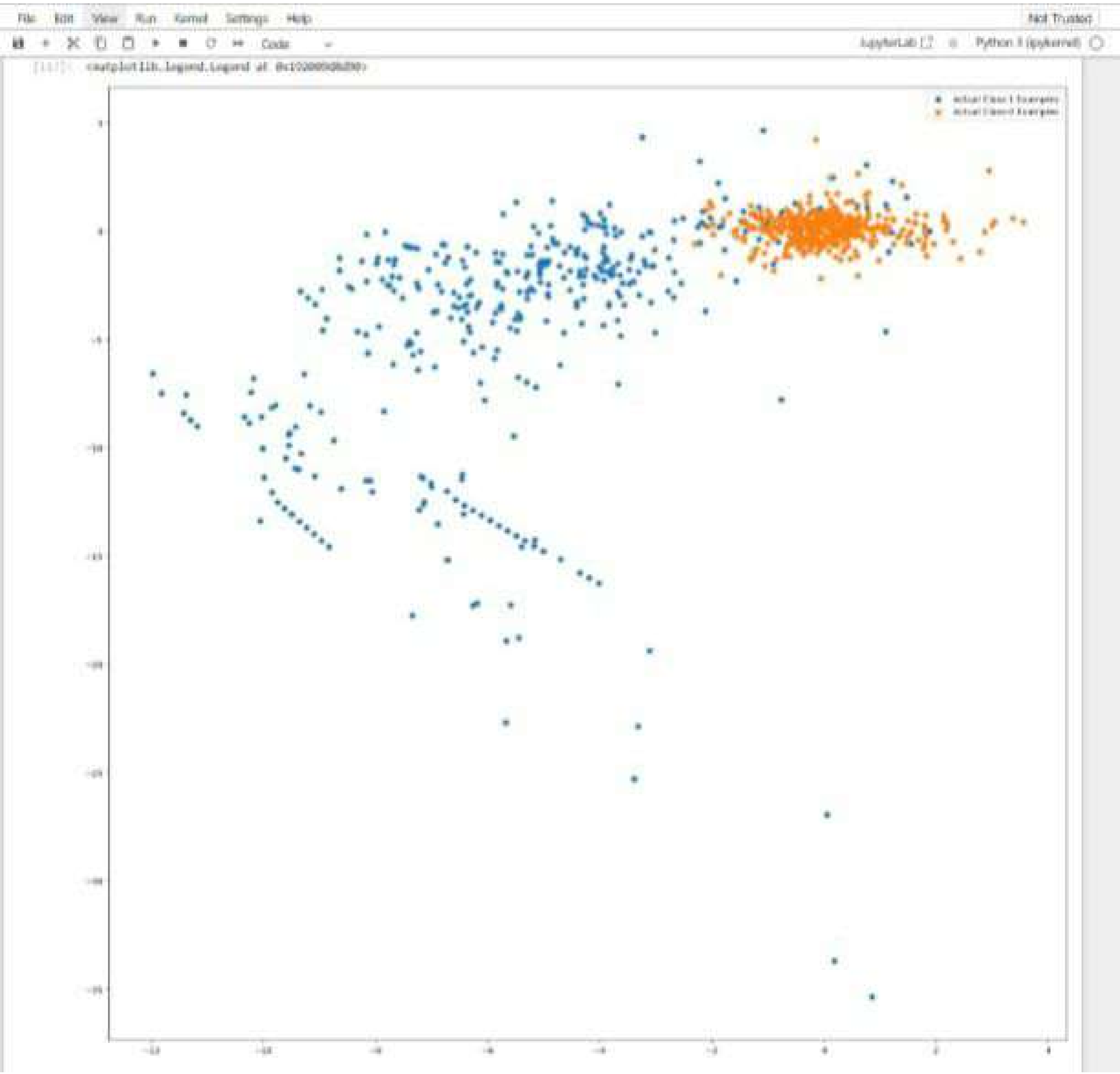
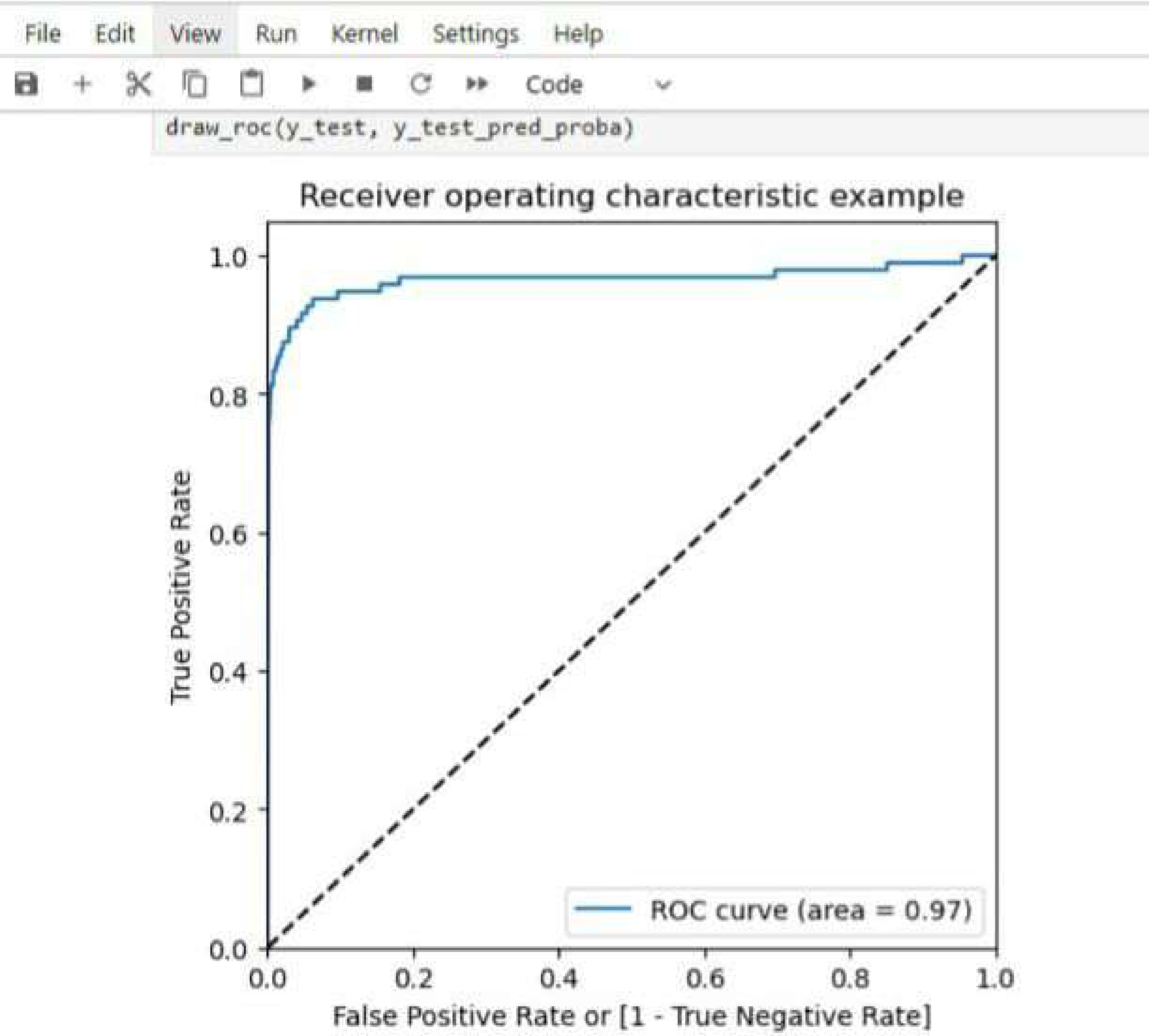
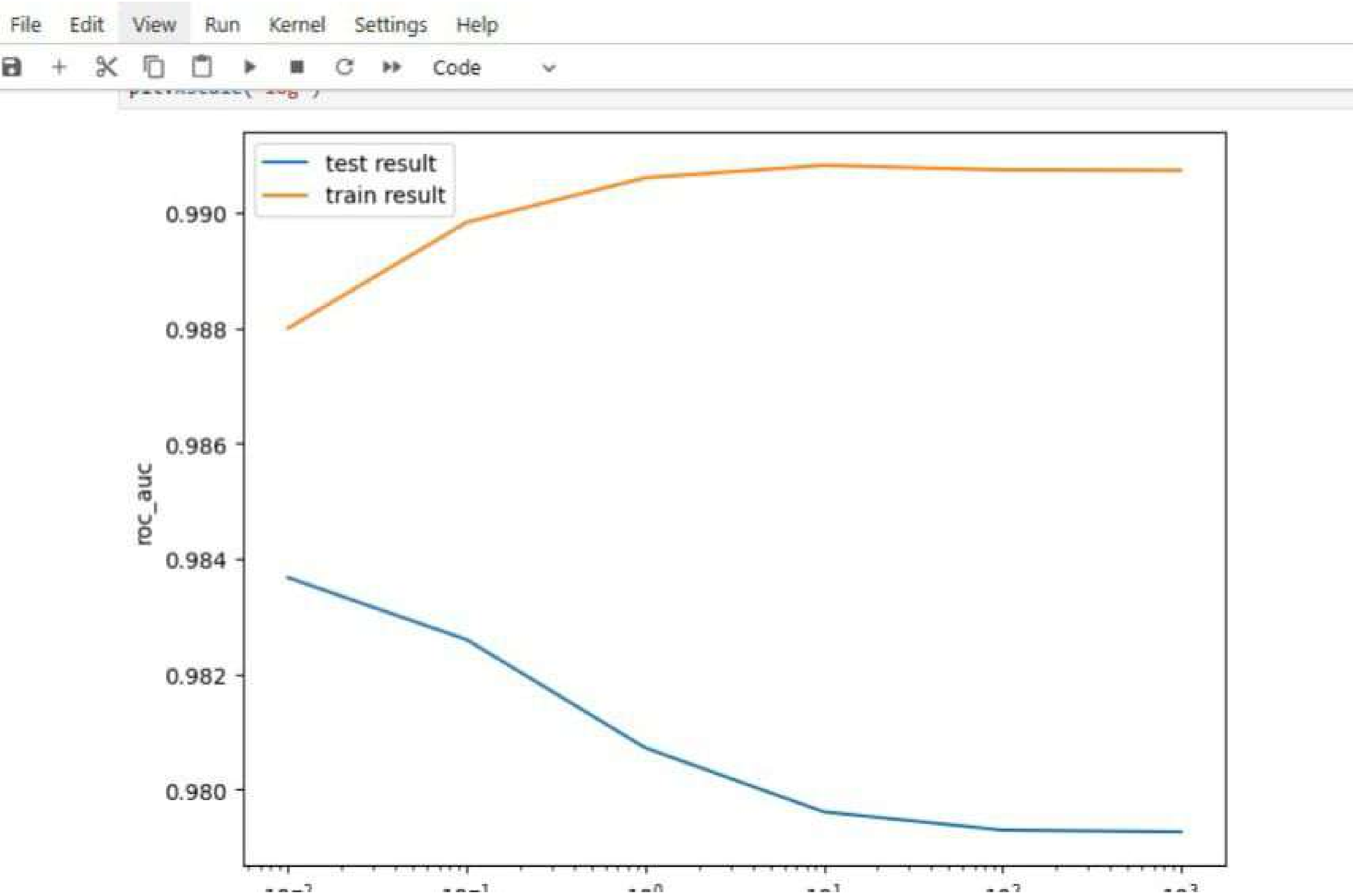

File
Edit
View
Run
Kernel
Settings
Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)
Actual Class 1 Examples
