ARIN DEV

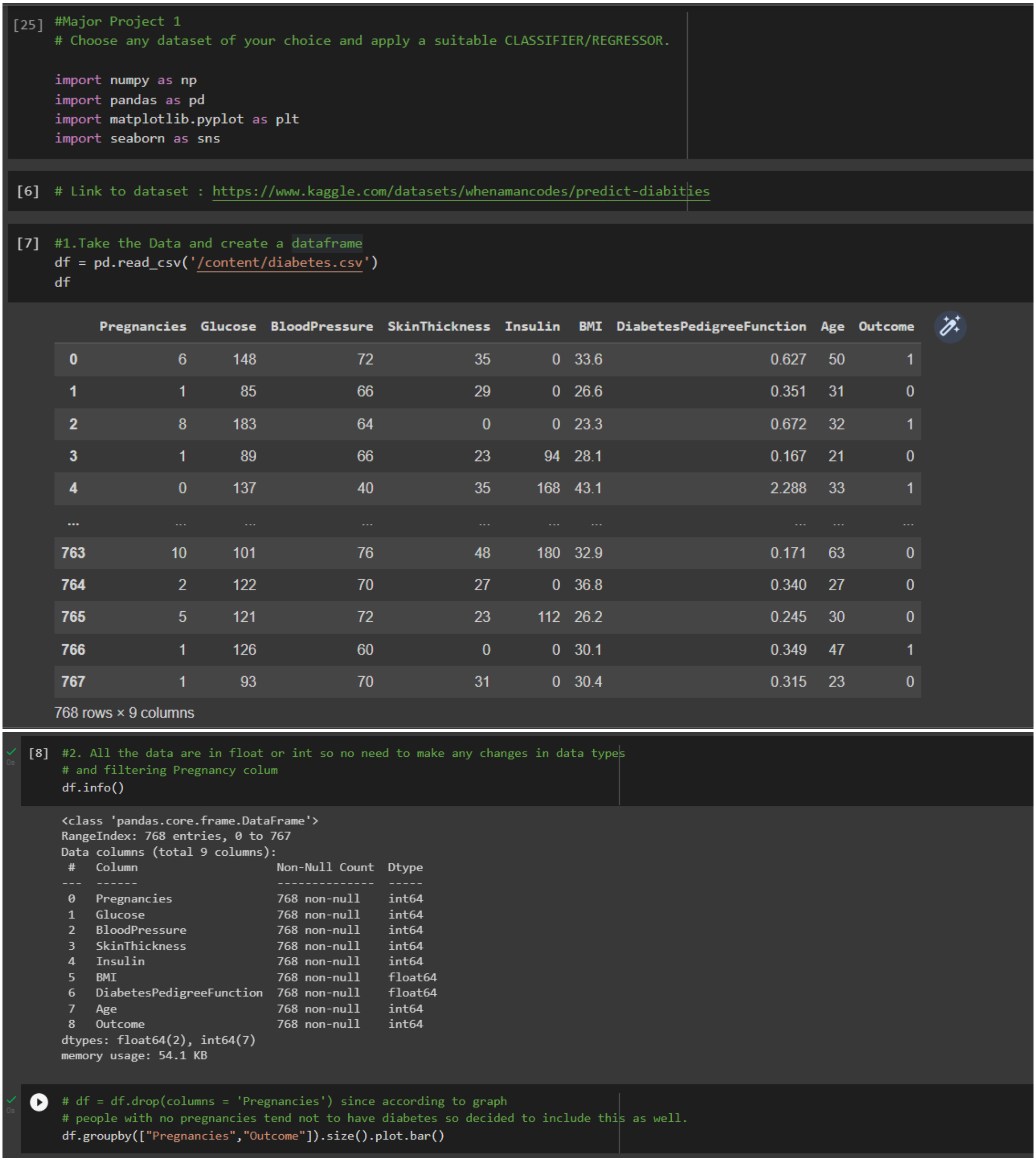
IIT BHUBANESWAR

Second Year, B.Tech (ECE)

Contact : arindev30@gmail.com

21ec01048@gmail.com

Major Project 1



```
[25] #Major Project 1
     # Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR.

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

