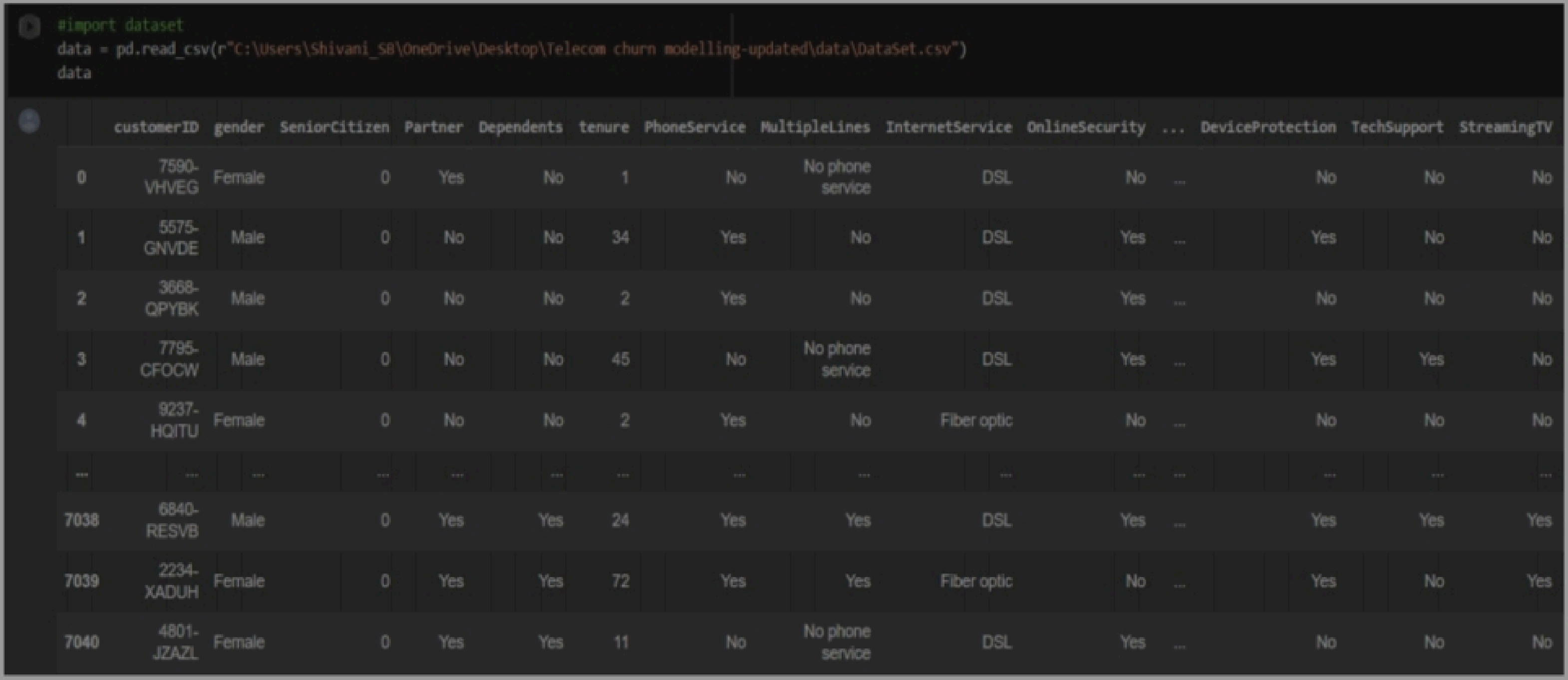
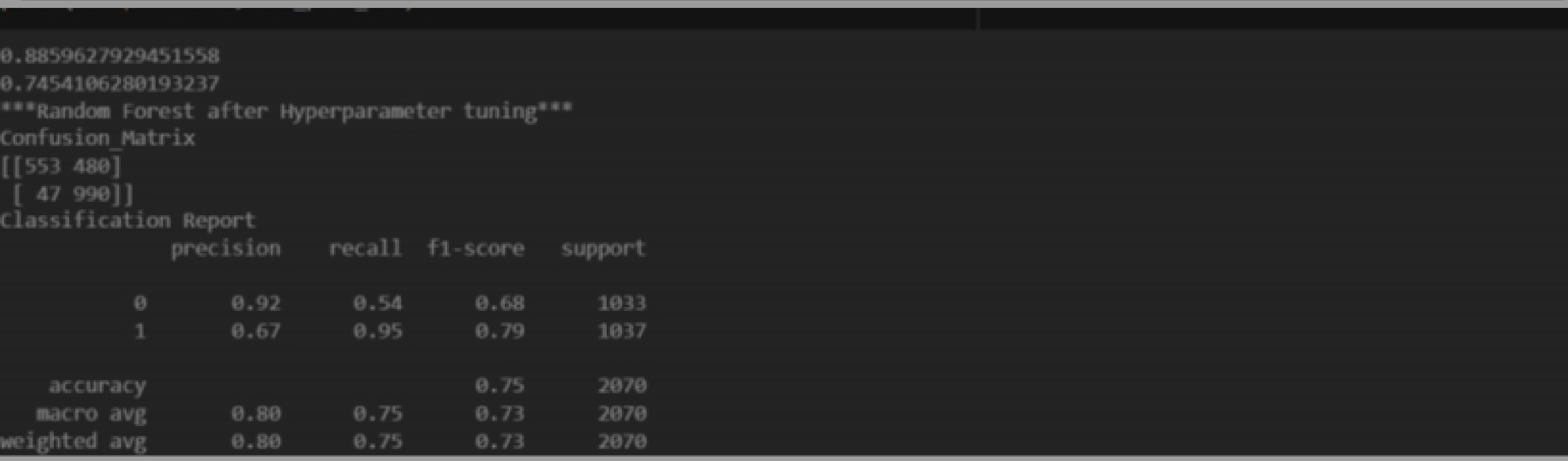
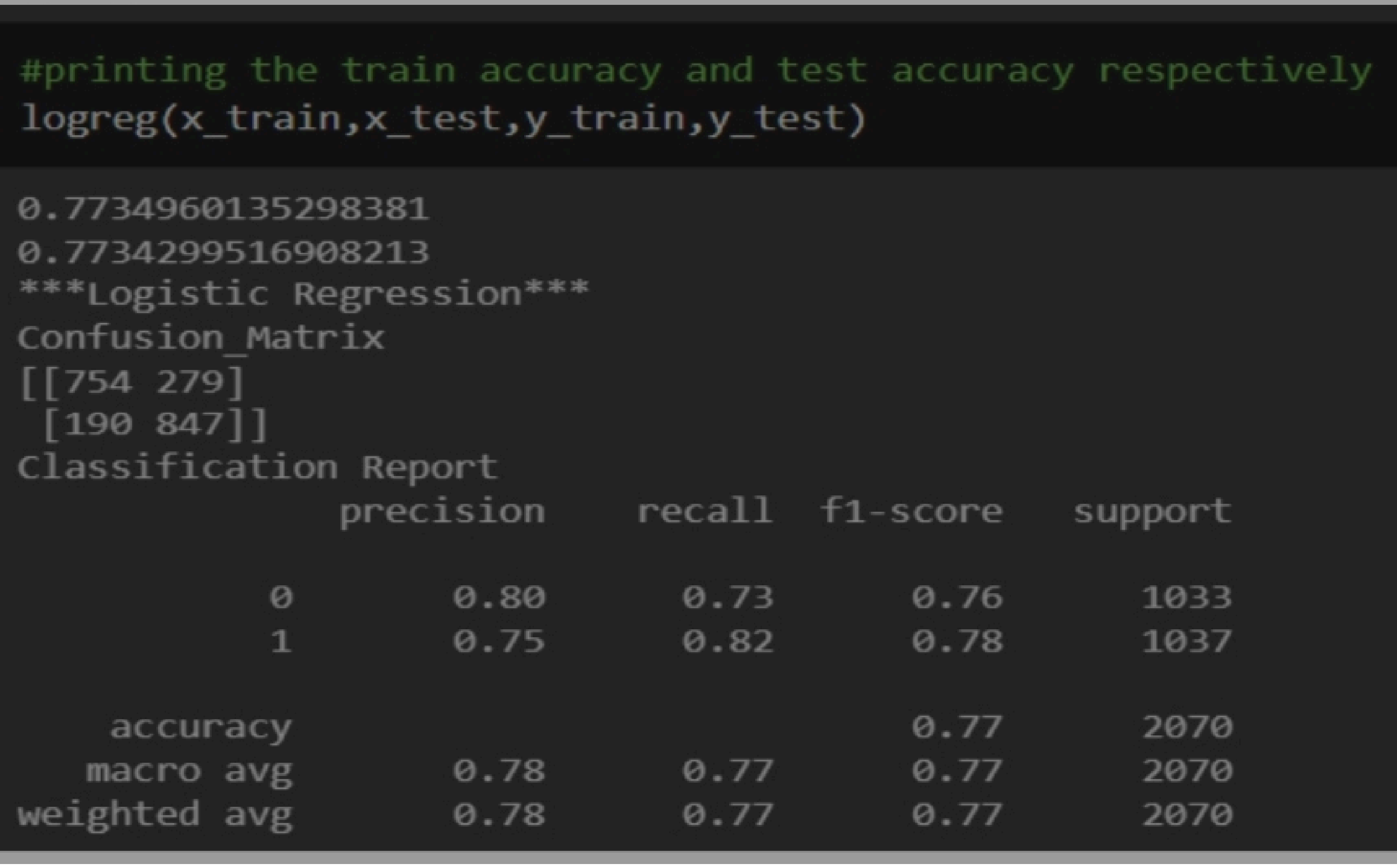


```
#import dataset
data = pd.read_csv(r"C:\Users\Shivani_SB\OneDrive\Desktop\Telecom churn modelling-updated\data\DataSet.csv")
data
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	Yes	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	No	Yes
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No	No	No

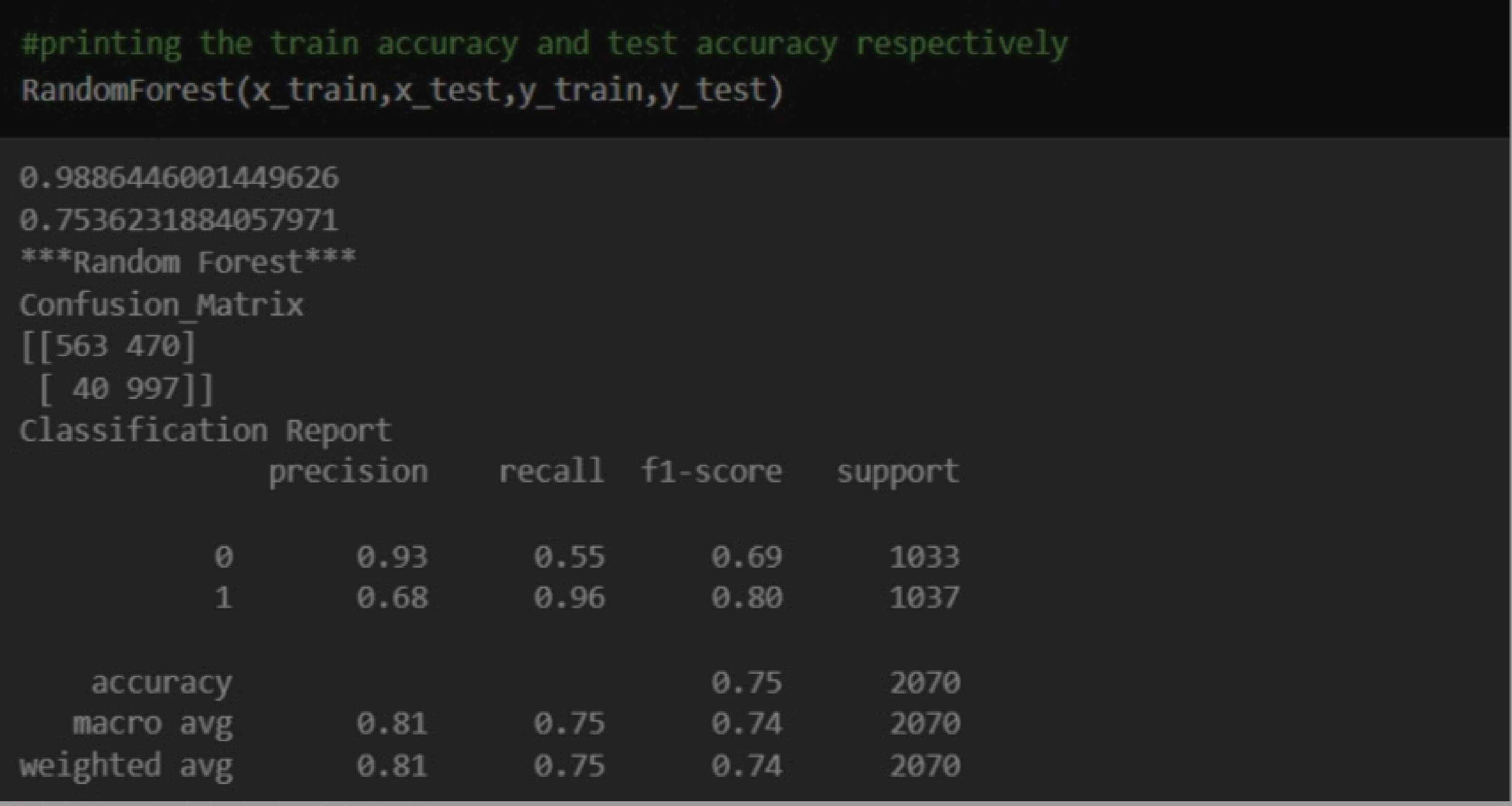


