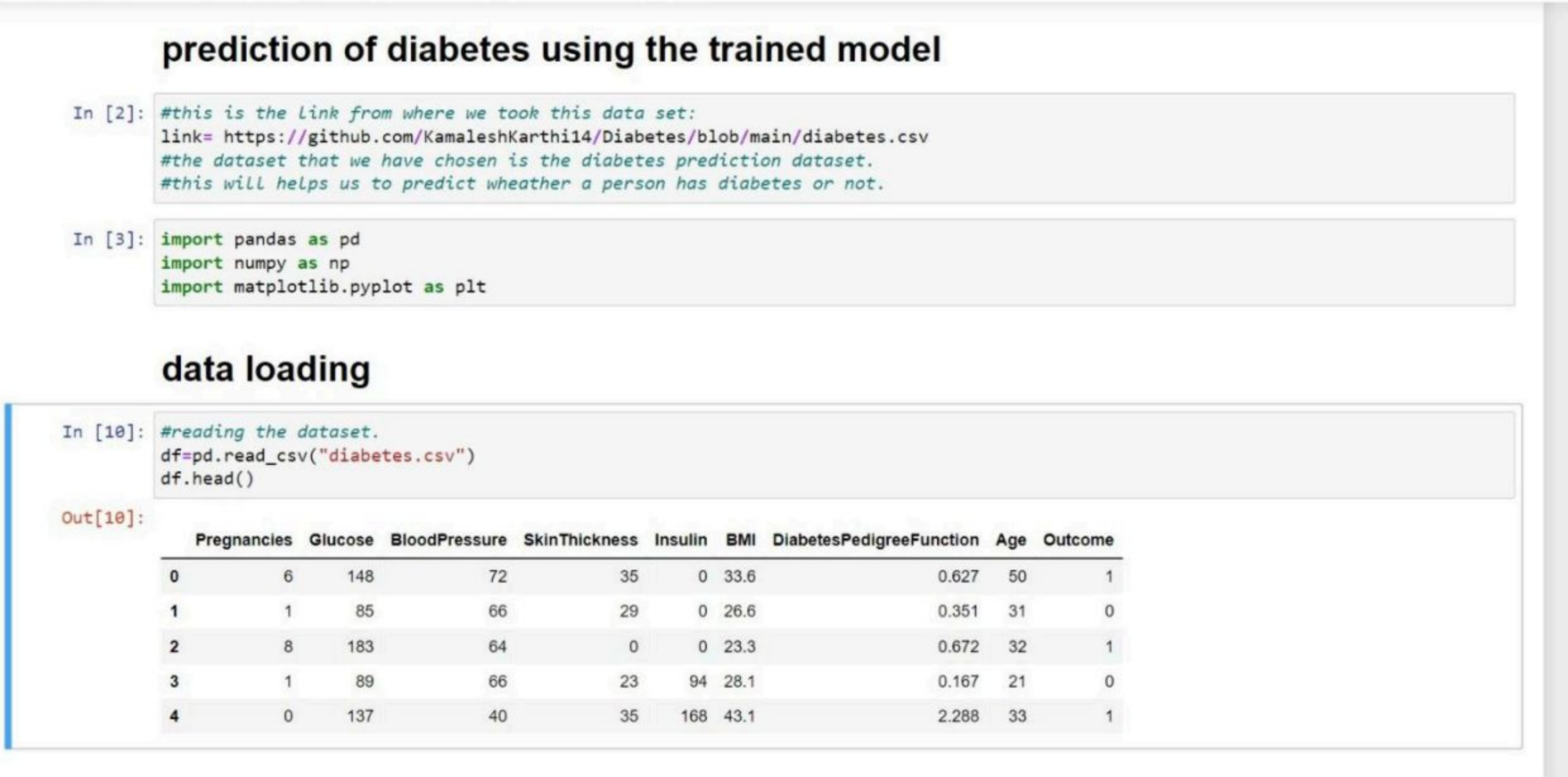


prediction of diabetes using the trained model

```
In [2]: #this is the link from where we took this data set:
        link= https://github.com/KamaleshKarthi14/Diabetes/blob/main/diabetes.csv
        #the dataset that we have chosen is the diabetes prediction dataset.
        #this will helps us to predict wheather a person has diabetes or not.
```