```
[6]  # Link to dataset : https://www.kaggle.com/datasets/whenamancodes/predict-diabities
```

```
[7]  #1.Take the Data and create a dataframe
     df = pd.read_csv('/content/diabetes.csv')
     df
```

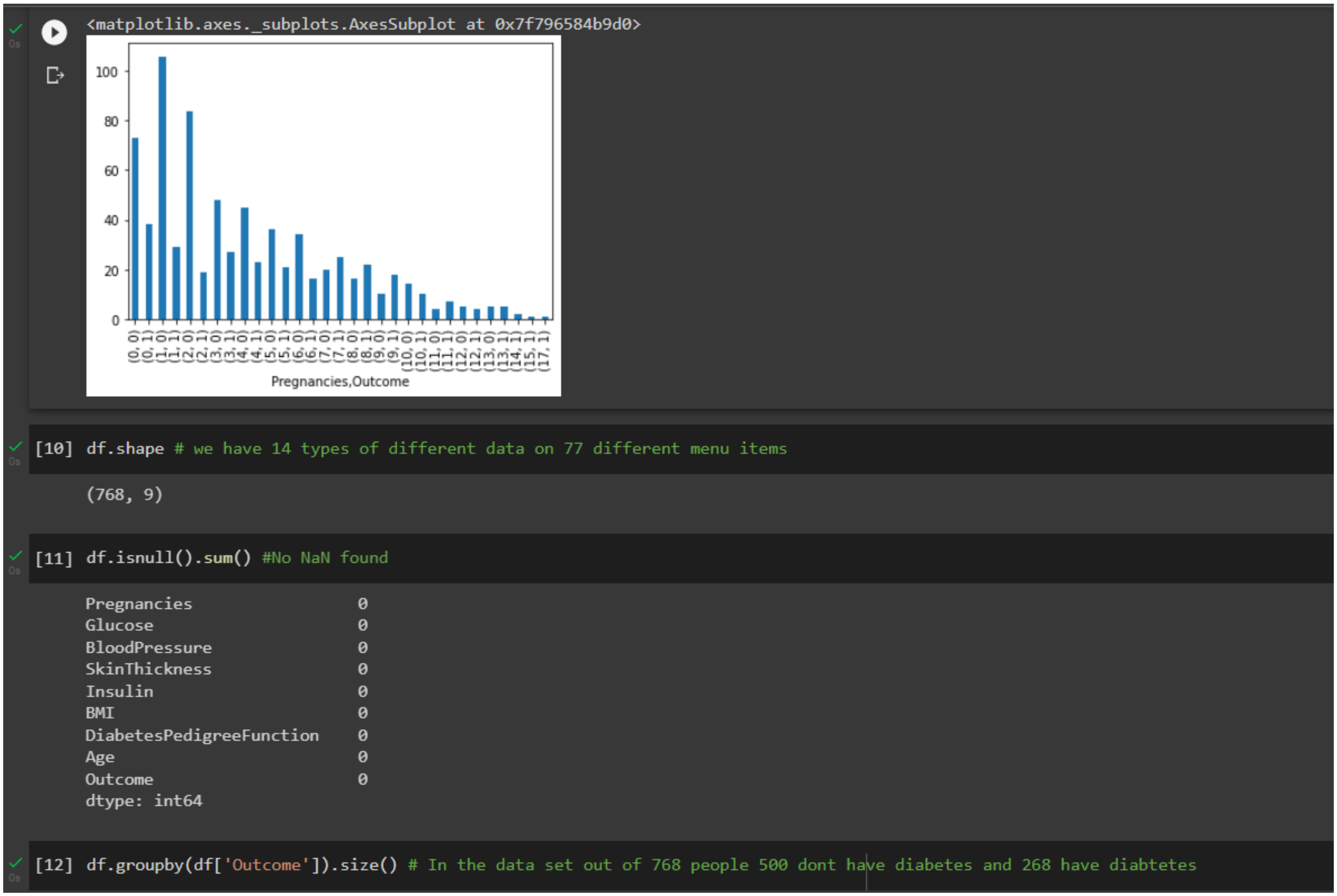
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763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

768 rows × 9 columns