```
0.8859627929451558
0.7454106280193237
***Random Forest after Hyperparameter tuning***
Confusion_Matrix
[[553 480]
 [ 47 990]]
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.54      0.68      1033
           1       0.67      0.95      0.79      1037

    accuracy                           0.75      2070
   macro avg       0.80      0.75      0.73      2070
weighted avg       0.80      0.75      0.73      2070
```



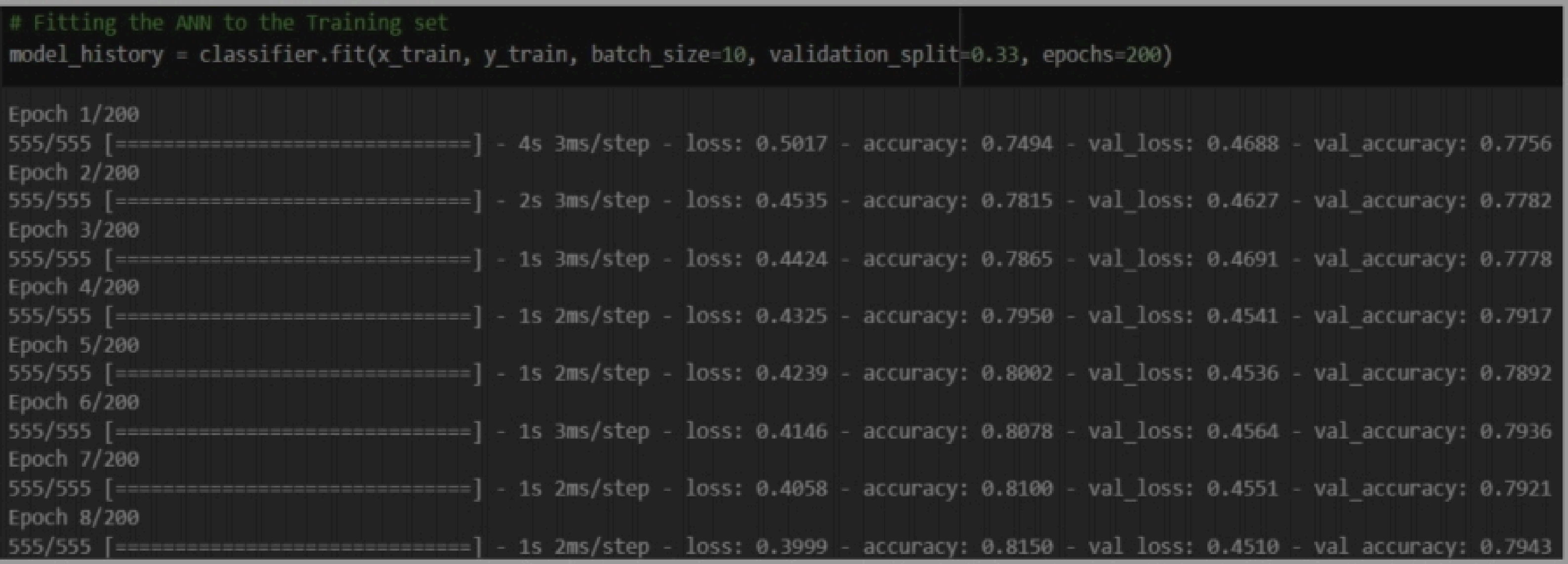
```
#printing the train accuracy and test accuracy respectively
logreg(x_train,x_test,y_train,y_test)
```

```
0.7734960135298381
0.7734299516908213
***Logistic Regression***
Confusion_Matrix
[[754 279]
 [190 847]]
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.73      0.76      1033
           1       0.75      0.82      0.78      1037

    accuracy                           0.77      2070
   macro avg       0.78      0.77      0.77      2070
weighted avg       0.78      0.77      0.77      2070
```



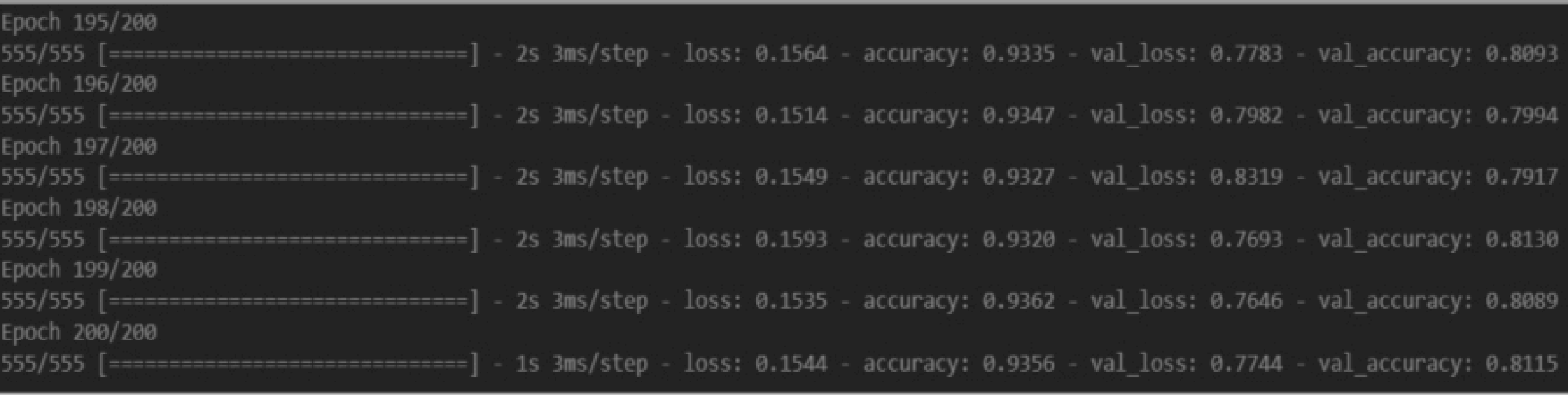
```
#printing the train accuracy and test accuracy respectively
RandomForest(x_train,x_test,y_train,y_test)
```

```
0.9886446001449626
0.7536231884057971
***Random Forest***
Confusion_Matrix
[[563 470]
 [ 40 997]]
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.55      0.69      1033
           1       0.68      0.96      0.80      1037

    accuracy                           0.75      2070
   macro avg       0.81      0.75      0.74      2070
weighted avg       0.81      0.75      0.74      2070
```

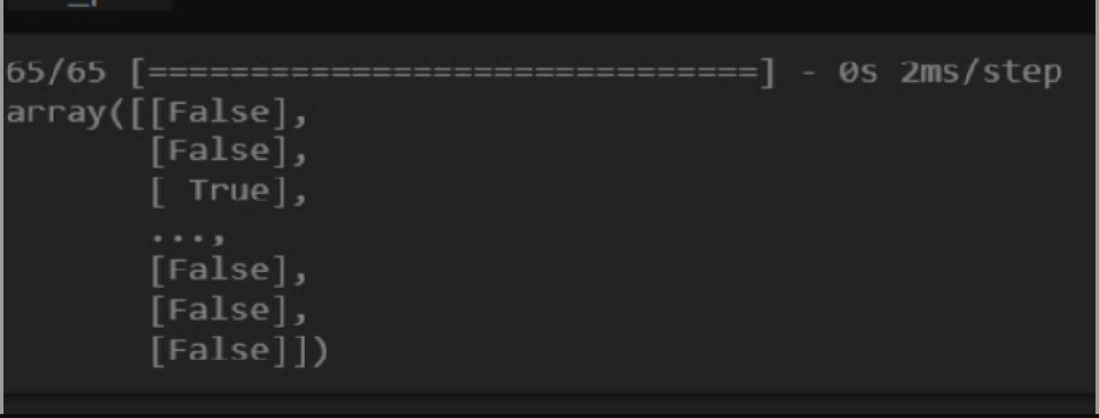