Actual Class 0 Examples
5
0
-5
-10
-15
-20
-25
-30
-35
-12
-10
-8
-6
-4
-2
0
2
4

```
[11]: <matplotlib.legend.Legend at 0x[illegible]>
```

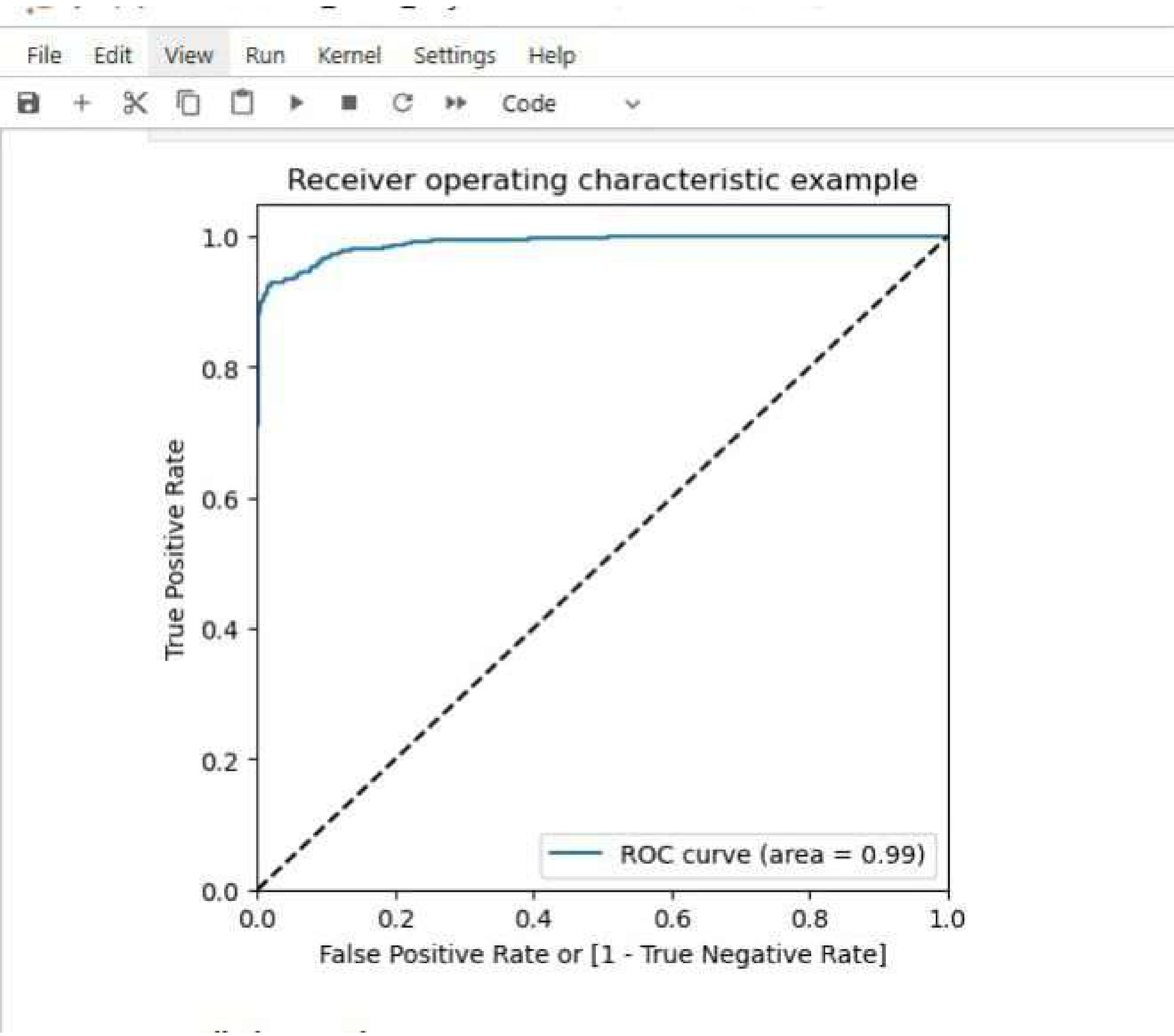


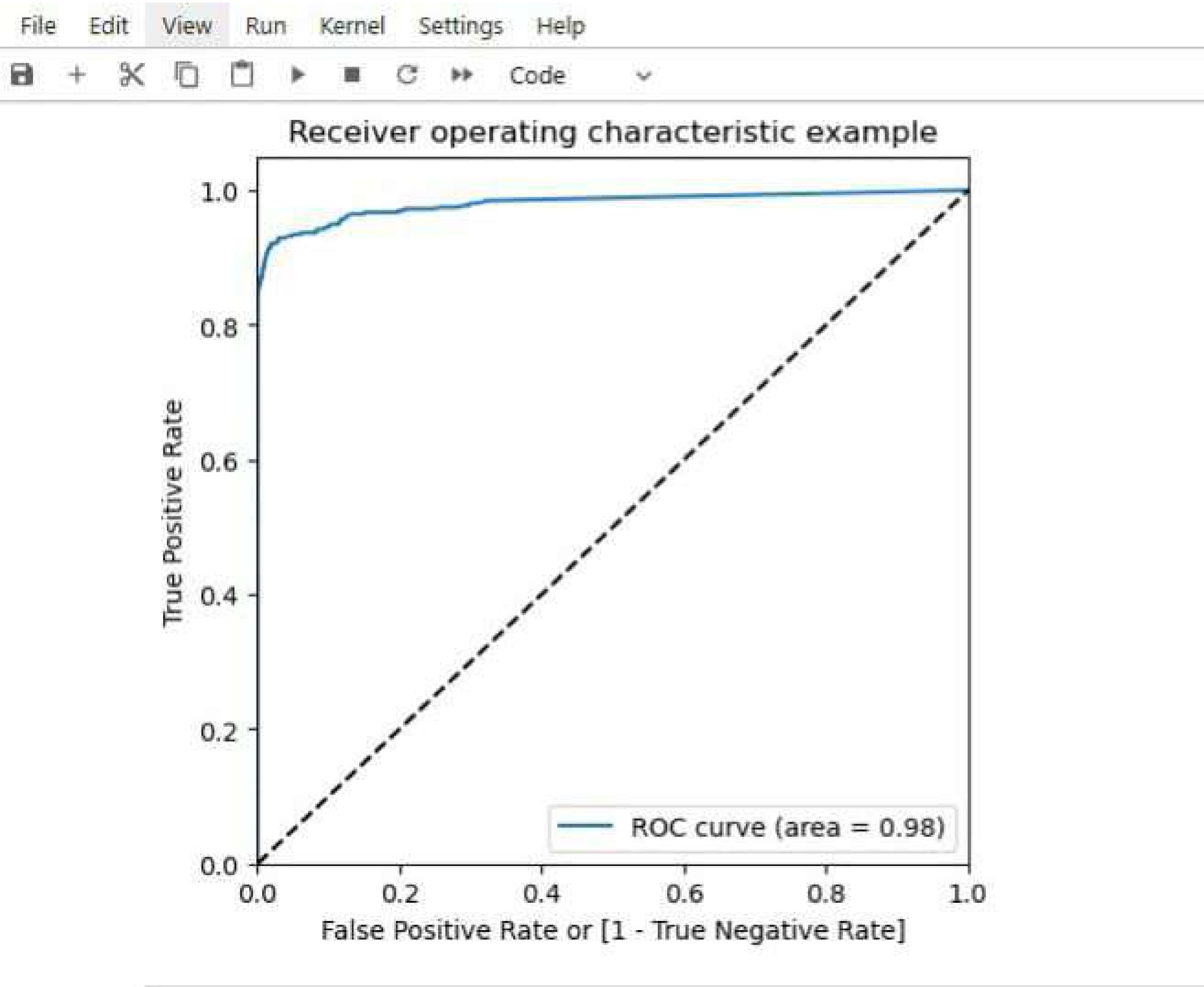
