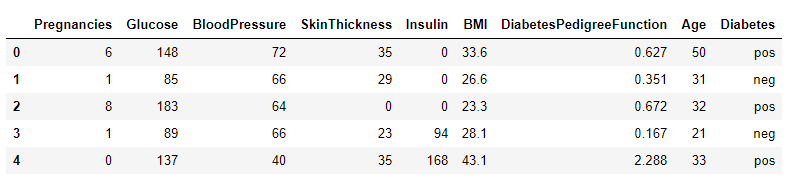
**Scope Of Project:**

In this project, we need diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

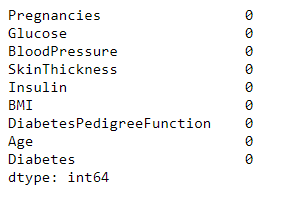
The datasets consist of several medical predictor variables and one target variable, Diabetes. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. from sklearn.preprocessing import LabelEncoder
5. from sklearn.preprocessing import StandardScaler
6. from sklearn.model\_selection import train\_test\_split
7. from sklearn.linear\_model import LogisticRegression
8. from sklearn.neighbors import KNeighborsClassifier
9. from sklearn.metrics import accuracy\_score,confusion\_matrix,roc\_curve,auc
10. from sklearn.metrics import plot\_confusion\_matrix,classification\_report
11. import seaborn as sns

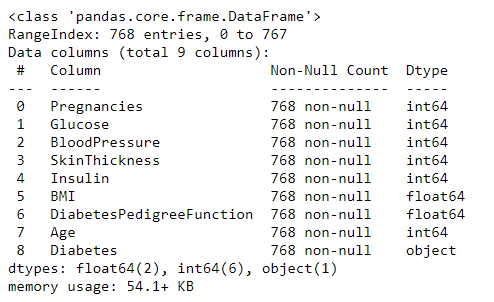
1. dataset=pd.read\_csv('diabetes.csv')
2. dataset.head()



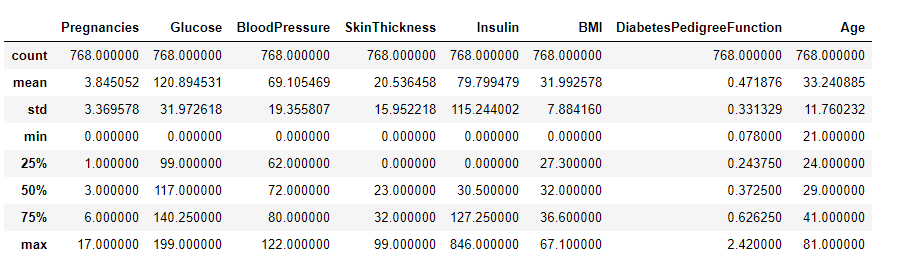
1. dataset.isnull().sum()



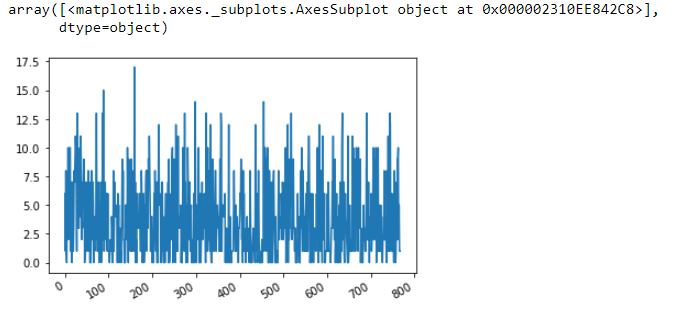
1. dataset.info()



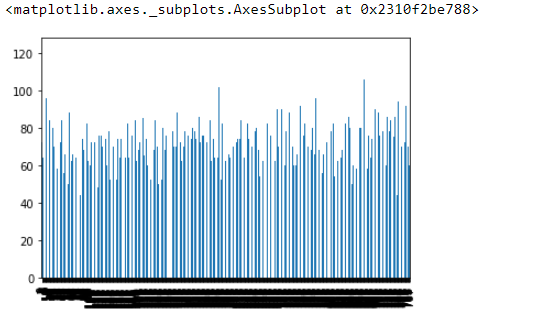
1. dataset.describe()