```
# Fitting the ANN to the Training set
model_history = classifier.fit(x_train, y_train, batch_size=10, validation_split=0.33, epochs=200)
```

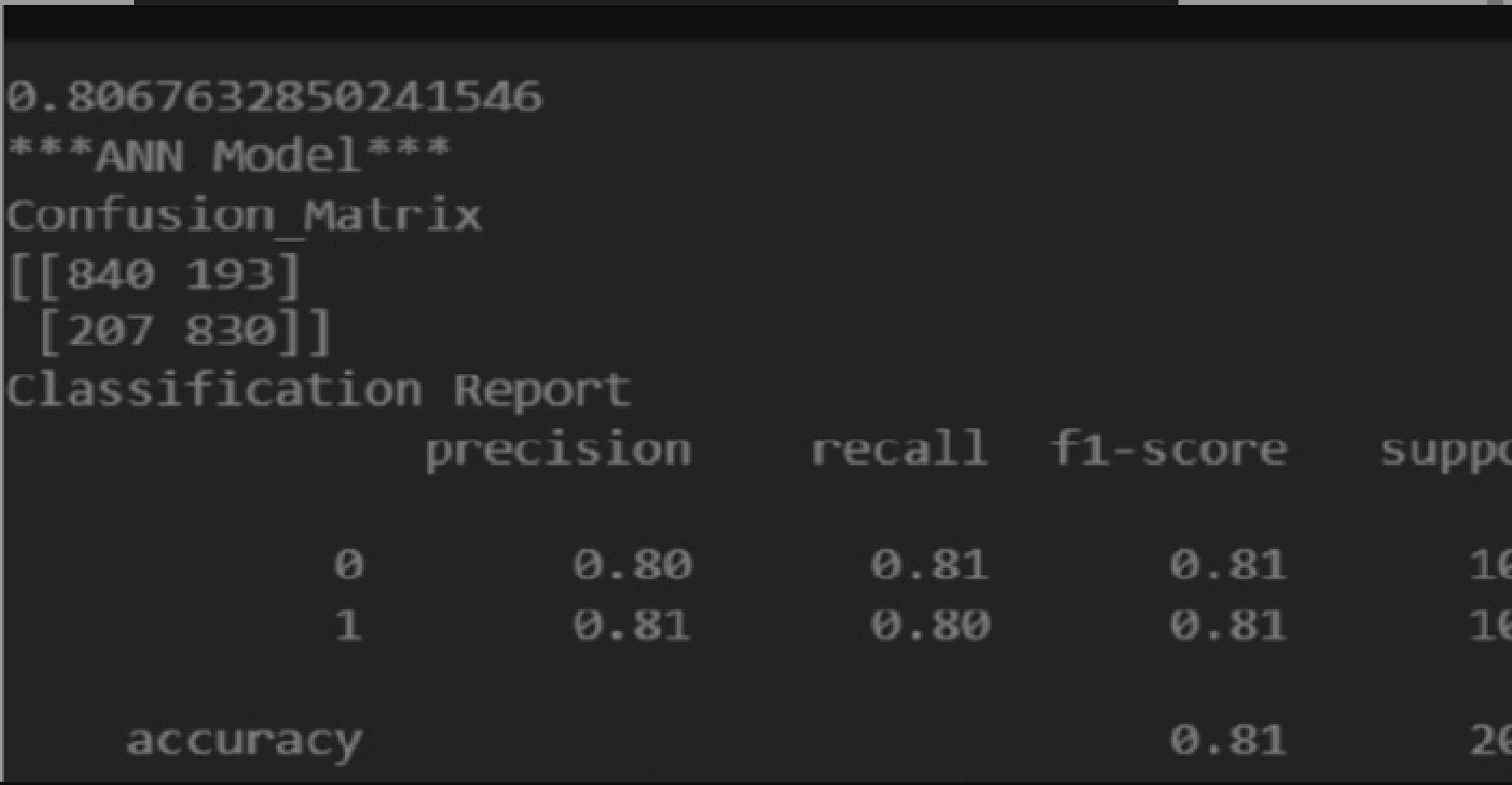
```
Epoch 1/200
555/555 [==============================] - 4s 3ms/step - loss: 0.5017 - accuracy: 0.7494 - val_loss: 0.4688 - val_accuracy: 0.7756
Epoch 2/200
555/555 [==============================] - 2s 3ms/step - loss: 0.4535 - accuracy: 0.7815 - val_loss: 0.4627 - val_accuracy: 0.7782
Epoch 3/200
555/555 [==============================] - 1s 3ms/step - loss: 0.4424 - accuracy: 0.7865 - val_loss: 0.4691 - val_accuracy: 0.7778
Epoch 4/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4325 - accuracy: 0.7950 - val_loss: 0.4541 - val_accuracy: 0.7917
Epoch 5/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4239 - accuracy: 0.8002 - val_loss: 0.4536 - val_accuracy: 0.7892
Epoch 6/200
555/555 [==============================] - 1s 3ms/step - loss: 0.4146 - accuracy: 0.8078 - val_loss: 0.4564 - val_accuracy: 0.7936
Epoch 7/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4058 - accuracy: 0.8100 - val_loss: 0.4551 - val_accuracy: 0.7921
Epoch 8/200
555/555 [==============================] - 1s 2ms/step - loss: 0.3999 - accuracy: 0.8150 - val_loss: 0.4510 - val_accuracy: 0.7943
```



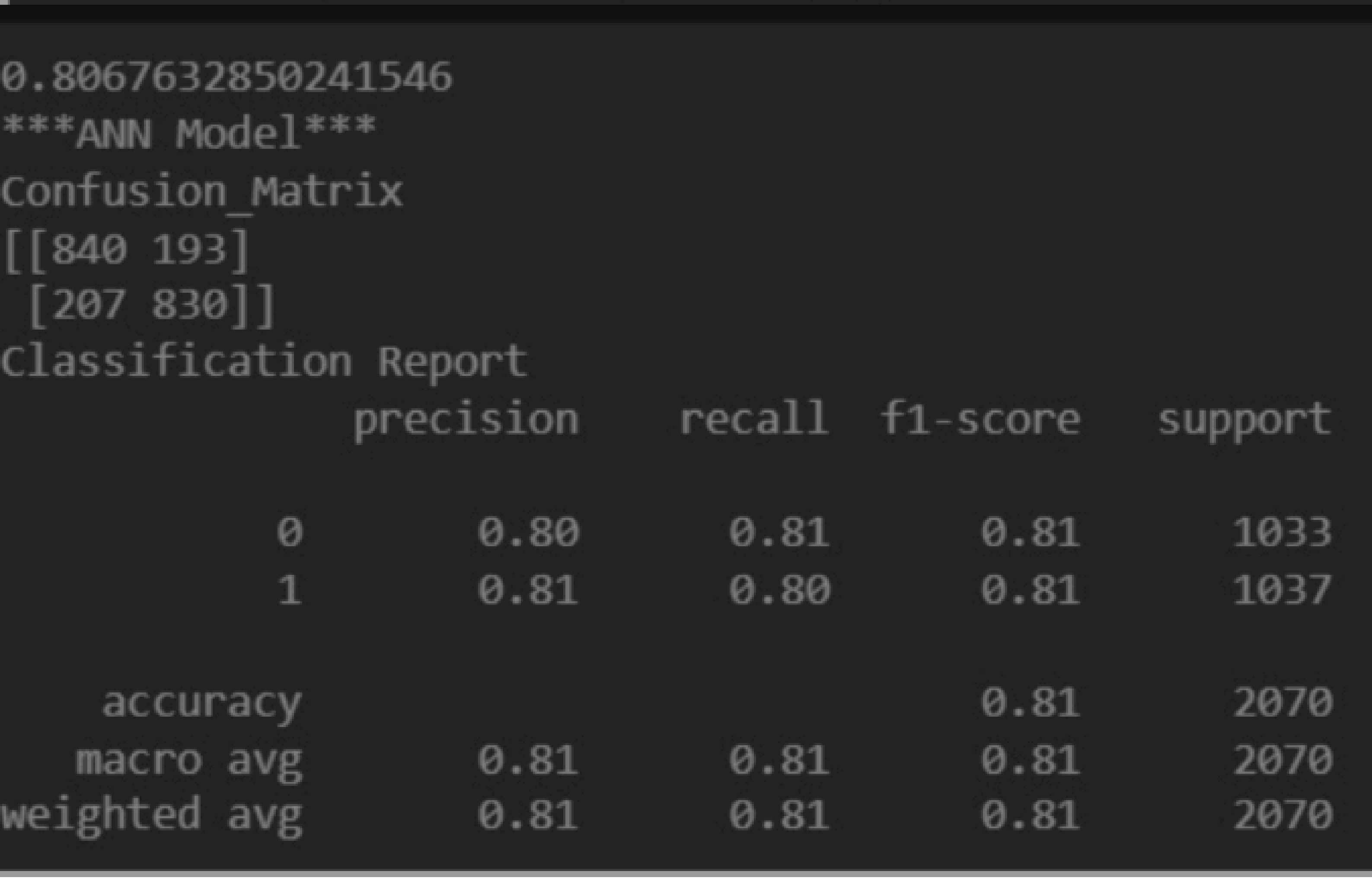
```
Epoch 195/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1564 - accuracy: 0.9335 - val_loss: 0.7783 - val_accuracy: 0.8093
Epoch 196/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1514 - accuracy: 0.9347 - val_loss: 0.7982 - val_accuracy: 0.7994
Epoch 197/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1549 - accuracy: 0.9327 - val_loss: 0.8319 - val_accuracy: 0.7917
Epoch 198/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1593 - accuracy: 0.9320 - val_loss: 0.7693 - val_accuracy: 0.8130
Epoch 199/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1535 - accuracy: 0.9362 - val_loss: 0.7646 - val_accuracy: 0.8089
Epoch 200/200
555/555 [==============================] - 1s 3ms/step - loss: 0.1544 - accuracy: 0.9356 - val_loss: 0.7744 - val_accuracy: 0.8115
```



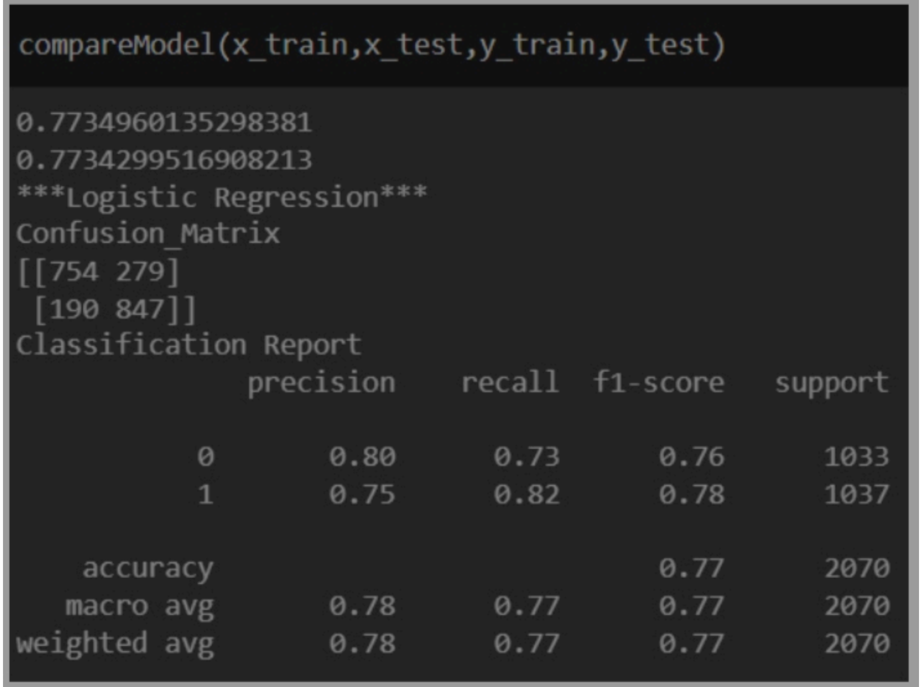
```
65/65 [==============================] - 0s 2ms/step
array([[False],
       [False],
       [ True],
       ...,
       [False],
       [False],
       [False]])
```



```
0.8067632850241546
***ANN Model***
Confusion_Matrix
[[840 193]
 [207 830]]
Classification Report
              precision    recall  f1-score   suppo

           0       0.80      0.81      0.81      10
           1       0.81      0.80      0.81      10

    accuracy                           0.81      20
```

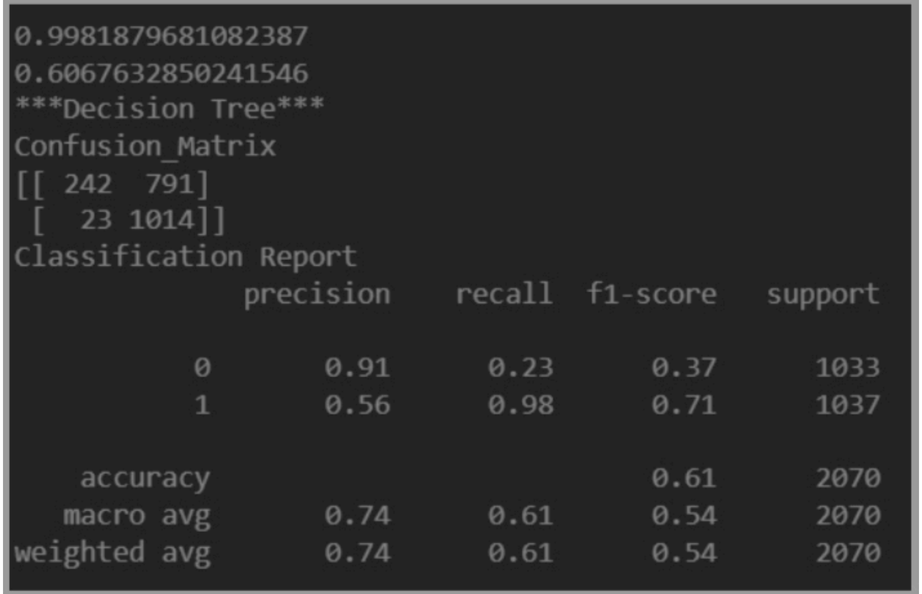