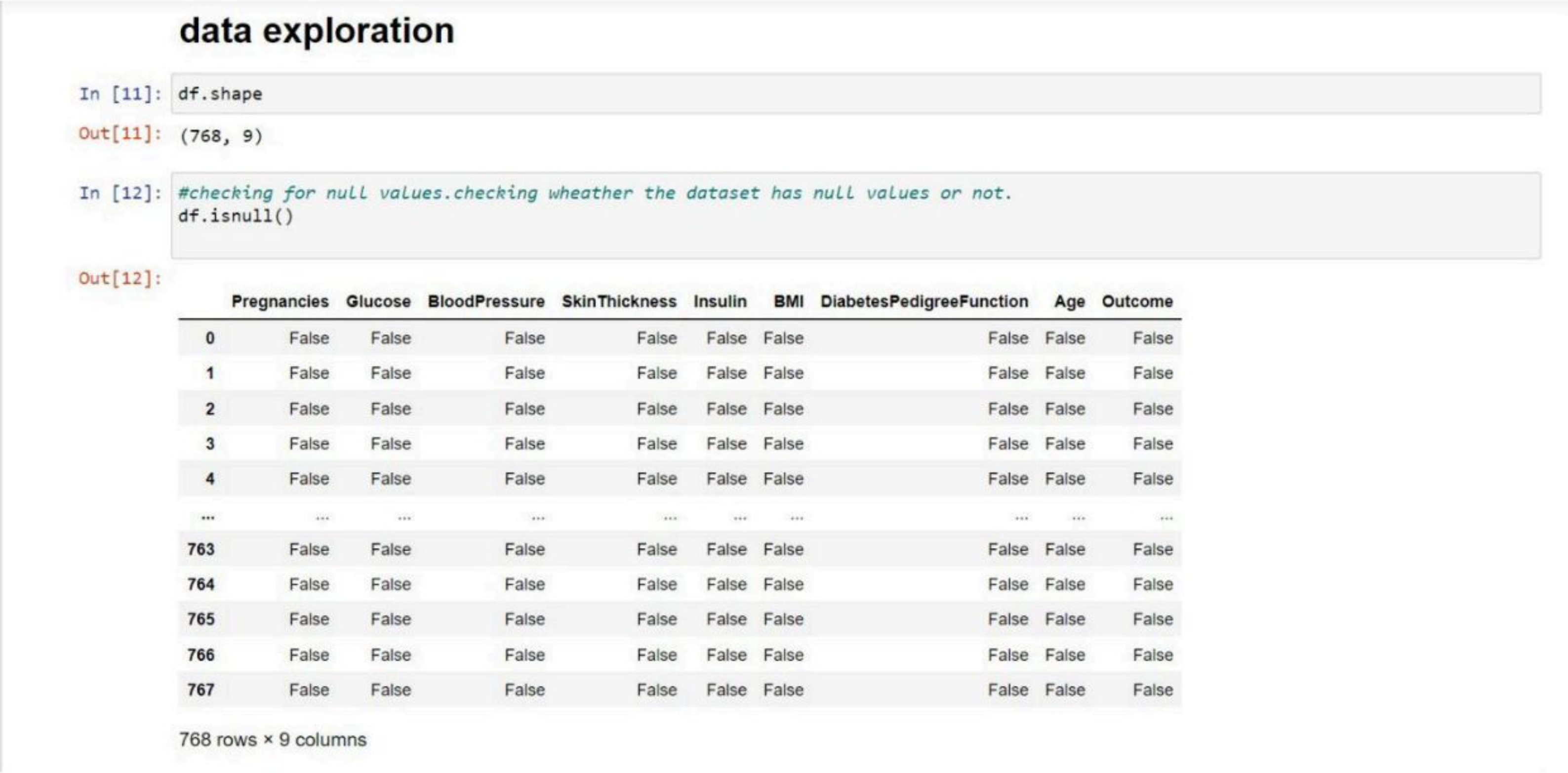
```
In [3]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

data loading

```
In [10]: #reading the dataset.
         df=pd.read_csv("diabetes.csv")
         df.head()
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1