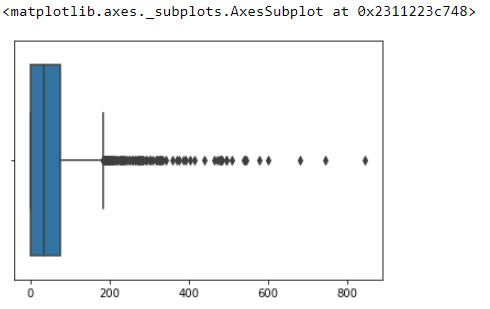
1. dataset['Pregnancies'].plot(kind='line',subplots=True)



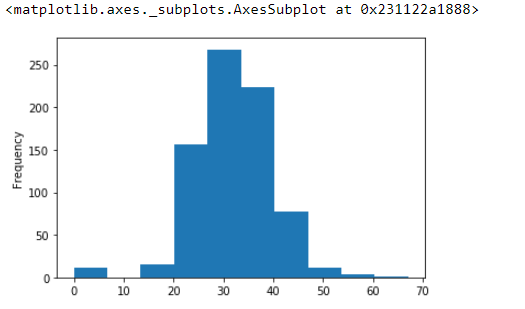
1. dataset['BloodPressure'].plot(kind='bar')



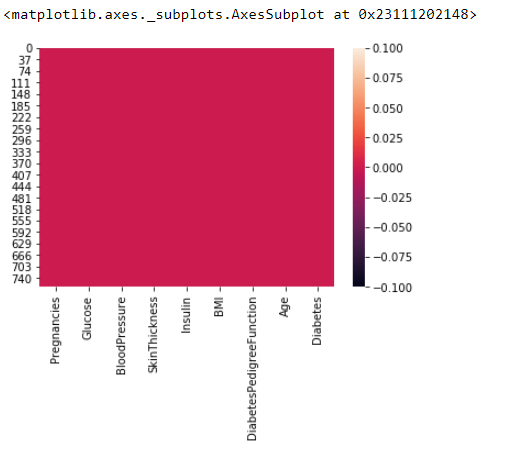
1. sns.boxplot(dataset.iloc[:,1:-1])



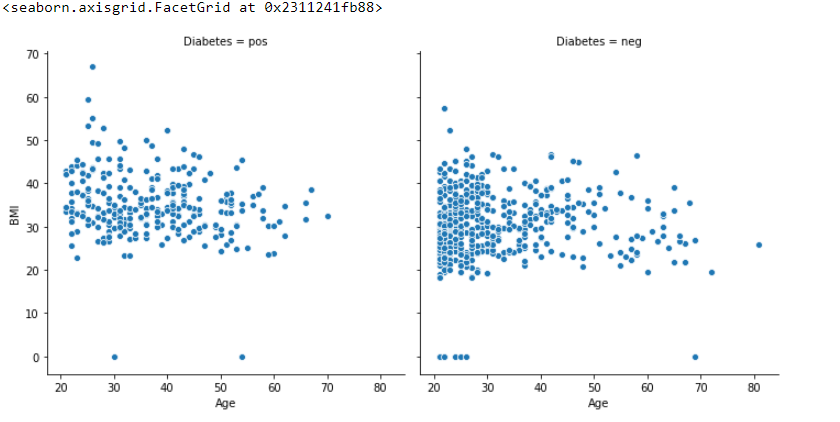
1. dataset['BMI'].plot(kind='hist')



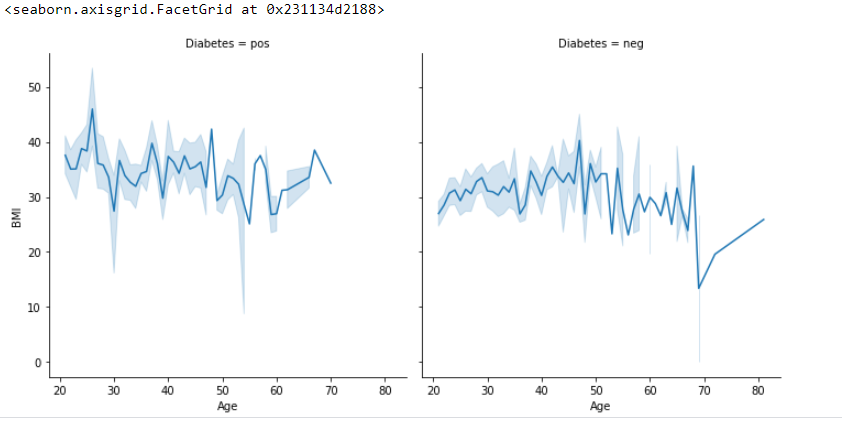
1. sns.heatmap(dataset.isnull())