```
[8]  #2. All the data are in float or int so no need to make any changes in data types
     # and filtering Pregnancy colum
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
# df = df.drop(columns = 'Pregnancies') since according to graph
# peoplee with no pregnancies tend not to have diabetes so decided to include this as well.
df.groupby(["Pregnancies","Outcome"]).size().plot.bar()
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f796584b9d0>
```

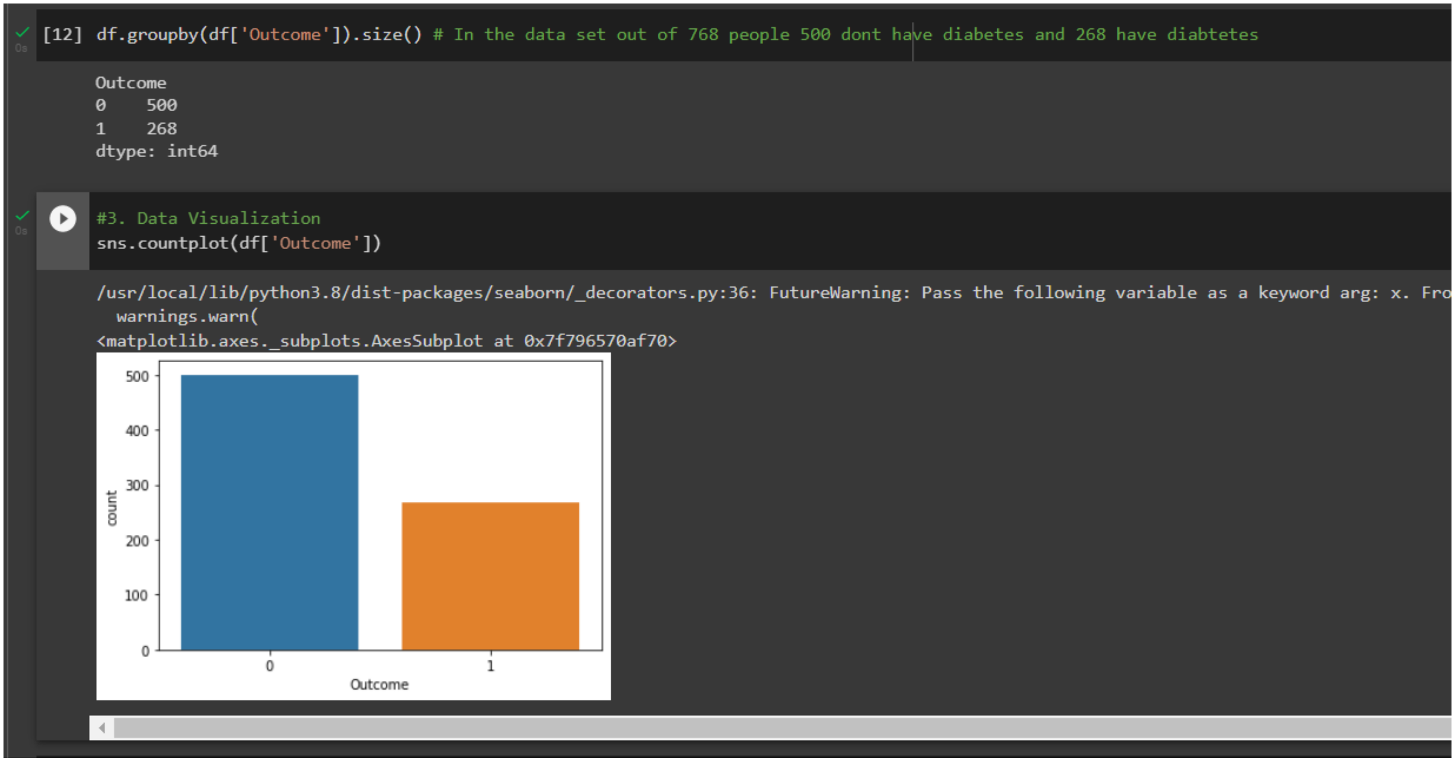
```
[10] df.shape # we have 14 types of different data on 77 different menu items
```

```
(768, 9)
```

```
[11] df.isnull().sum() #No NaN found
```

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