data exploration

```
In [11]: df.shape
```

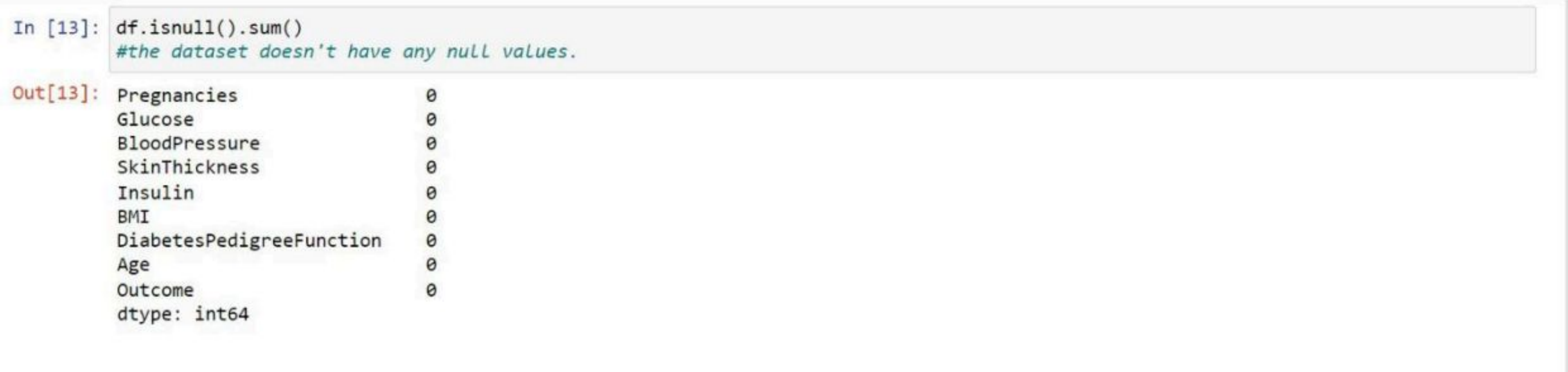
```
Out[11]: (768, 9)
```

```
In [12]: #checking for null values.checking wheather the dataset has null values or not.
         df.isnull()
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
763	False	False	False	False	False	False	False	False	False
764	False	False	False	False	False	False	False	False	False
765	False	False	False	False	False	False	False	False	False
766	False	False	False	False	False	False	False	False	False
767	False	False	False	False	False	False	False	False	False

768 rows × 9 columns

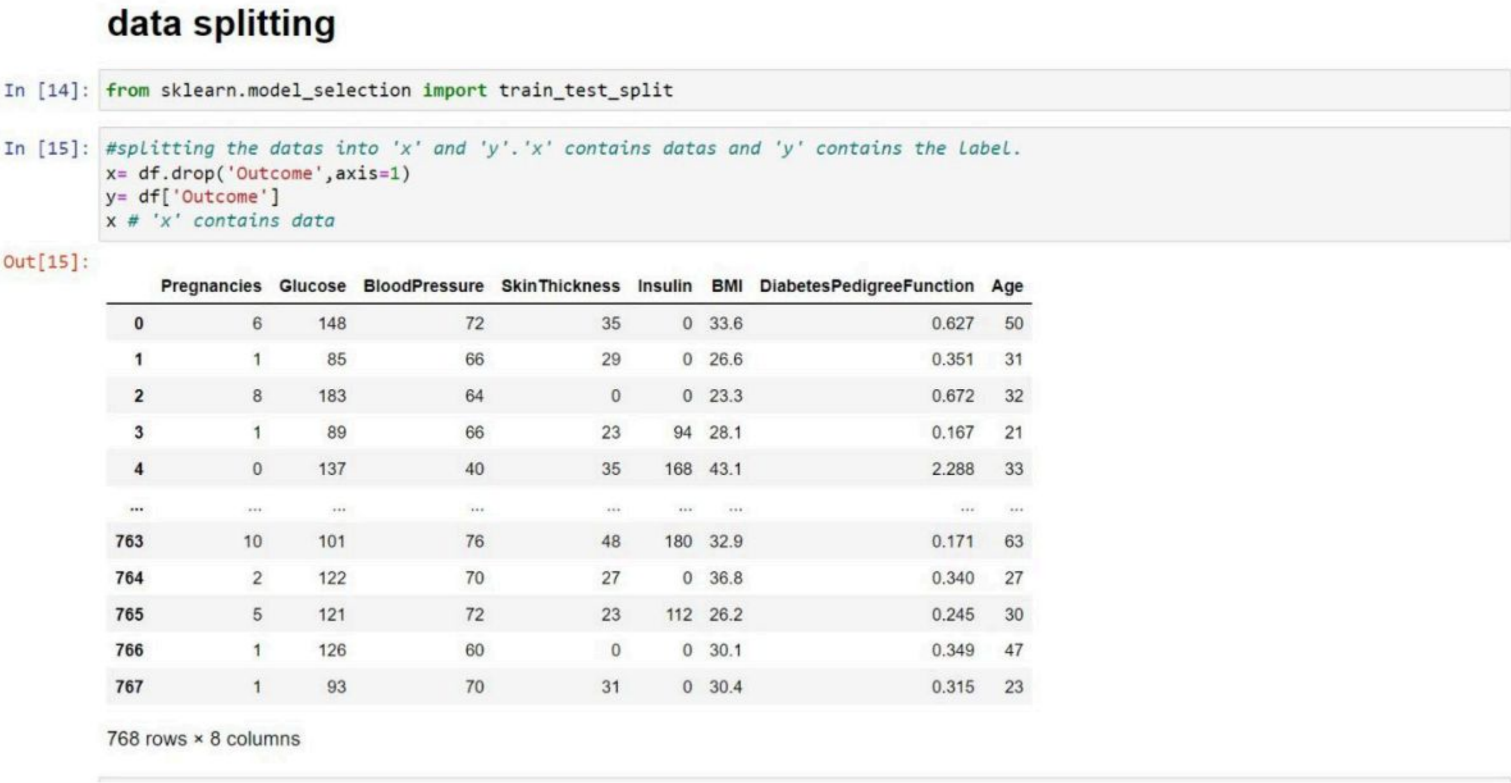


In [13]:

```
df.isnull().sum()
#the dataset doesn't have any null values.
```

Out[13]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```



data splitting

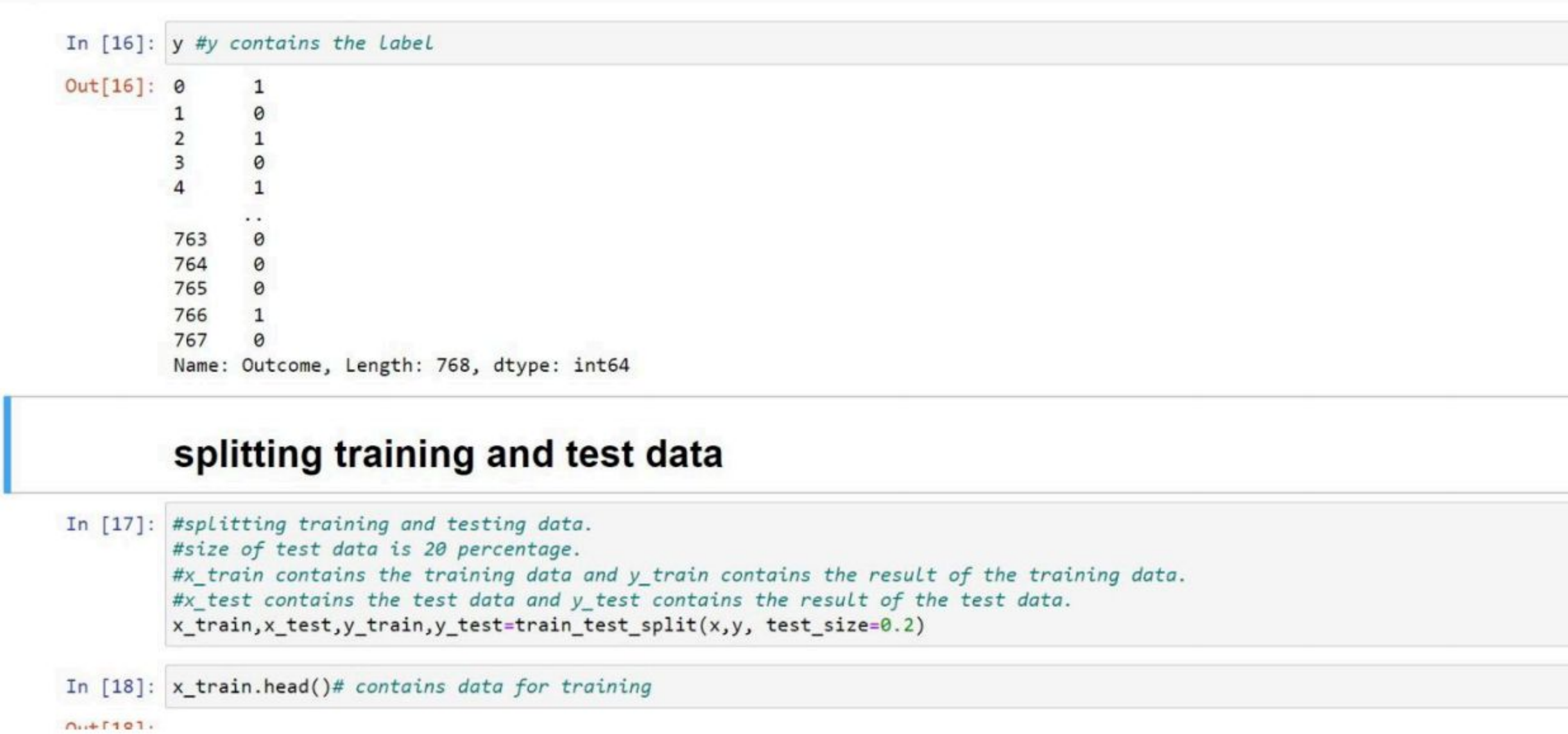
In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
#splitting the datas into 'x' and 'y'.'x' contains datas and 'y' contains the label.
x= df.drop('Outcome',axis=1)
y= df['Outcome']
x # 'x' contains data
```