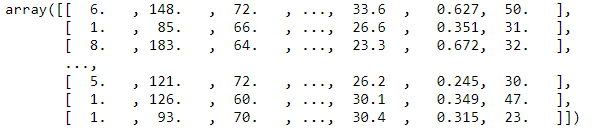
1. sns.relplot(x='Age',y='BMI',data=dataset,col='Diabetes')



1. sns.relplot(x='Age',y='BMI',data=dataset,col='Diabetes',kind='line')



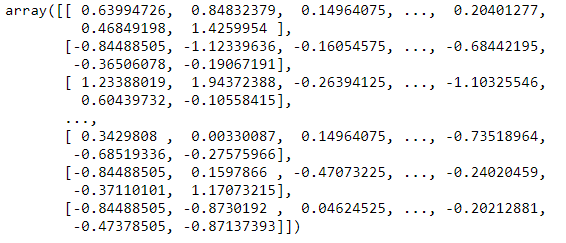
1. x=dataset.iloc[:,0:-1].values
2. x



1. y=dataset.iloc[:,-1].values
2. le=LabelEncoder()
3. y=le.fit\_transform(y)
4. y[:10]



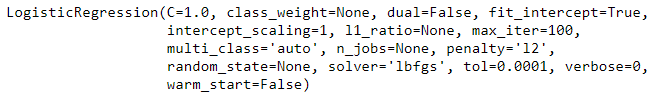
1. sc=StandardScaler()
2. x=sc.fit\_transform(x)
3. x



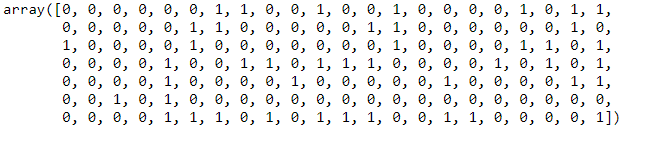
1. x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2)
2. print(x\_train.shape)
3. print(y\_train.shape)
4. print(x\_test.shape)
5. print(y\_test.shape)



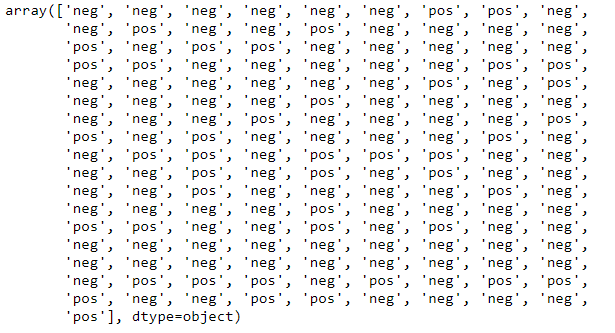
1. model1=LogisticRegression()
2. model1.fit(x\_train,y\_train)



1. pred1=model1.predict(x\_test)
2. pred1



1. pr1=le.inverse\_transform(pred1)
2. pr1



1. usr=[4,140,80,30,90,26.9,1.25,27]
2. usr=np.array([usr])
3. usr=sc.fit\_transform(usr)
4. le.inverse\_transform(model1.predict(usr))



1. accuracy\_score(y\_test,pred1)



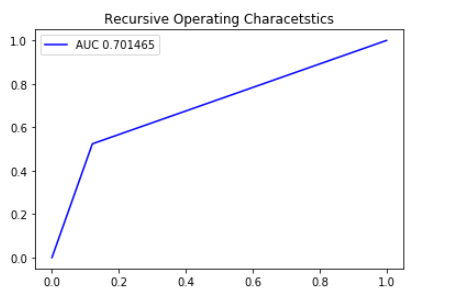
1. confusion\_matrix(y\_test,pred1)



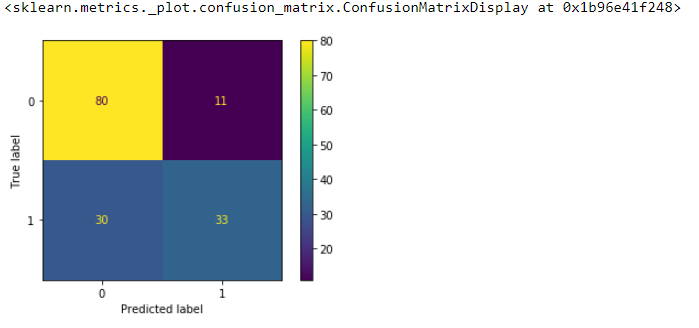
1. fpr,tpr,threshold=roc\_curve(y\_test,pred1)
2. roc\_auc=auc(fpr,tpr)
3. roc\_auc