```
0.8067632850241546
***ANN Model***
Confusion_Matrix
[[840 193]
 [207 830]]
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.81      0.81      1033
           1       0.81      0.80      0.81      1037

    accuracy                           0.81      2070
   macro avg       0.81      0.81      0.81      2070
weighted avg       0.81      0.81      0.81      2070
```

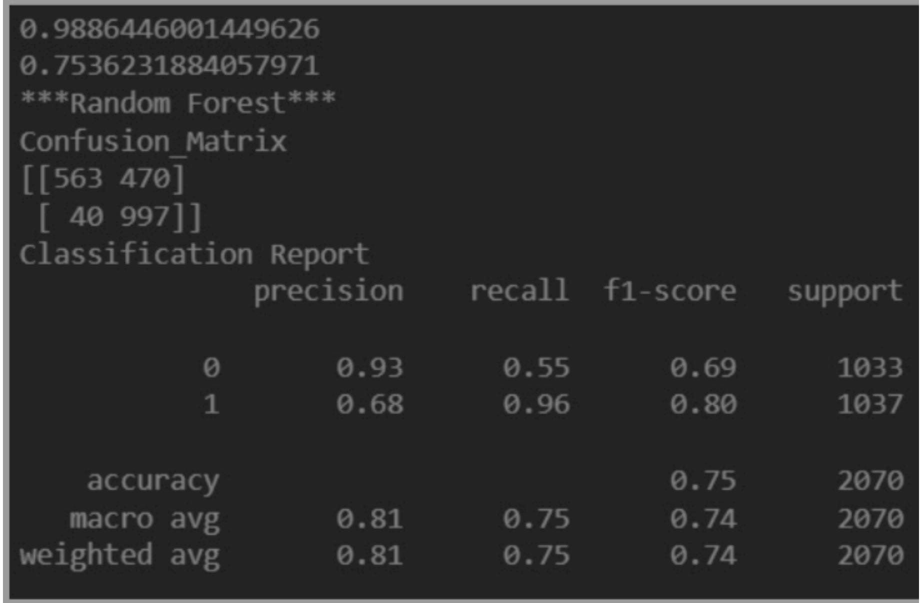


```
compareModel(x_train,x_test,y_train,y_test)
```

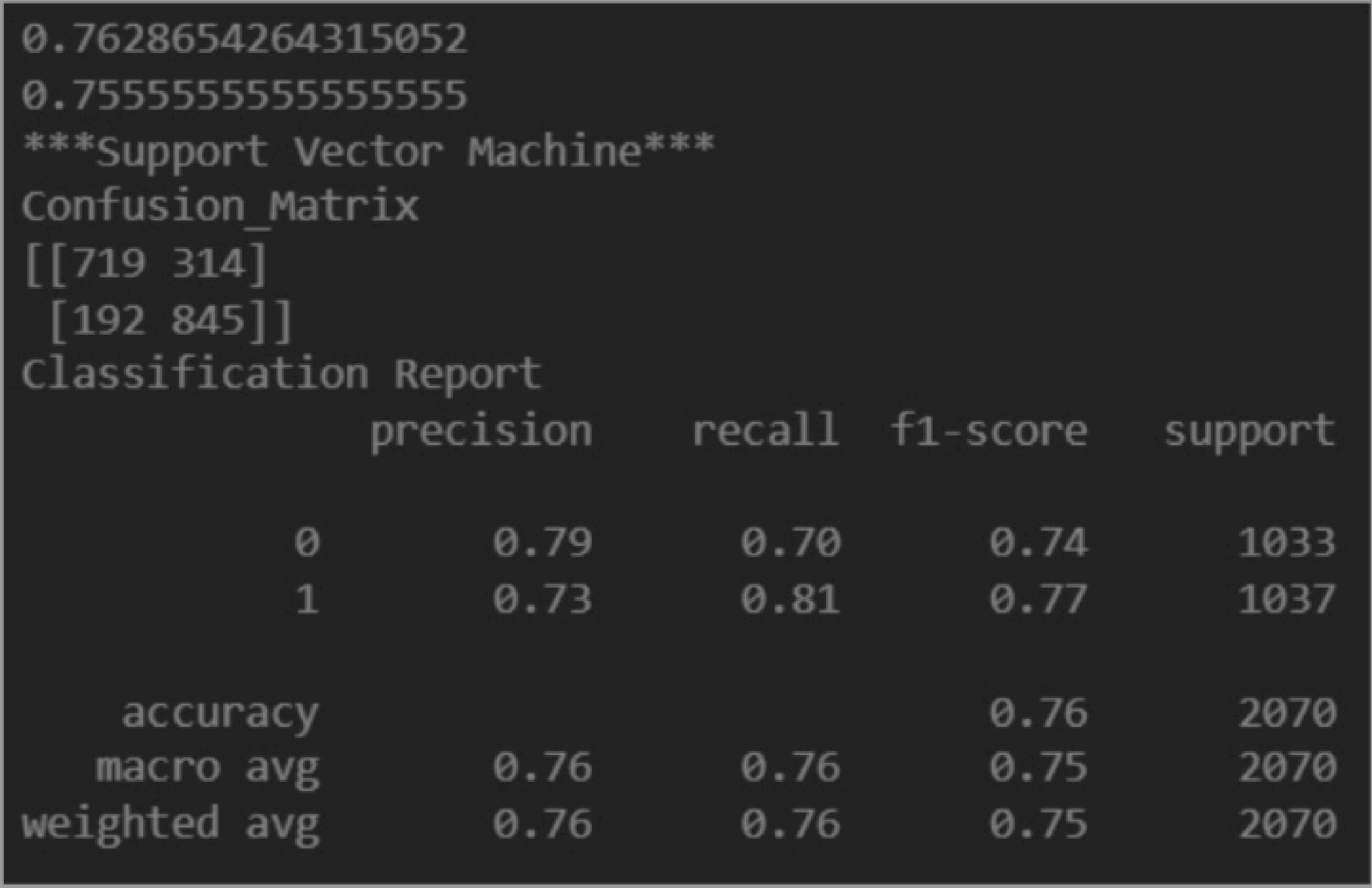
```
0.7734960135298381
0.7734299516908213
***Logistic Regression***
Confusion_Matrix
[[754 279]
 [190 847]]
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.73      0.76      1033
           1       0.75      0.82      0.78      1037

    accuracy                           0.77      2070
   macro avg       0.78      0.77      0.77      2070
weighted avg       0.78      0.77      0.77      2070
```



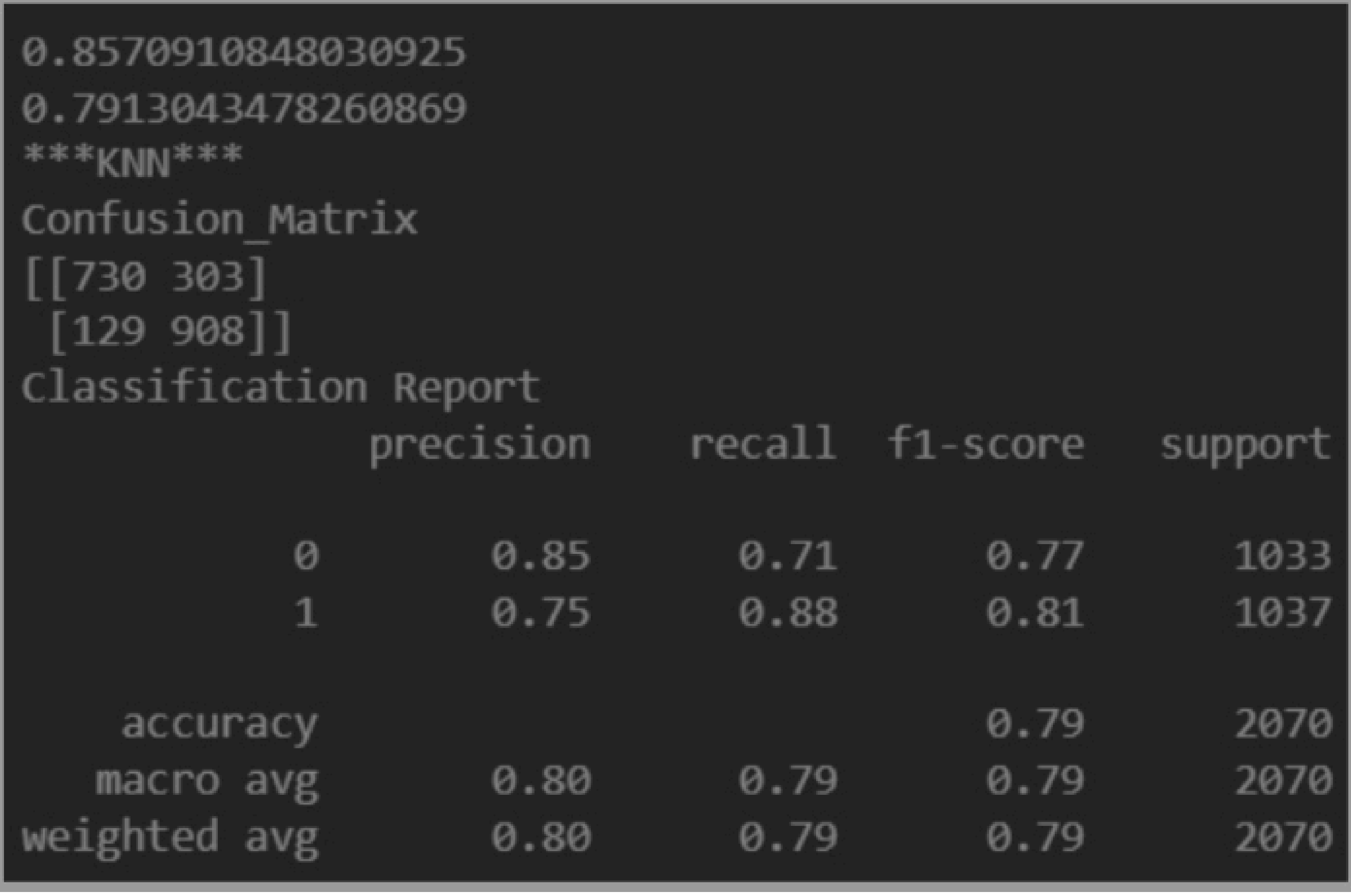
```
0.9981879681082387
0.6067632850241546
***Decision Tree***
Confusion_Matrix
[[ 242  791]
 [  23 1014]]
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.23      0.37      1033
           1       0.56      0.98      0.71      1037

    accuracy                           0.61      2070
   macro avg       0.74      0.61      0.54      2070
weighted avg       0.74      0.61      0.54      2070
```



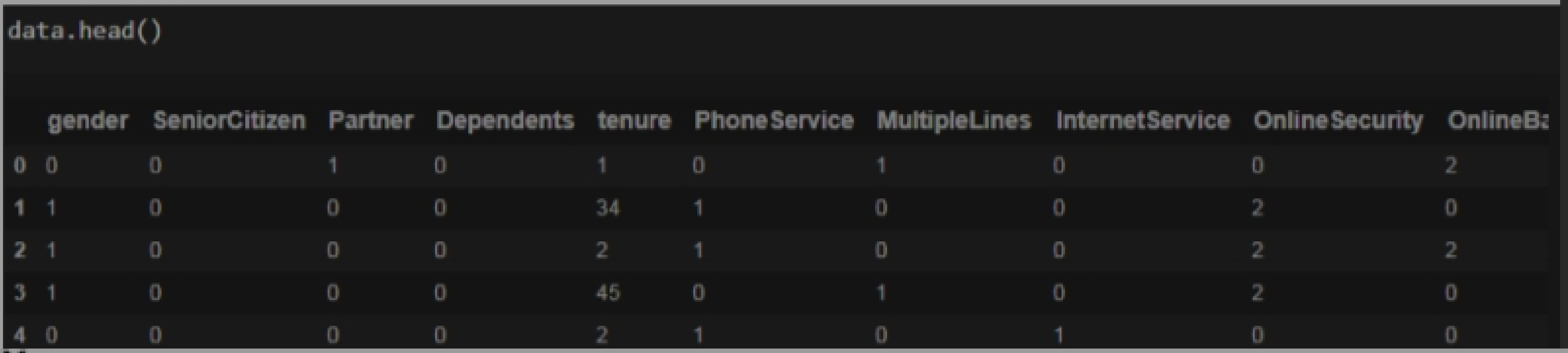
```
0.9886446001449626
0.7536231884057971
***Random Forest***
Confusion_Matrix
[[563 470]
 [ 40 997]]
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.55      0.69      1033
           1       0.68      0.96      0.80      1037

    accuracy                           0.75      2070
   macro avg       0.81      0.75      0.74      2070
weighted avg       0.81      0.75      0.74      2070
```