Out[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

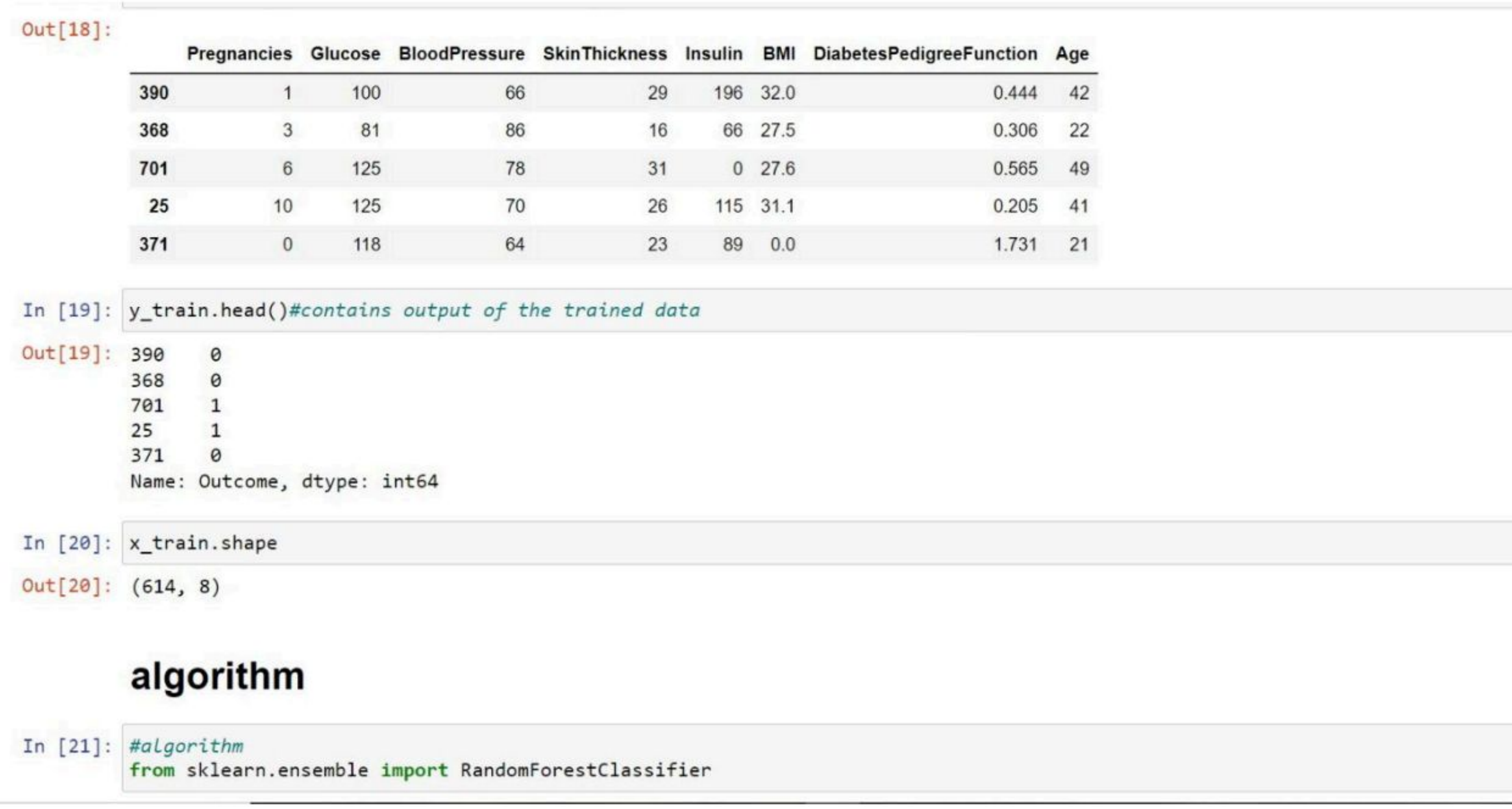
768 rows × 8 columns



```
In [16]: y #y contains the label

Out[16]: 0      1
         1      0
         2      1
         3      0
         4      1
               ..
         763    0
         764    0
         765    0
         766    1
         767    0
         Name: Outcome, Length: 768, dtype: int64
```

splitting training and test data

```
In [17]: #splitting training and testing data.
         #size of test data is 20 percentage.
         #x_train contains the training data and y_train contains the result of the training data.
         #x_test contains the test data and y_test contains the result of the test data.
         x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2)

In [18]: x_train.head()# contains data for training

Out[18]:
```



Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
390	1	100	66	29	196	32.0	0.444	42
368	3	81	86	16	66	27.5	0.306	22
701	6	125	78	31	0	27.6	0.565	49
25	10	125	70	26	115	31.1	0.205	41
371	0	118	64	23	89	0.0	1.731	21

In [19]:

```
y_train.head()#contains output of the trained data
```

Out[19]:

```
390    0
368    0
701    1
25     1
371    0
Name: Outcome, dtype: int64
```

In [20]:

```
x_train.shape
```

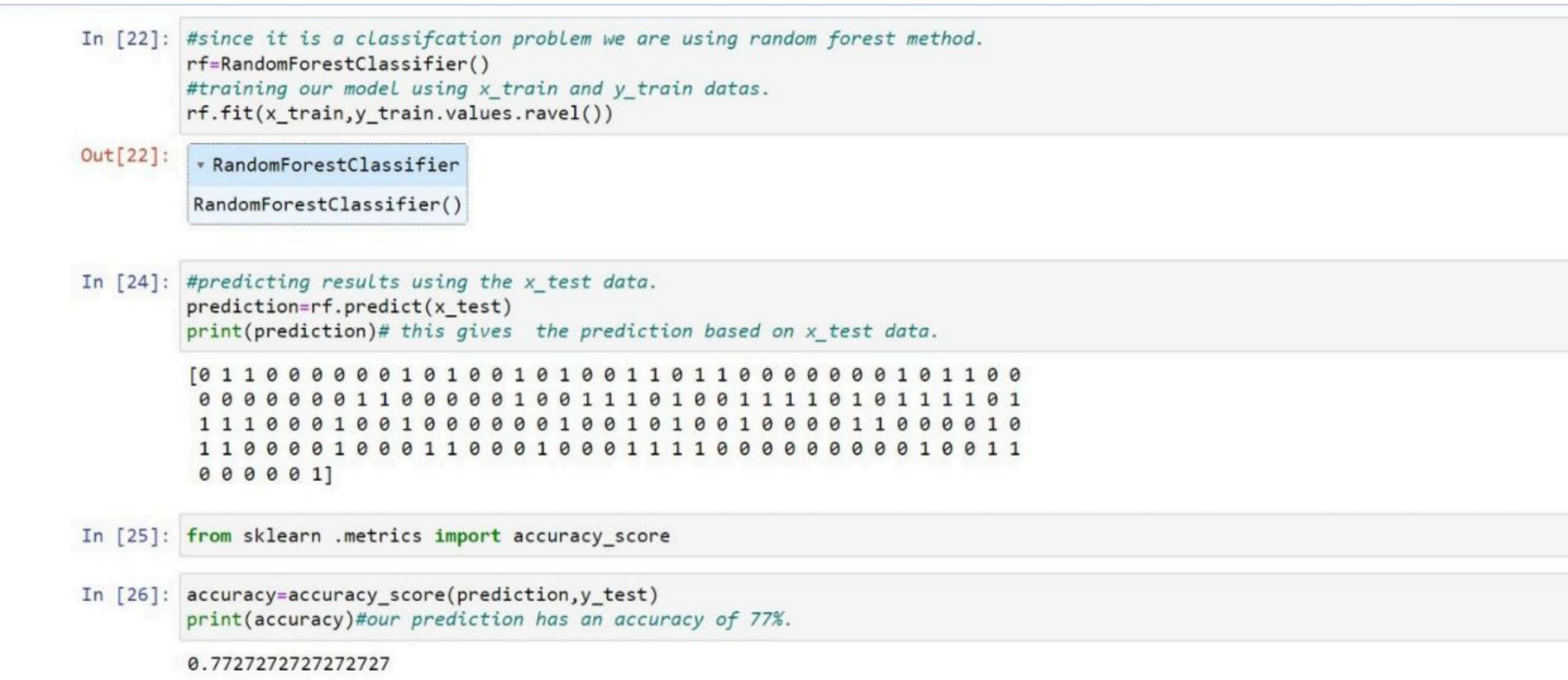
Out[20]:

```
(614, 8)
```

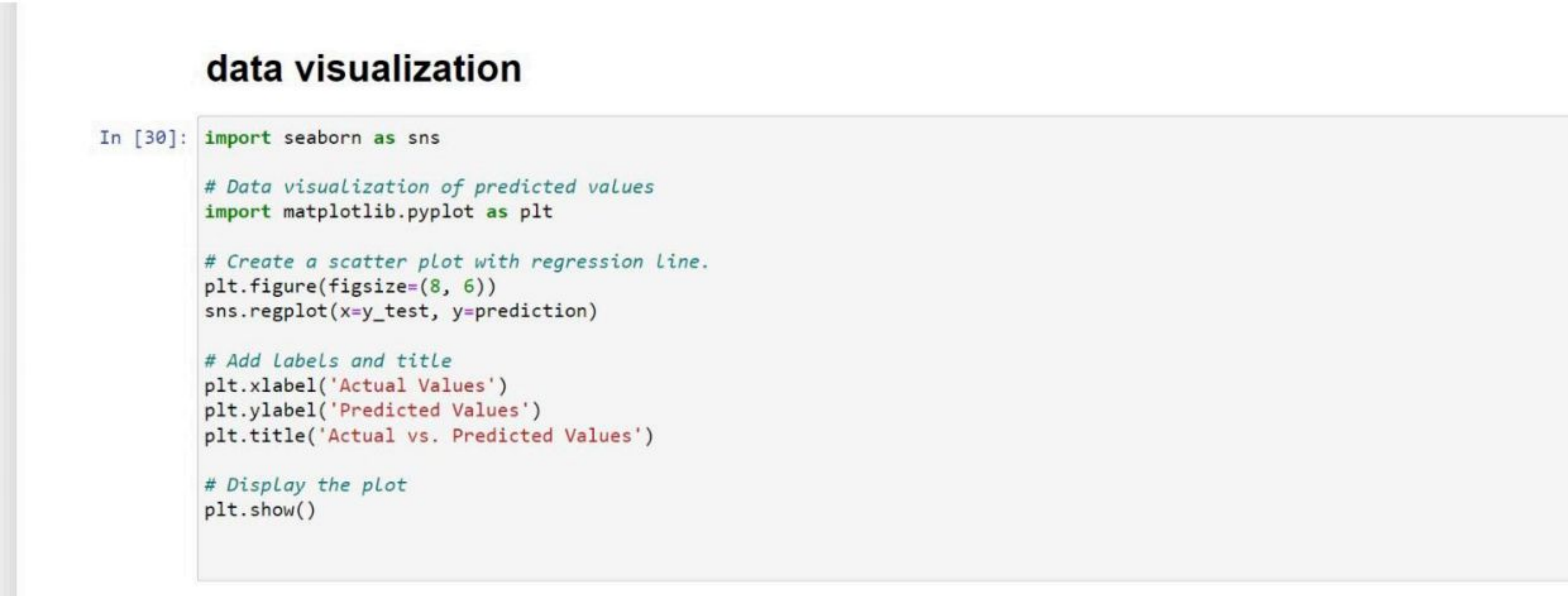
algorithm

In [21]:

```
#algorithm
from sklearn.ensemble import RandomForestClassifier
```



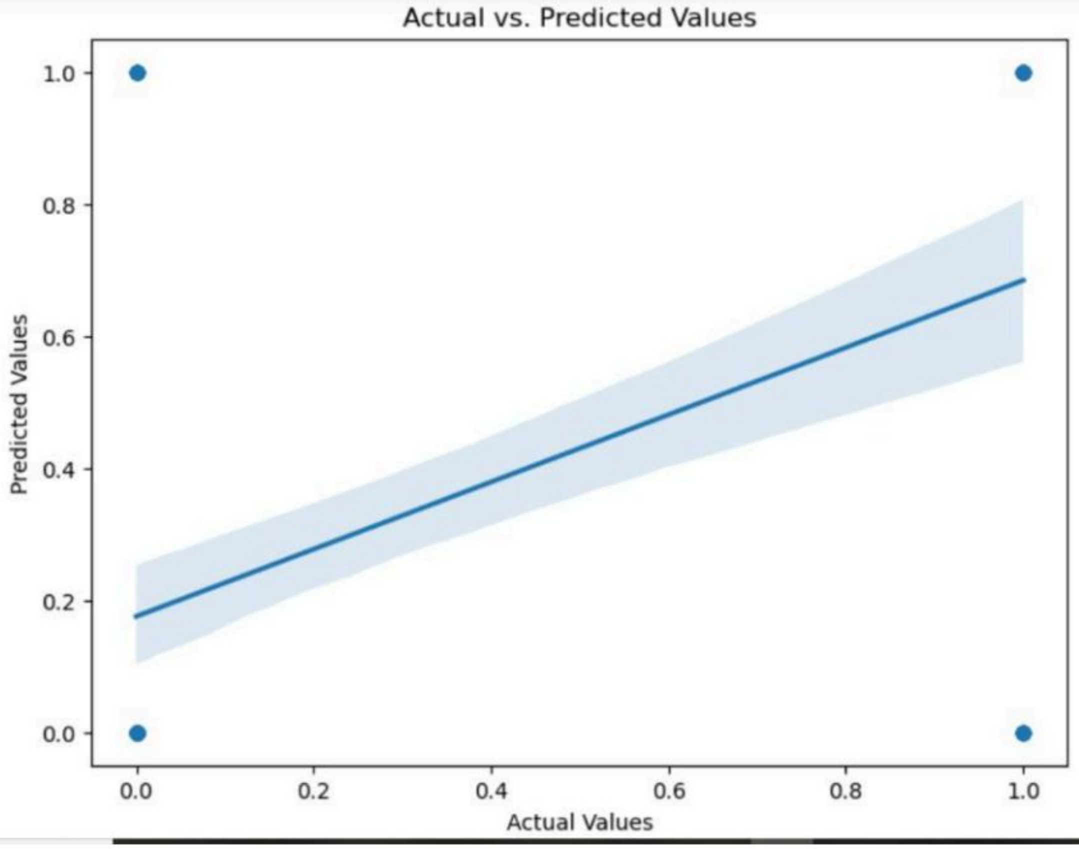
```
In [22]: #since it is a classifcation problem we are using random forest method.