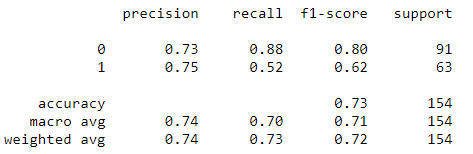
1. plt.title('Recursive Operating Characetstics')
2. plt.plot(fpr,tpr,'b',label='AUC %f'%roc\_auc)
3. plt.legend()
4. plt.show()



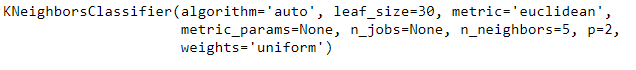
1. plot\_confusion\_matrix(model1,x\_test,y\_test)



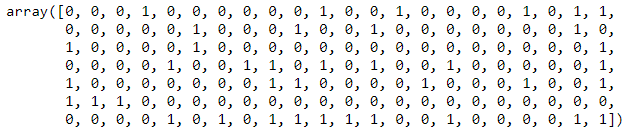
1. report=classification\_report(y\_test,pred1)
2. print(report)



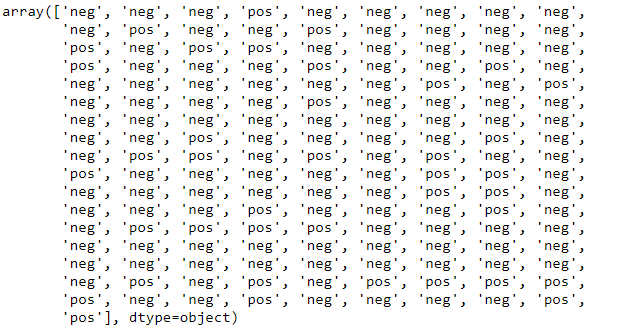
1. model2=KNeighborsClassifier(n\_neighbors=5,metric='euclidean')
2. model2.fit(x\_train,y\_train)



1. pred2=model2.predict(x\_test)
2. pred2



1. pr2=le.inverse\_transform(pred2)
2. pr2



1. usr1=[4,140,80,30,90,26.9,1.25,27]
2. usr1=np.array([usr1])
3. usr1=sc.fit\_transform(usr1)
4. le.inverse\_transform(model2.predict(usr1))



1. accuracy\_score(y\_test,pred2)



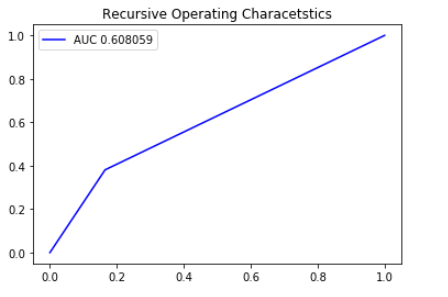
1. confusion\_matrix(y\_test,pred2)



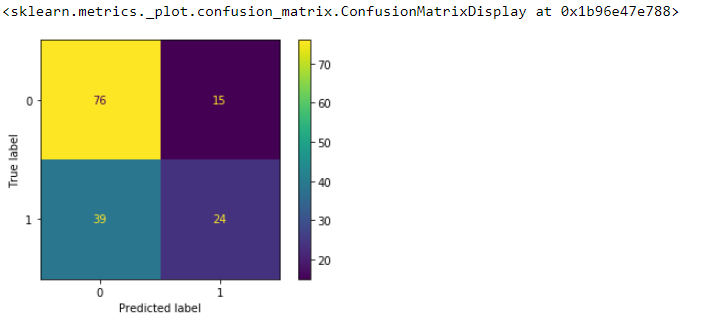
1. fpr,tpr,threshold=roc\_curve(y\_test,pred2)
2. roc\_auc=auc(fpr,tpr)
3. roc\_auc



1. plt.title('Recursive Operating Characetstics')
2. plt.plot(fpr,tpr,'b',label='AUC %f'%roc\_auc)
3. plt.legend()
4. plt.show()



1. plot\_confusion\_matrix(model2,x\_test,y\_test)



1. report1=classification\_report(y\_test,pred2)
2. print(report1)