```
draw_roc(y_test, y_test_pred_proba)
```

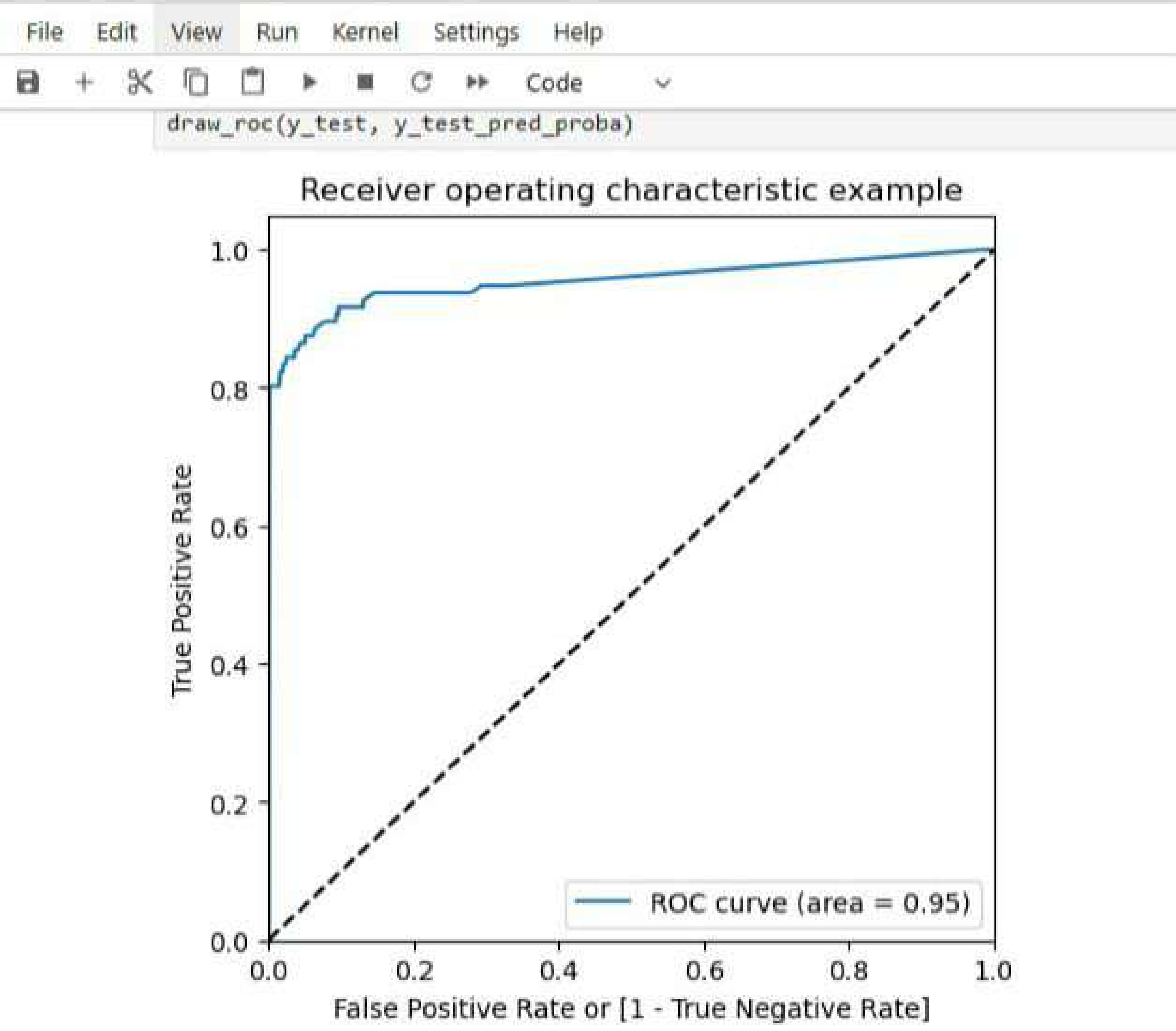
Receiver operating characteristic example
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate or [1 - True Negative Rate]