         rf=RandomForestClassifier()
         #training our model using x_train and y_train datas.
         rf.fit(x_train,y_train.values.ravel())

Out[22]: ▾ RandomForestClassifier
         RandomForestClassifier()

In [24]: #predicting results using the x_test data.
         prediction=rf.predict(x_test)
         print(prediction)# this gives  the prediction based on x_test data.

         [0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 1 0 1 1 0 0
          0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 1
          1 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0
          1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1
          0 0 0 0 0 1]

In [25]: from sklearn .metrics import accuracy_score

In [26]: accuracy=accuracy_score(prediction,y_test)
         print(accuracy)#our prediction has an accuracy of 77%.

         0.7727272727272727
```

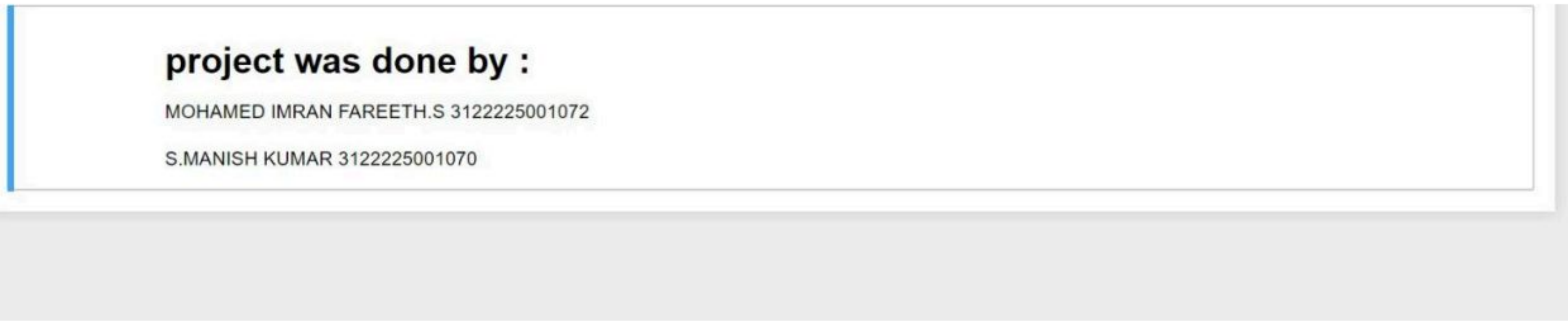


data visualization

In [30]:

```
import seaborn as sns

# Data visualization of predicted values
import matplotlib.pyplot as plt

# Create a scatter plot with regression line.
plt.figure(figsize=(8, 6))
sns.regplot(x=y_test, y=prediction)

# Add labels and title
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')

# Display the plot
plt.show()
```


Actual vs. Predicted Values
Predicted Values
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Actual Values



project was done by :

MOHAMED IMRAN FAREETH.S 3122225001072

S.MANISH KUMAR 3122225001070