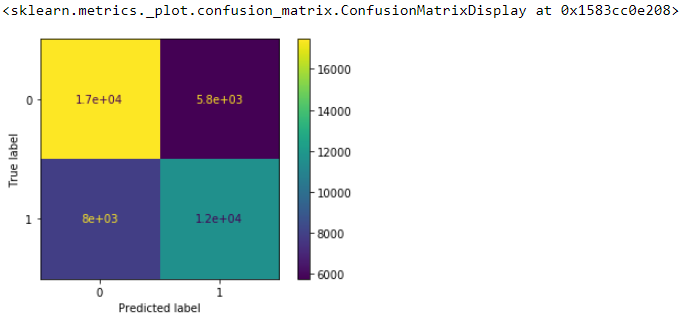
1. conf=confusion\_matrix(Y\_test,predictions)
2. conf



1. acc=accuracy\_score(Y\_test,predictions)
2. acc



1. plot\_confusion\_matrix(model,X\_test,Y\_test)



1. report=classification\_report(Y\_test,predictions)
2. print(report)