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.97)



```
plt.xscale('log')
```

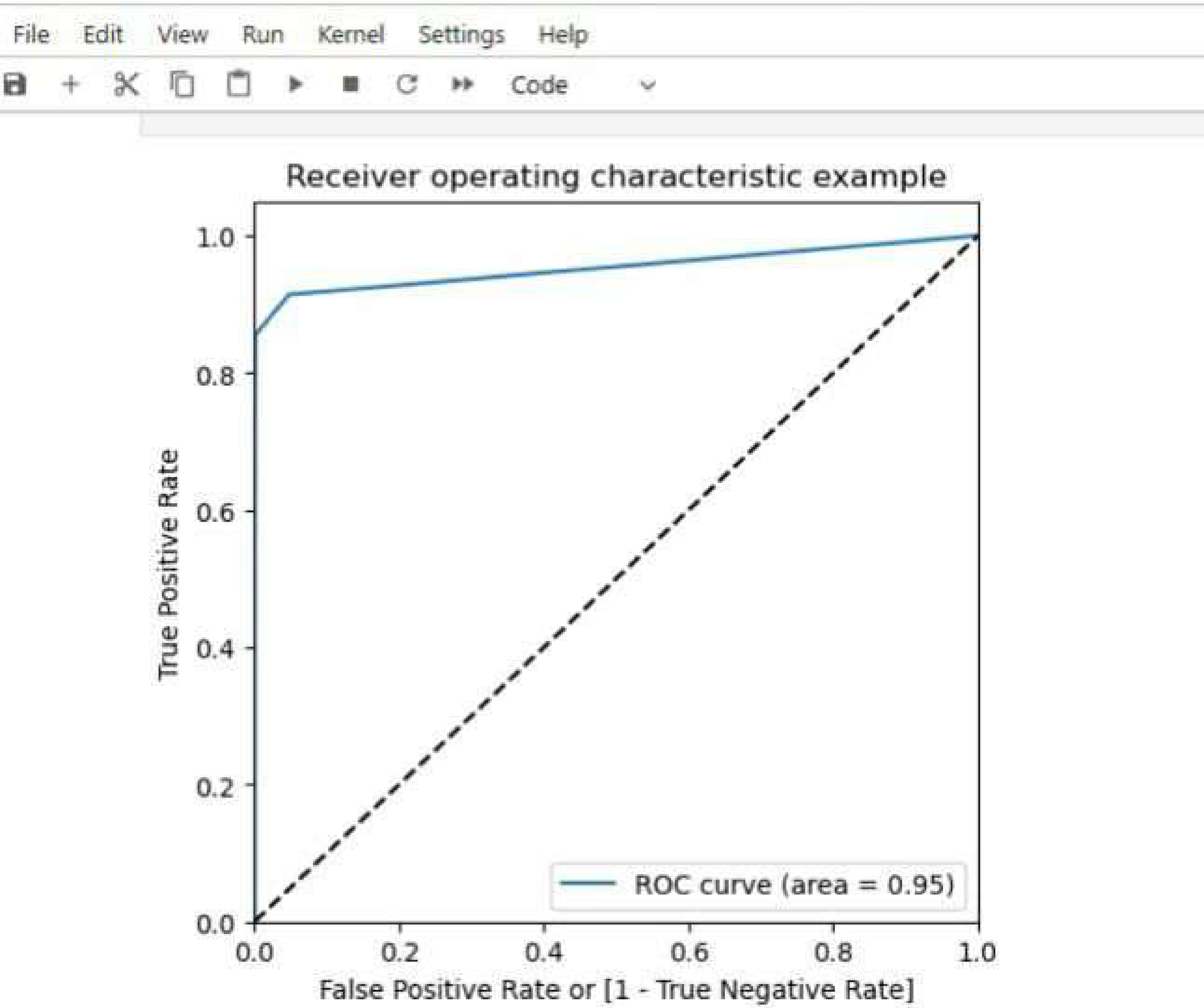

File
Edit
View
Run
Kernel
Settings
Help
Code
Receiver operating characteristic example