```
[12] df.groupby(df['Outcome']).size() # In the data set out of 768 people 500 dont have diabetes and 268 have diabtetes
```

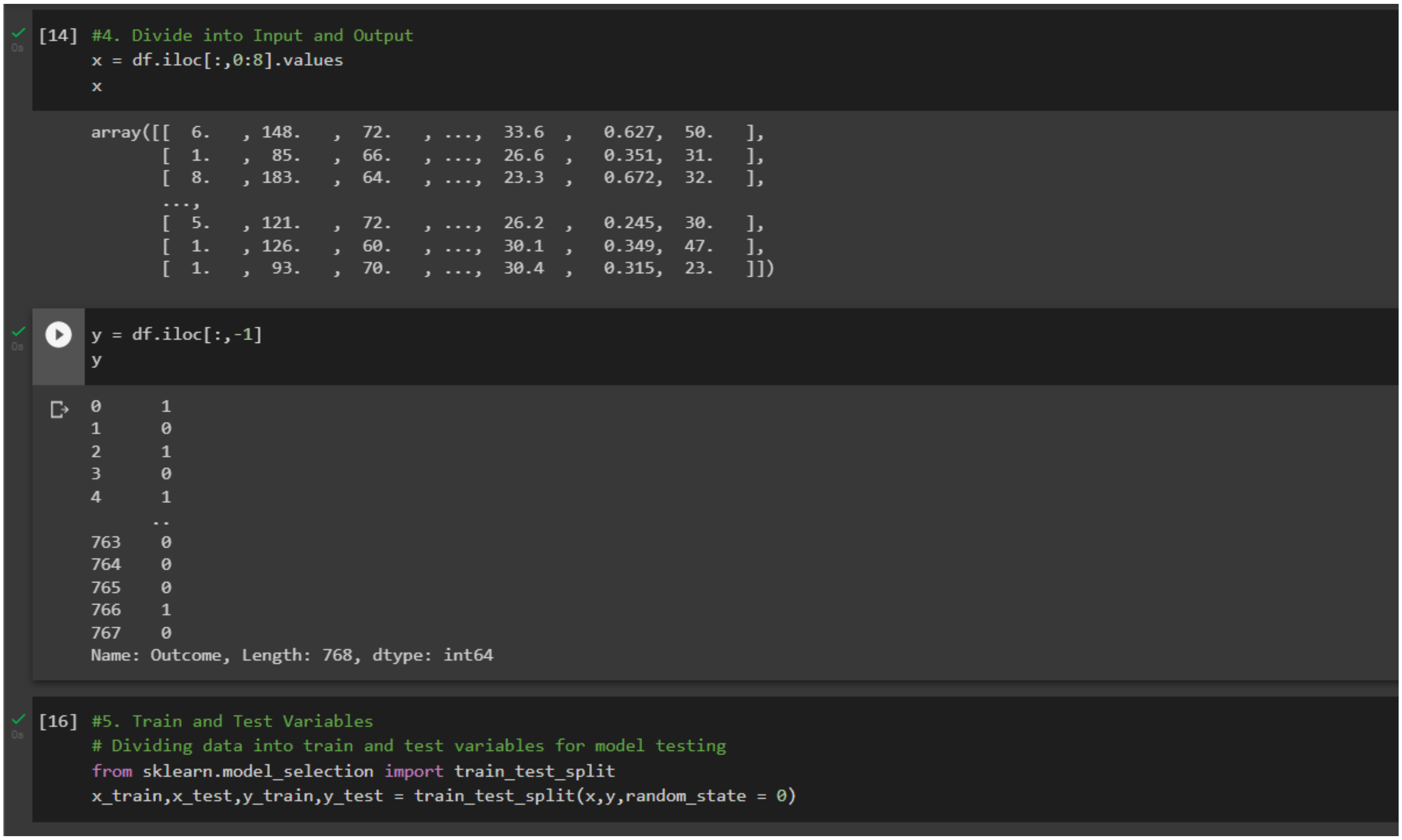


```
[12] df.groupby(df['Outcome']).size() # In the data set out of 768 people 500 dont have diabetes and 268 have diabtetes
```