```
0.7628654264315052
0.7555555555555555
***Support Vector Machine***
Confusion_Matrix
[[719 314]
 [192 845]]
Classification Report
              precision    recall  f1-score   support


           0       0.79      0.70      0.74      1033
           1       0.73      0.81      0.77      1037

    accuracy                           0.76      2070
   macro avg       0.76      0.76      0.75      2070
weighted avg       0.76      0.76      0.75      2070
```



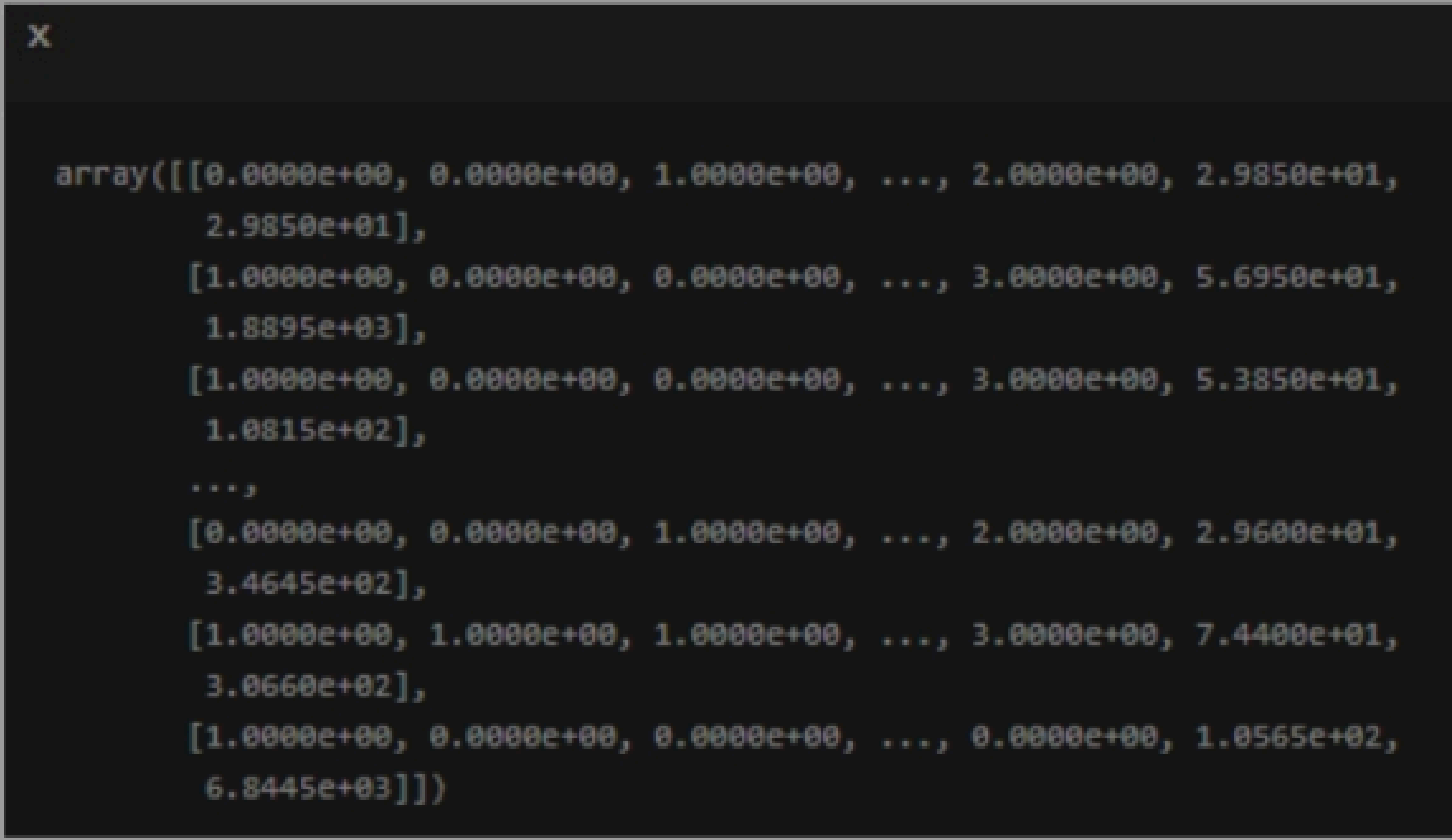
```
0.8570910848030925
0.7913043478260869
***KNN***
Confusion_Matrix
[[730 303]
 [129 908]]
Classification Report
              precision    recall  f1-score   support


           0       0.85      0.71      0.77      1033
           1       0.75      0.88      0.81      1037

    accuracy                           0.79      2070
   macro avg       0.80      0.79      0.79      2070
weighted avg       0.80      0.79      0.79      2070
```



```
data.head()
```

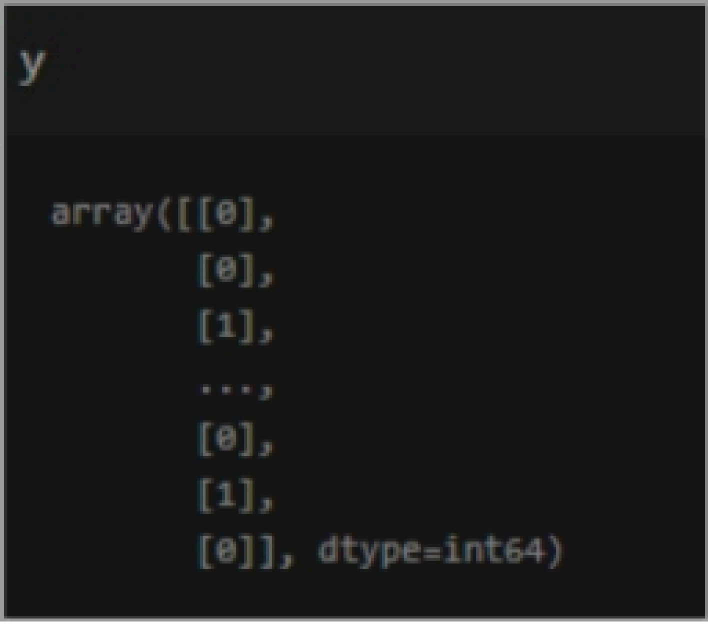
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBa
0	0	0	1	0	1	0	1	0	0	2
1	1	0	0	0	34	1	0	0	2	0
2	1	0	0	0	2	1	0	0	2	2
3	1	0	0	0	45	0	1	0	2	0
4	0	0	0	0	2	1	0	1	0	0

X

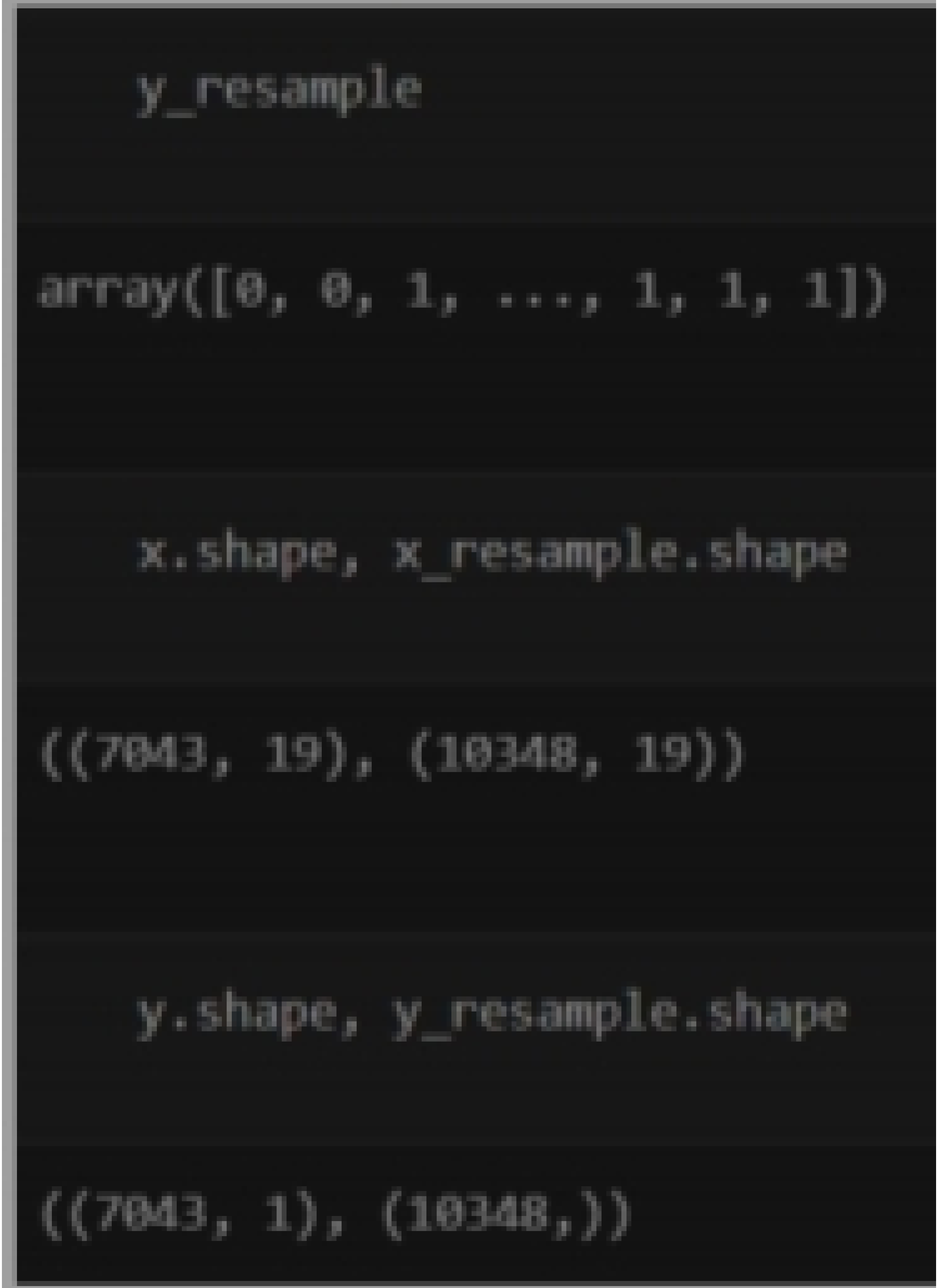


```
x

array([[0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 2.0000e+00, 2.9850e+01,
        2.9850e+01],
       [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 3.0000e+00, 5.6950e+01,
        1.8895e+03],
       [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 3.0000e+00, 5.3850e+01,
        1.0815e+02],
       ...,
       [0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 2.0000e+00, 2.9600e+01,
        3.4645e+02],
       [1.0000e+00, 1.0000e+00, 1.0000e+00, ..., 3.0000e+00, 7.4400e+01,
        3.0660e+02],
       [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 0.0000e+00, 1.0565e+02,
        6.8445e+03]])
```

Y



```
y

array([[0],
       [0],
       [1],
       ...,
       [0],
       [1],
       [0]], dtype=int64)
```