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate or [1 - True Negative Rate]
ROC curve (area = 0.99)

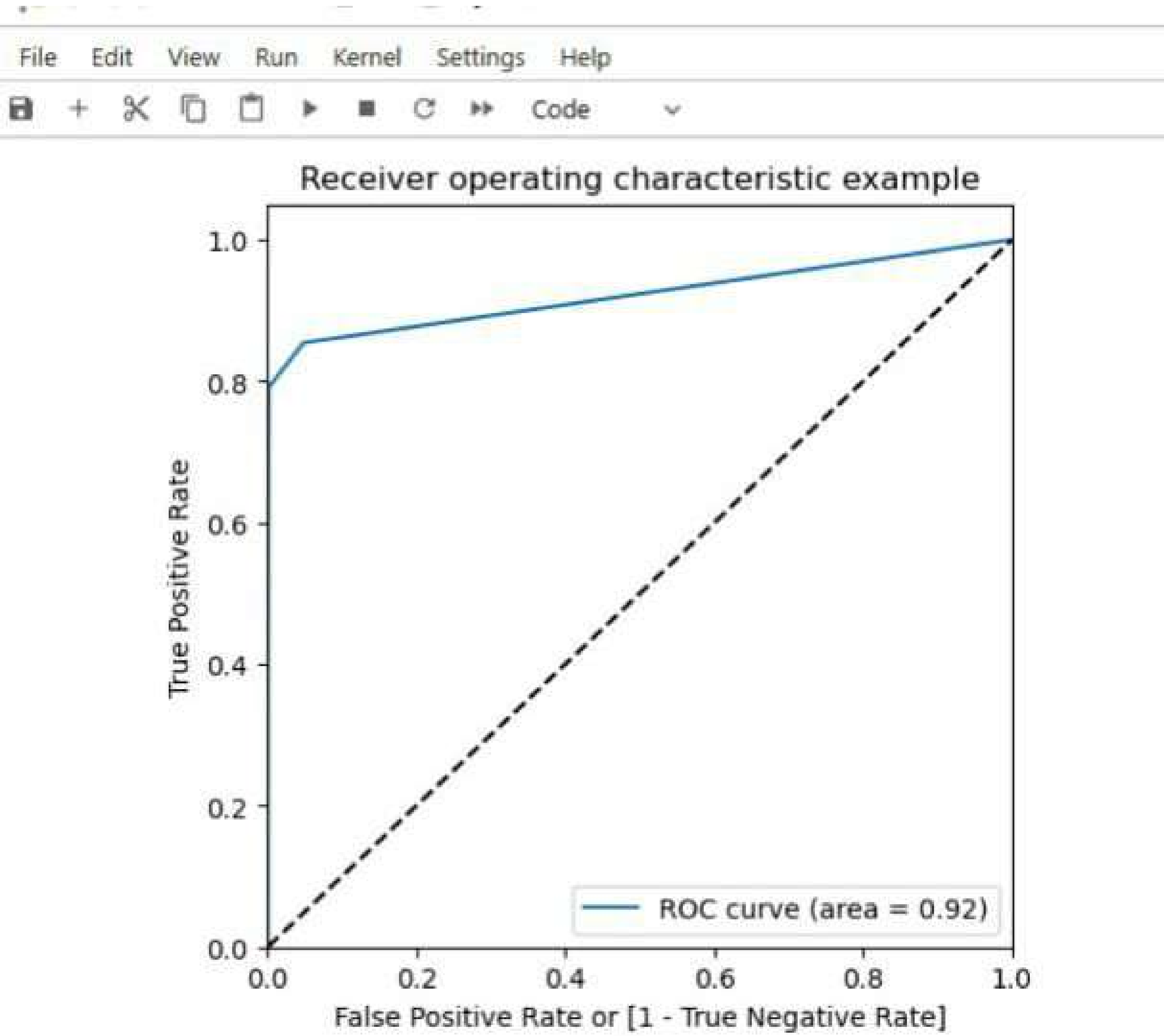

File
Edit
View
Run
Kernel
Settings
Help
Code
Receiver operating characteristic example