```
Outcome
0    500
1    268
dtype: int64
```

```
#3. Data Visualization
sns.countplot(df['Outcome'])
```

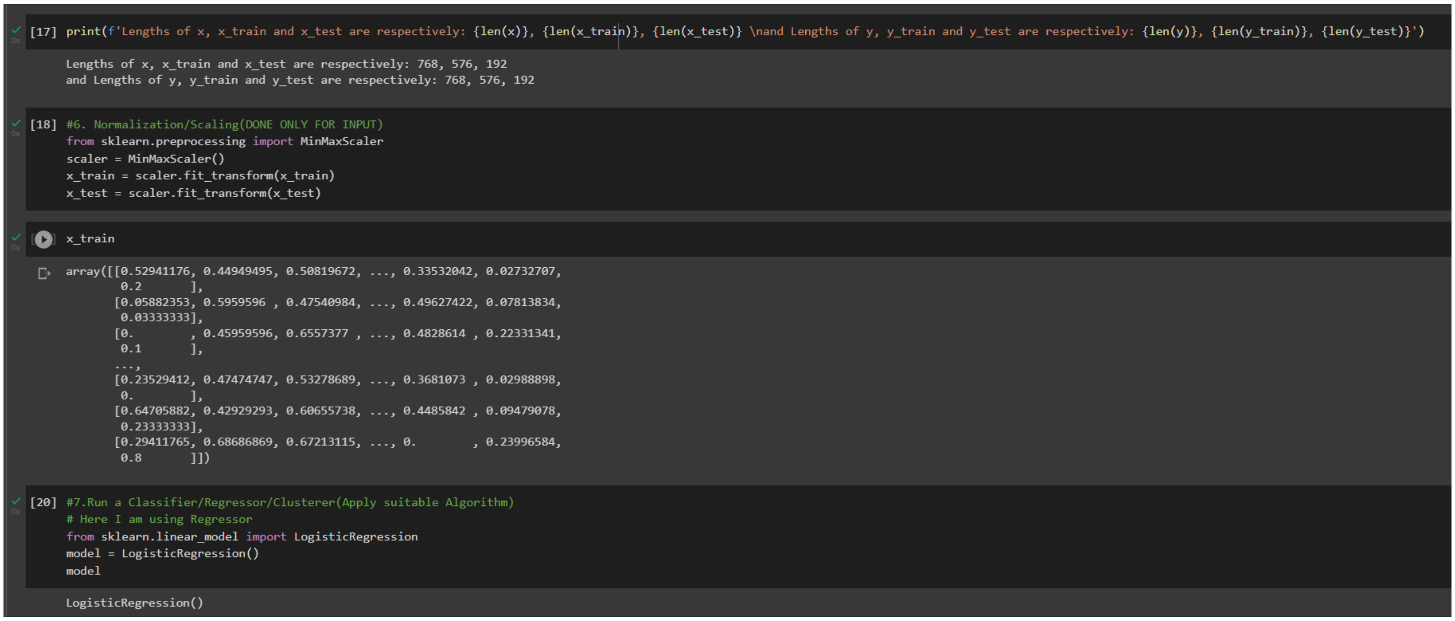
```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. Fro
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f796570af70>
```



```
[14] #4. Divide into Input and Output
     x = df.iloc[:,0:8].values
     x

     array([[  6.   , 148.   ,  72.   , ...,  33.6  ,   0.627,  50.   ],
            [  1.   ,  85.   ,  66.   , ...,  26.6  ,   0.351,  31.   ],
            [  8.   , 183.   ,  64.   , ...,  23.3  ,   0.672,  32.   ],
            ...,
            [  5.   , 121.   ,  72.   , ...,  26.2  ,   0.245,  30.   ],
            [  1.   , 126.   ,  60.   , ...,  30.1  ,   0.349,  47.   ],
            [  1.   ,  93.   ,  70.   , ...,  30.4  ,   0.315,  23.   ]])
```