```
y_resample
```

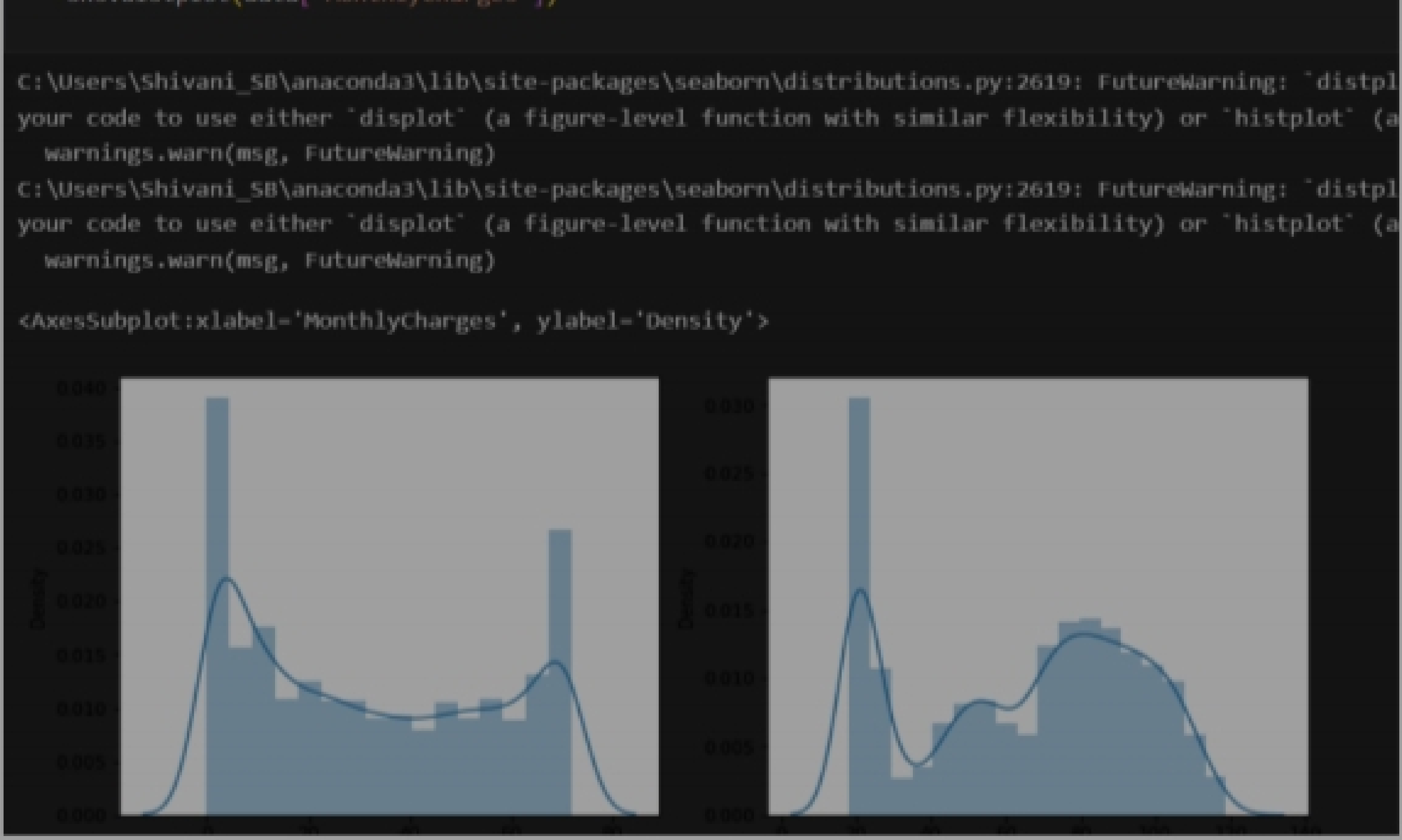
```
array([0, 0, 1, ..., 1, 1, 1])
```

```
x.shape, x_resample.shape
```

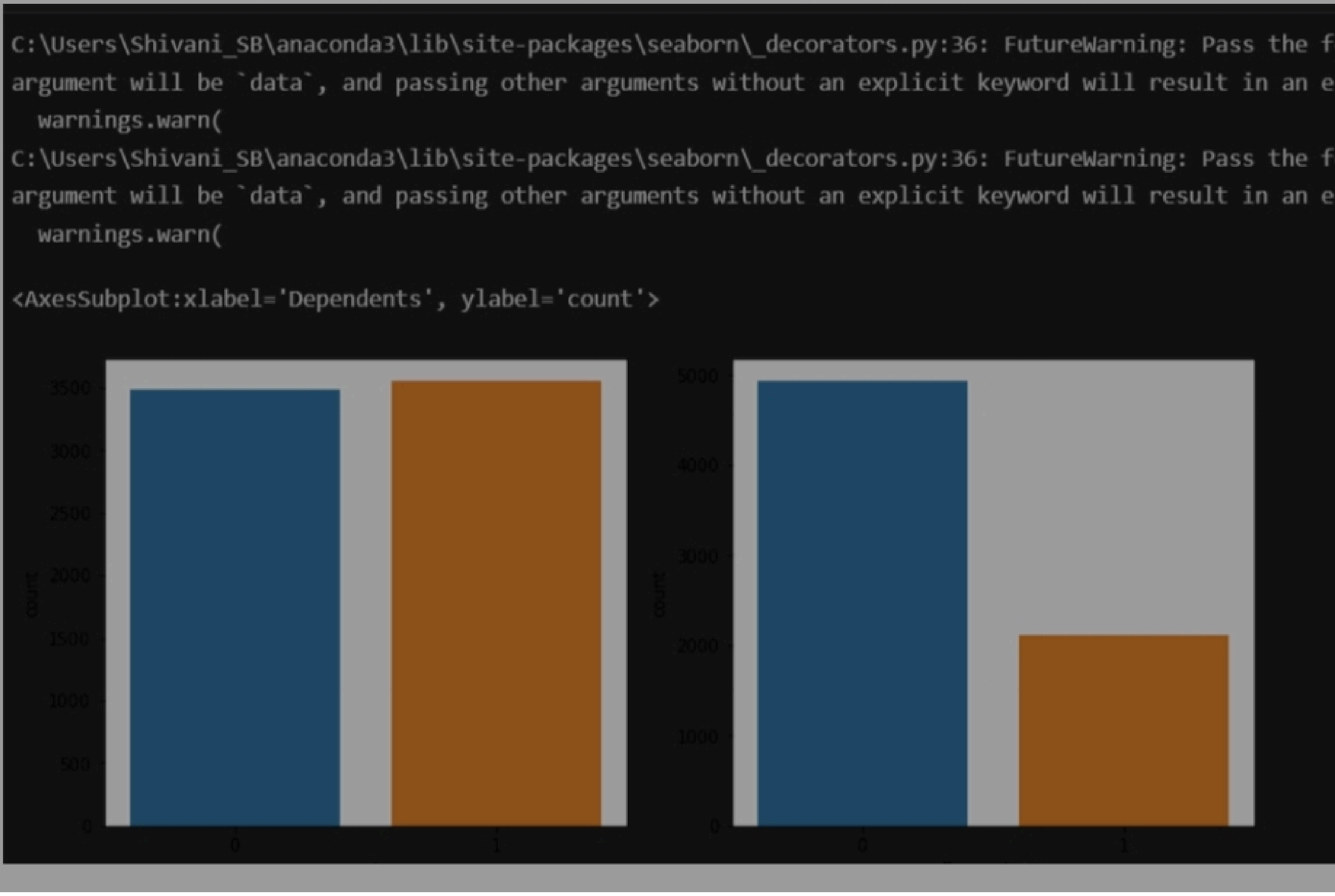
```
((7043, 19), (10348, 19))
```

```
y.shape, y_resample.shape
```

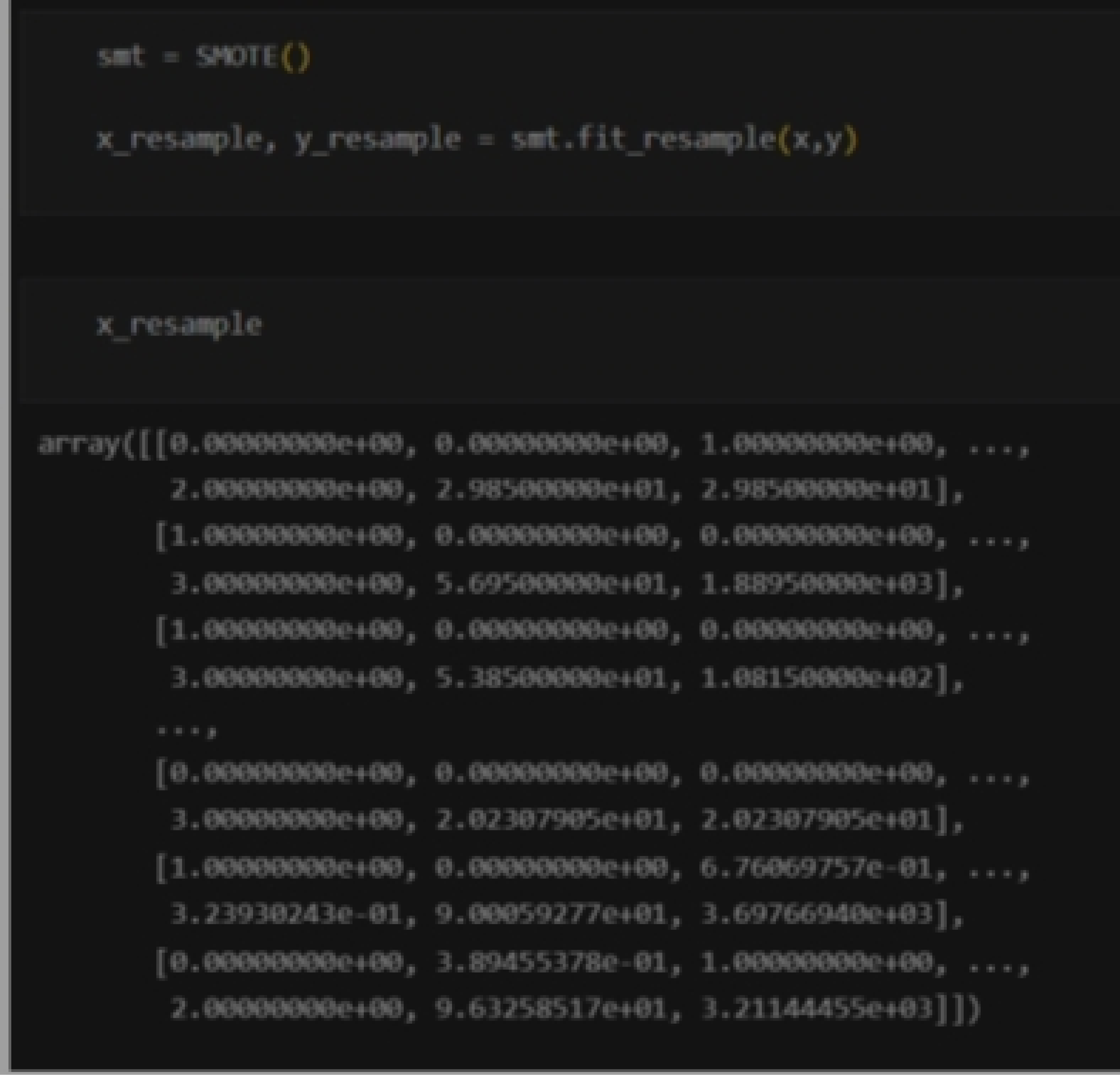
```
((7043, 1), (10348,))
```