True Positive Rate
False Positive Rate or [1 - True Negative Rate]
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.98)

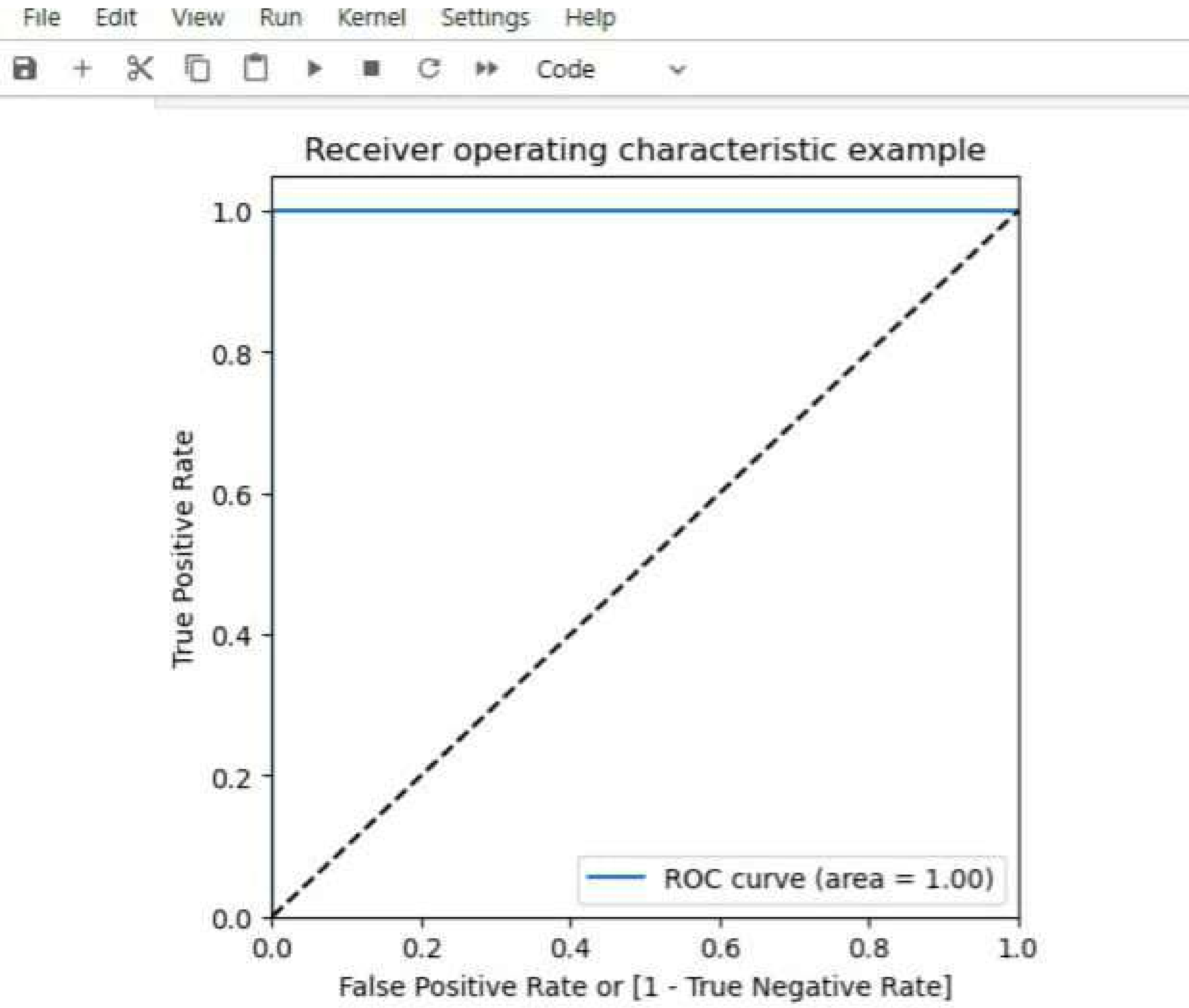


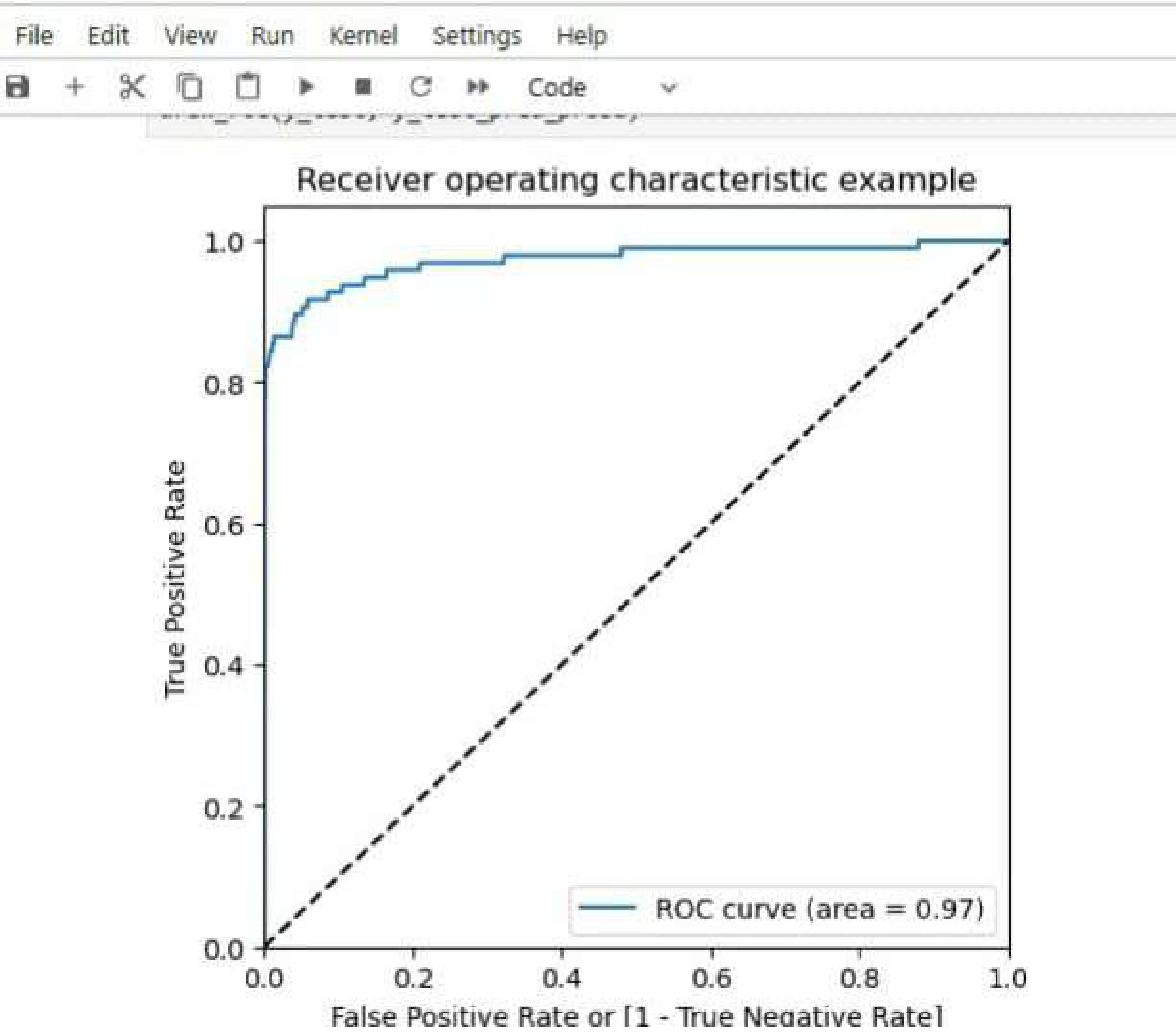
```