```
     y = df.iloc[:,-1]
     y

     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
```

```
[16] #5. Train and Test Variables
     # Dividing data into train and test variables for model testing
     from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0)
```

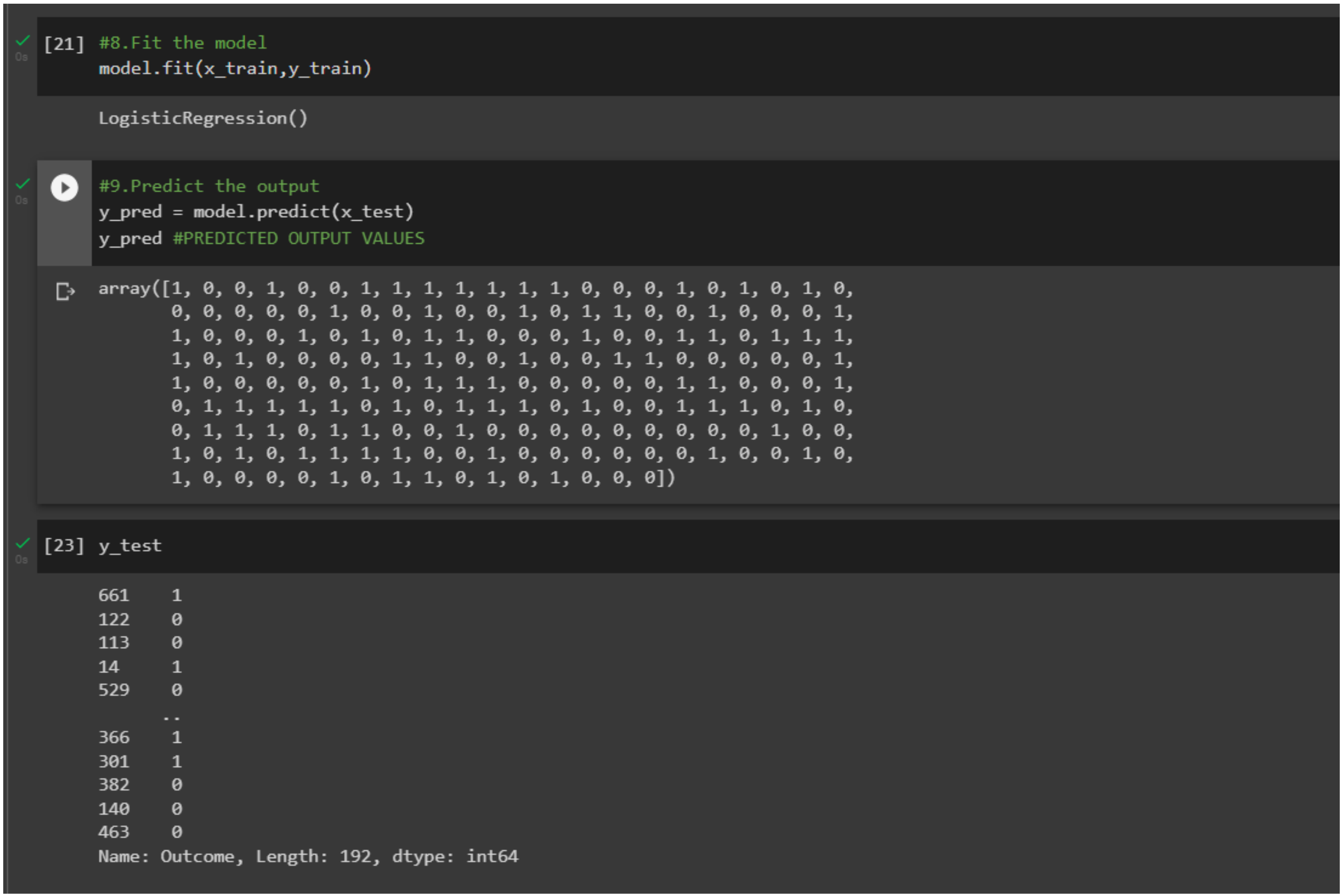


```
[17] print(f'Lengths of x, x_train and x_test are respectively: {len(x)}, {len(x_train)}, {len(x_test)} \nand Lengths of y, y_train and y_test are respectively: {len(y)}, {len(y_train)}, {len(y_test)}')

     Lengths of x, x_train and x_test are respectively: 768, 576, 192
     and Lengths of y, y_train and y_test are respectively: 768, 576, 192
```