```
C:\Users\Shivani_SB\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distpl
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (a
  warnings.warn(msg, FutureWarning)
C:\Users\Shivani_SB\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distpl
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (a
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='MonthlyCharges', ylabel='Density'>
```



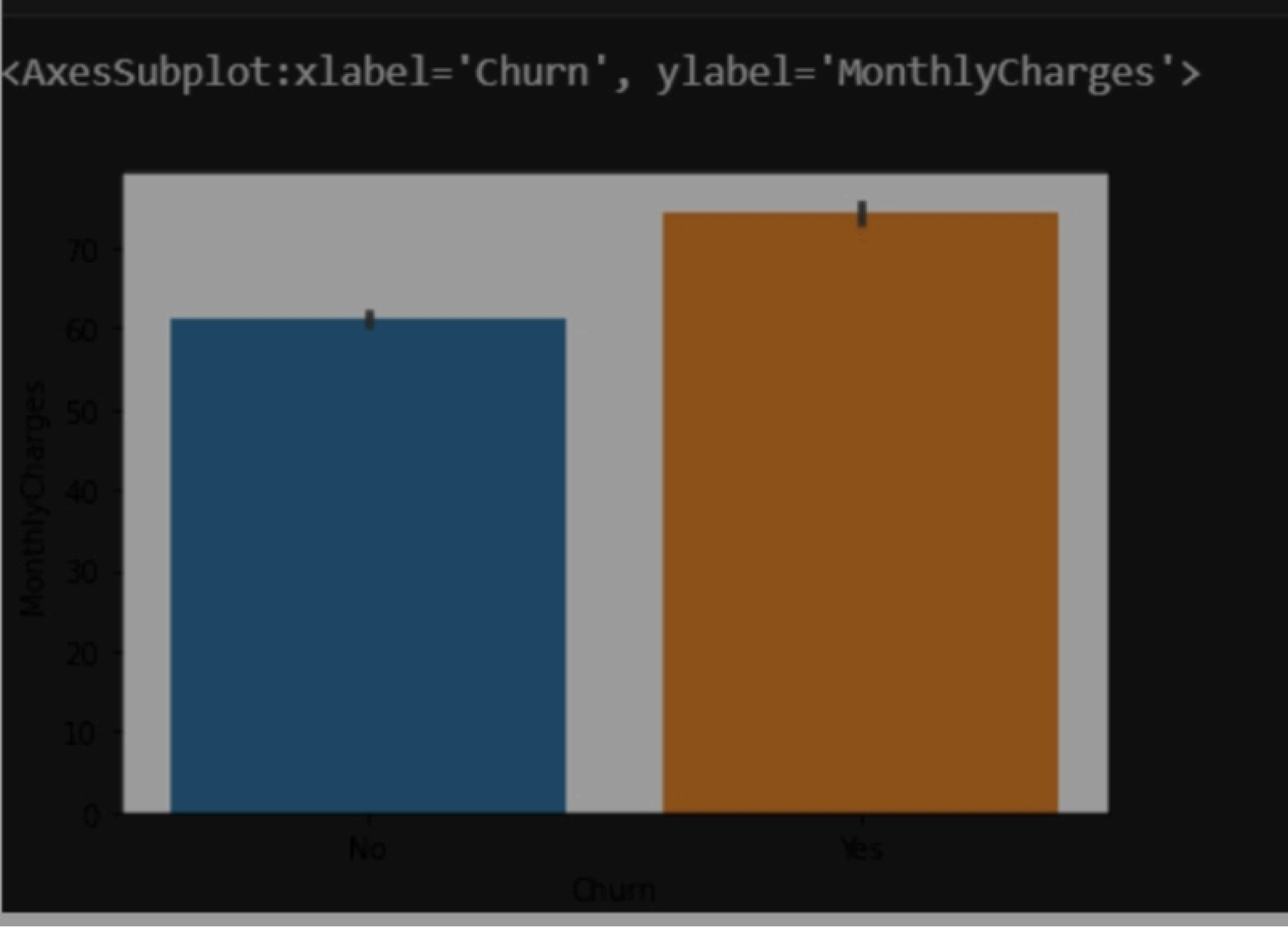
```
C:\Users\Shivani_SB\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
argument will be `data`, and passing other arguments without an explicit keyword will result in an e
  warnings.warn(
C:\Users\Shivani_SB\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
argument will be `data`, and passing other arguments without an explicit keyword will result in an e
  warnings.warn(

<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



```
smt = SMOTE()

x_resample, y_resample = smt.fit_resample(x,y)
```

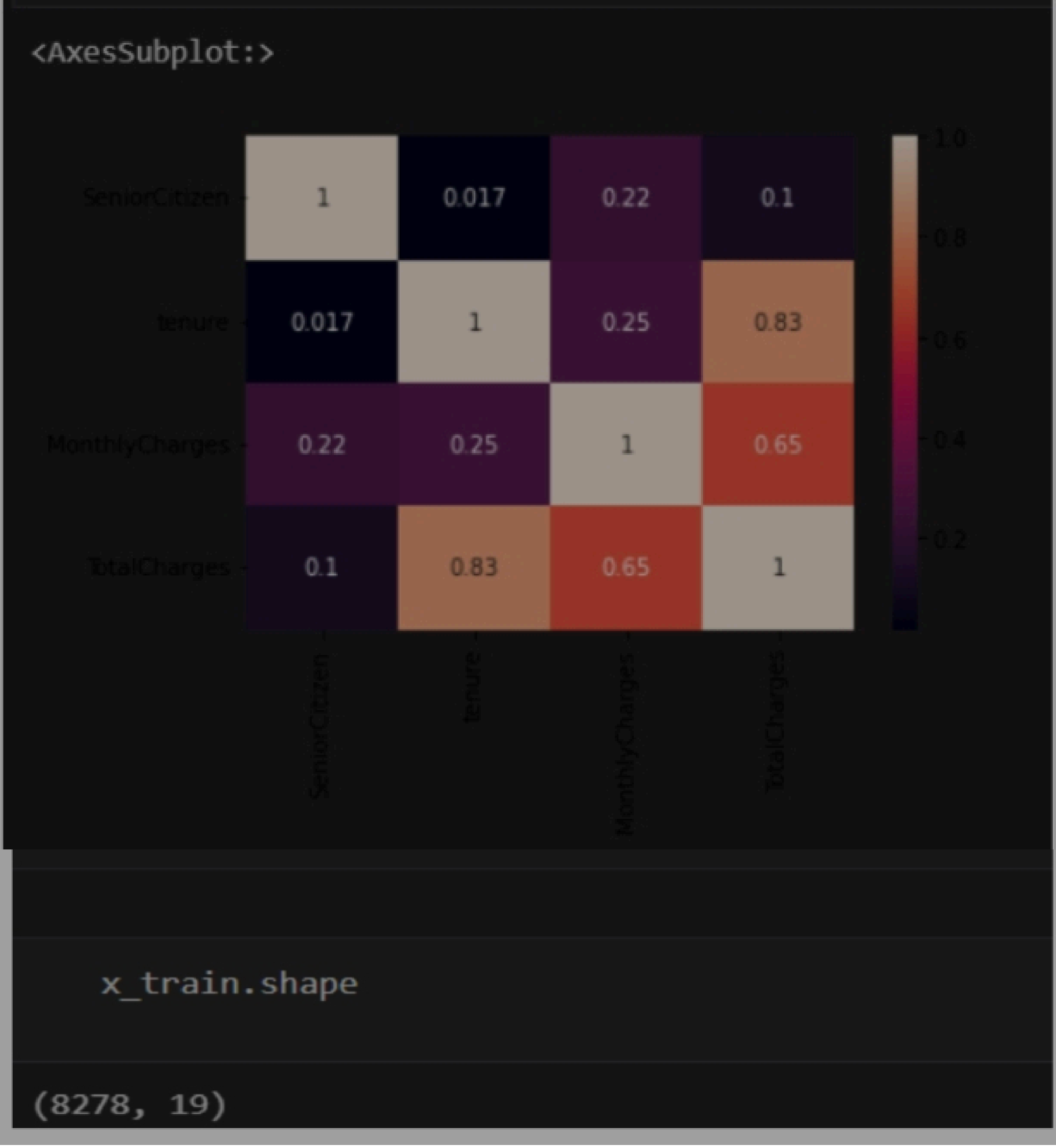
```
x_resample
```

```
array([[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, ...,
        2.00000000e+00, 2.98500000e+01, 2.98500000e+01],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        3.00000000e+00, 5.69500000e+01, 1.88950000e+03],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        3.00000000e+00, 5.38500000e+01, 1.08150000e+02],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        3.00000000e+00, 2.02307905e+01, 2.02307905e+01],
       [1.00000000e+00, 0.00000000e+00, 6.76069757e-01, ...,
        3.23930243e-01, 9.00059277e+01, 3.69766940e+03],
       [0.00000000e+00, 3.89455378e-01, 1.00000000e+00, ...,
        2.00000000e+00, 9.63258517e+01, 3.21144455e+03]])
```



```
<AxesSubplot:xlabel='Churn', ylabel='MonthlyCharges'>
```

MonthlyCharges
0
10
20
30
40
50
60
70
No
Yes
Churn

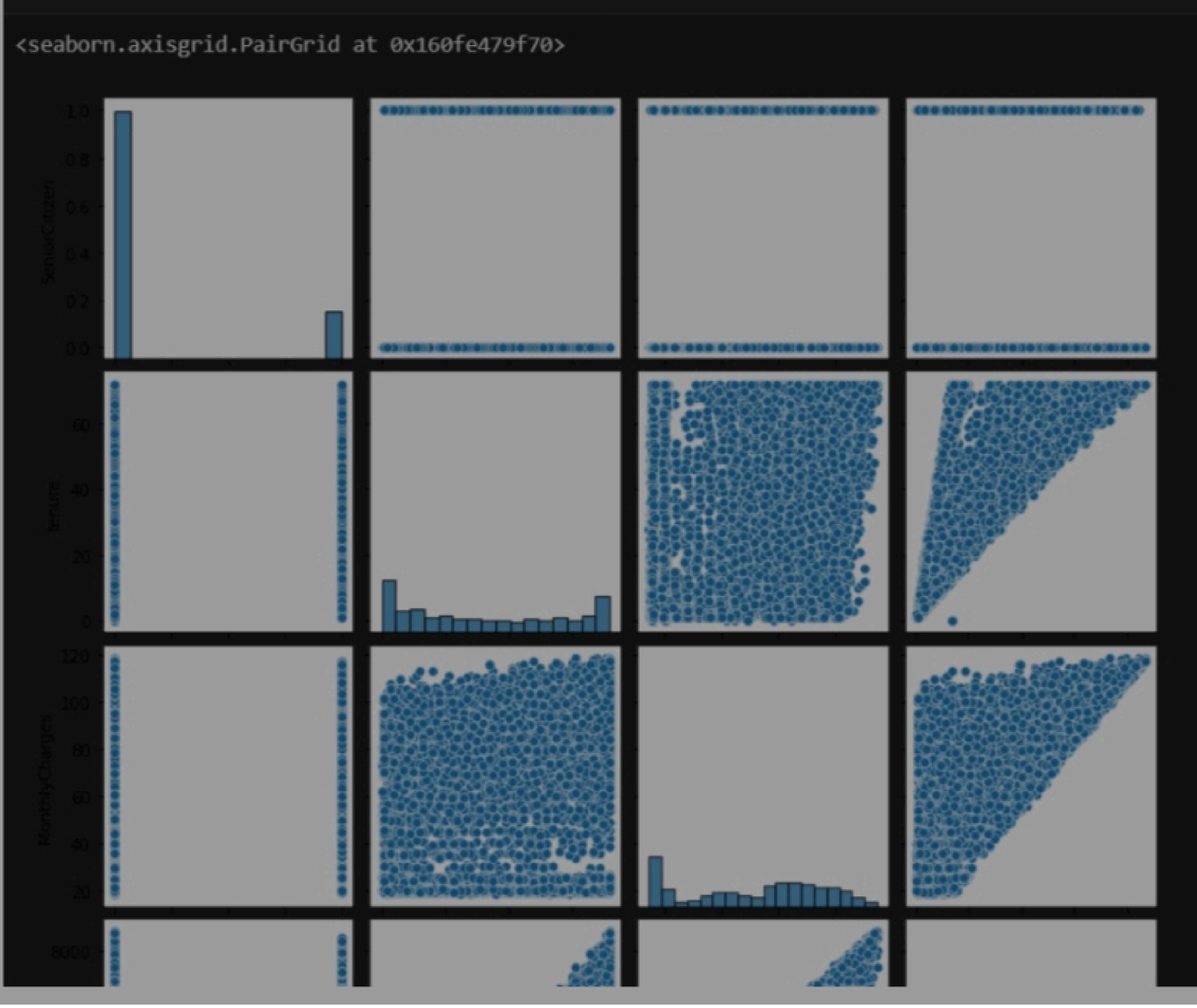


```
<AxesSubplot:>
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
SeniorCitizen	1	0.017	0.22	0.1
tenure	0.017	1	0.25	0.83
MonthlyCharges	0.22	0.25	1	0.65
TotalCharges	0.1	0.83	0.65	1

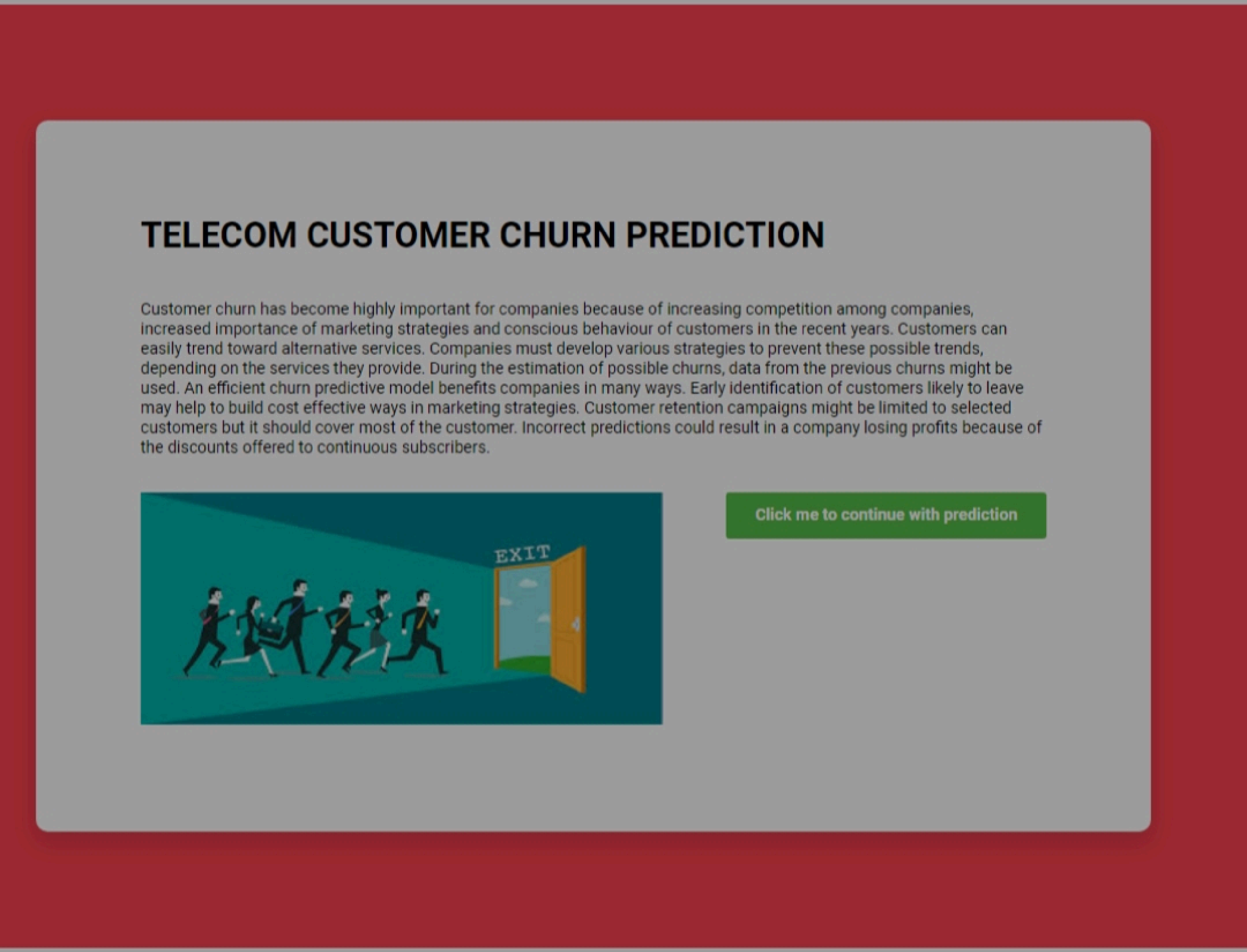
```
x_train.shape
```

```
(8278, 19)
```



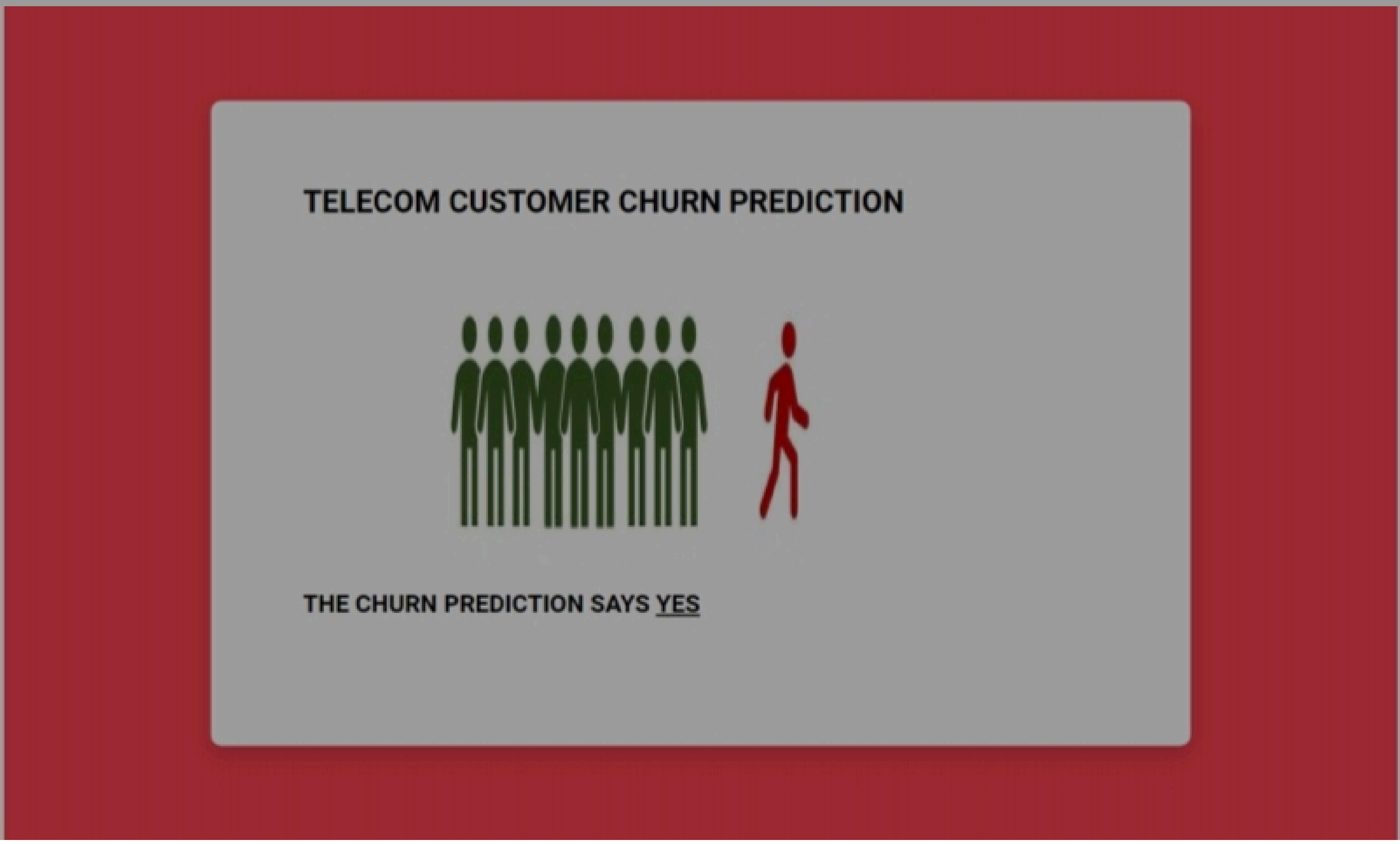
```
<seaborn.axisgrid.PairGrid at 0x160fe479f70>
```

SeniorCitizen
1.0
0.8
0.6
0.4
0.2
0.0
tenure
60
40
20
0
MonthlyCharges
120
100
80
60
40
20
8000



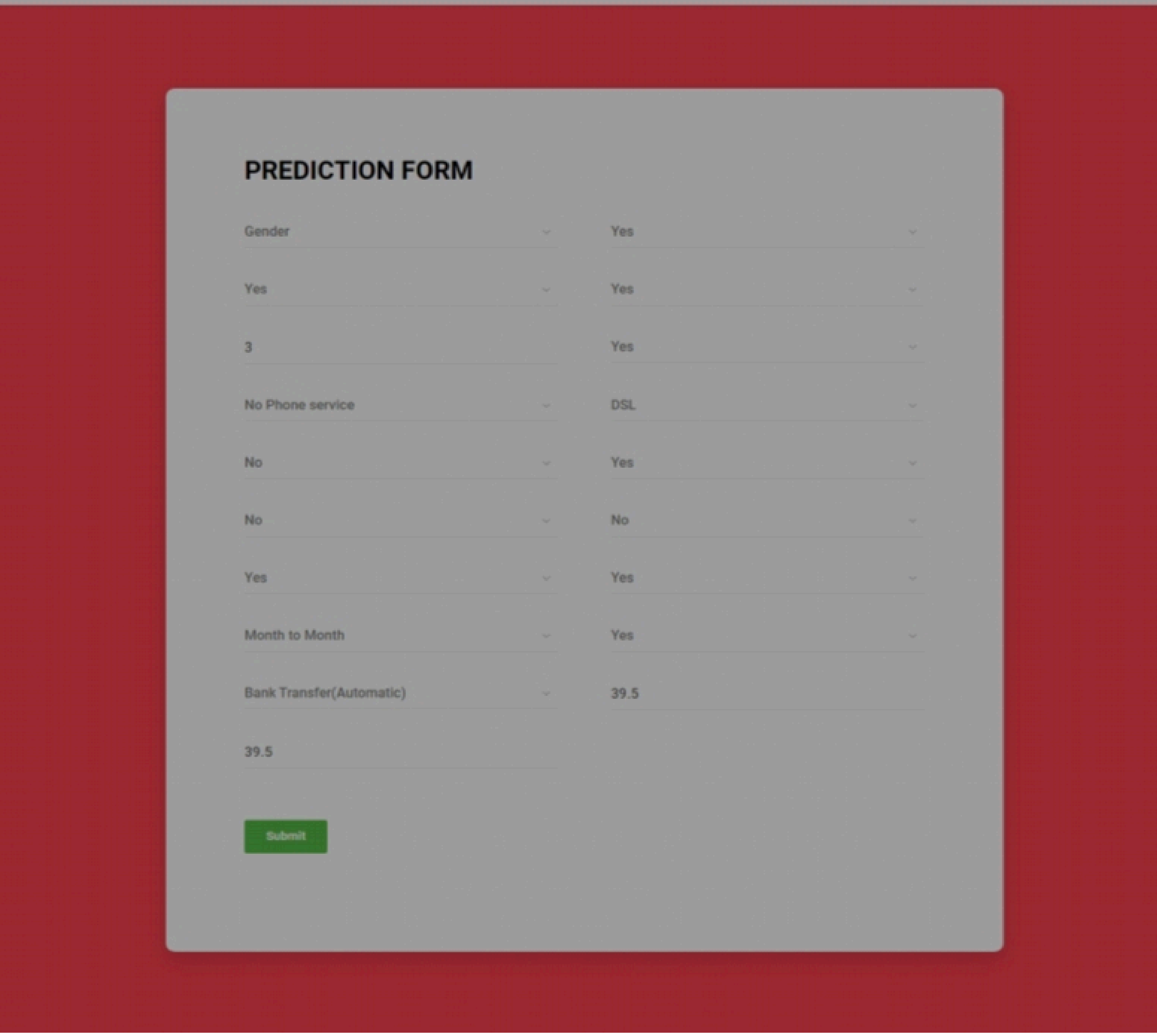
TELECOM CUSTOMER CHURN PREDICTION

Customer churn has become highly important for companies because of increasing competition among companies, increased importance of marketing strategies and conscious behaviour of customers in the recent years. Customers can easily trend toward alternative services. Companies must develop various strategies to prevent these possible trends, depending on the services they provide. During the estimation of possible churns, data from the previous churns might be used. An efficient churn predictive model benefits companies in many ways. Early identification of customers likely to leave may help to build cost effective ways in marketing strategies. Customer retention campaigns might be limited to selected customers but it should cover most of the customer. Incorrect predictions could result in a company losing profits because of the discounts offered to continuous subscribers.



TELECOM CUSTOMER CHURN PREDICTION

THE CHURN PREDICTION SAYS YES



PREDICTION FORM

Gender	Yes
Yes	Yes
3	Yes
No Phone service	DSL
No	Yes
No	No
Yes	Yes
Month to Month	Yes
Bank Transfer(Automatic)	39.5
39.5	