```
[18] #6. Normalization/Scaling(DONE ONLY FOR INPUT)
     from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     x_train = scaler.fit_transform(x_train)
     x_test = scaler.fit_transform(x_test)
```

```
     x_train

     array([[0.52941176, 0.44949495, 0.50819672, ..., 0.33532042, 0.02732707,
             0.2       ],
            [0.05882353, 0.5959596 , 0.47540984, ..., 0.49627422, 0.07813834,
             0.03333333],
            [0.        , 0.45959596, 0.6557377 , ..., 0.4828614 , 0.22331341,
             0.1       ],
            ...,
            [0.23529412, 0.47474747, 0.53278689, ..., 0.3681073 , 0.02988898,
             0.        ],
            [0.64705882, 0.42929293, 0.60655738, ..., 0.4485842 , 0.09479078,
             0.23333333],
            [0.29411765, 0.68686869, 0.67213115, ..., 0.        , 0.23996584,
             0.8       ]])
```

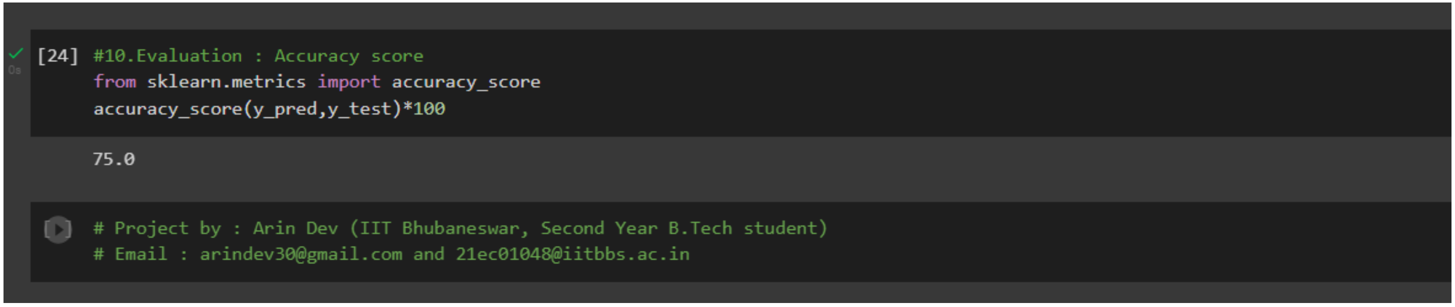
```
[20] #7.Run a Classifier/Regressor/Clusterer(Apply suitable Algorithm)
     # Here I am using Regressor
     from sklearn.linear_model import LogisticRegression
     model = LogisticRegression()
     model

     LogisticRegression()
```



```
[21] #8.Fit the model
     model.fit(x_train,y_train)

     LogisticRegression()
```

```
     #9.Predict the output
     y_pred = model.predict(x_test)
     y_pred #PREDICTED OUTPUT VALUES

     array([1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
            0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
            1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
            1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
            1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
            0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0,
            0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
            1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
            1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0])
```

```
[23] y_test

     661    1
     122    0
     113    0
     14     1
     529    0
           ..
     366    1
     301    1
     382    0
     140    0
     463    0
     Name: Outcome, Length: 192, dtype: int64
```



```
[24] #10.Evaluation : Accuracy score
     from sklearn.metrics import accuracy_score
     accuracy_score(y_pred,y_test)*100

     75.0
```

```
     # Project by : Arin Dev (IIT Bhubaneswar, Second Year B.Tech student)
     # Email : arindev30@gmail.com and 21ec01048@iitbbs.ac.in
```

Project by : Arin Dev (IIT Bhubaneswar, Second Year B.Tech student)
Email : arindev30@gmail.com and 21ec01048@iitbbs.ac.in

Link to this Notebook :
https://colab.research.google.com/drive/1JCJrnfjs5bGTXJz9K_o8QoY1dA3-P488?usp=sharing

Link to my Colab Notebooks :
https://drive.google.com/drive/folders/1WyxyVfRdAYuGWbqHI9O3rFCyMN0RSo1t?usp=sharing

Link to my GitHub : https://github.com/arin-dev/Rinex