draw_roc(y_test, y_test_pred_proba)
```

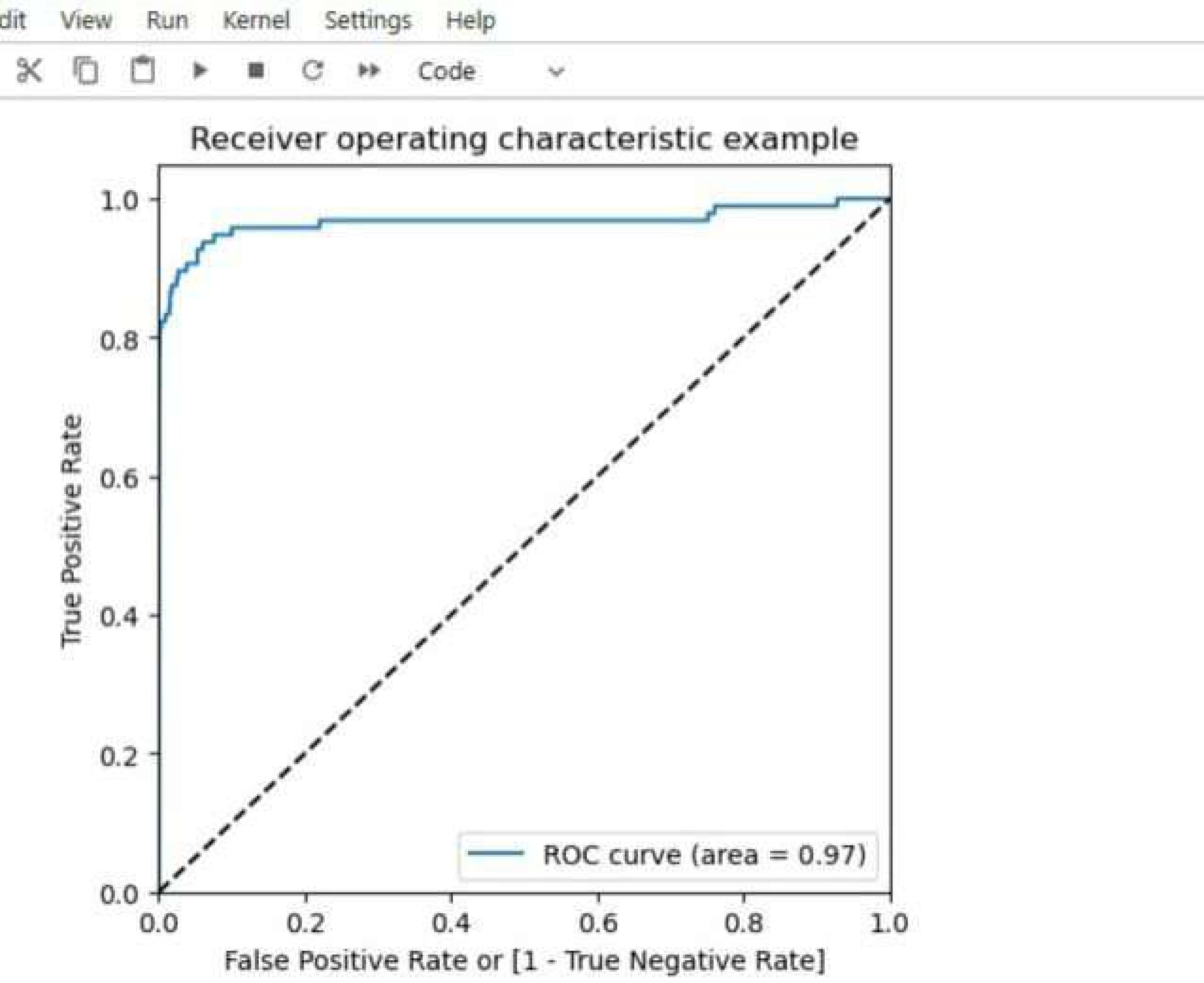
Receiver operating characteristic example
True Positive Rate
False Positive Rate or [1 - True Negative Rate]
ROC curve (area = 0.95)
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


File
Edit
View
Run
Kernel
Settings
Help
Code
Receiver operating characteristic example
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate or [1 - True Negative Rate]
ROC curve (area = 0.95)

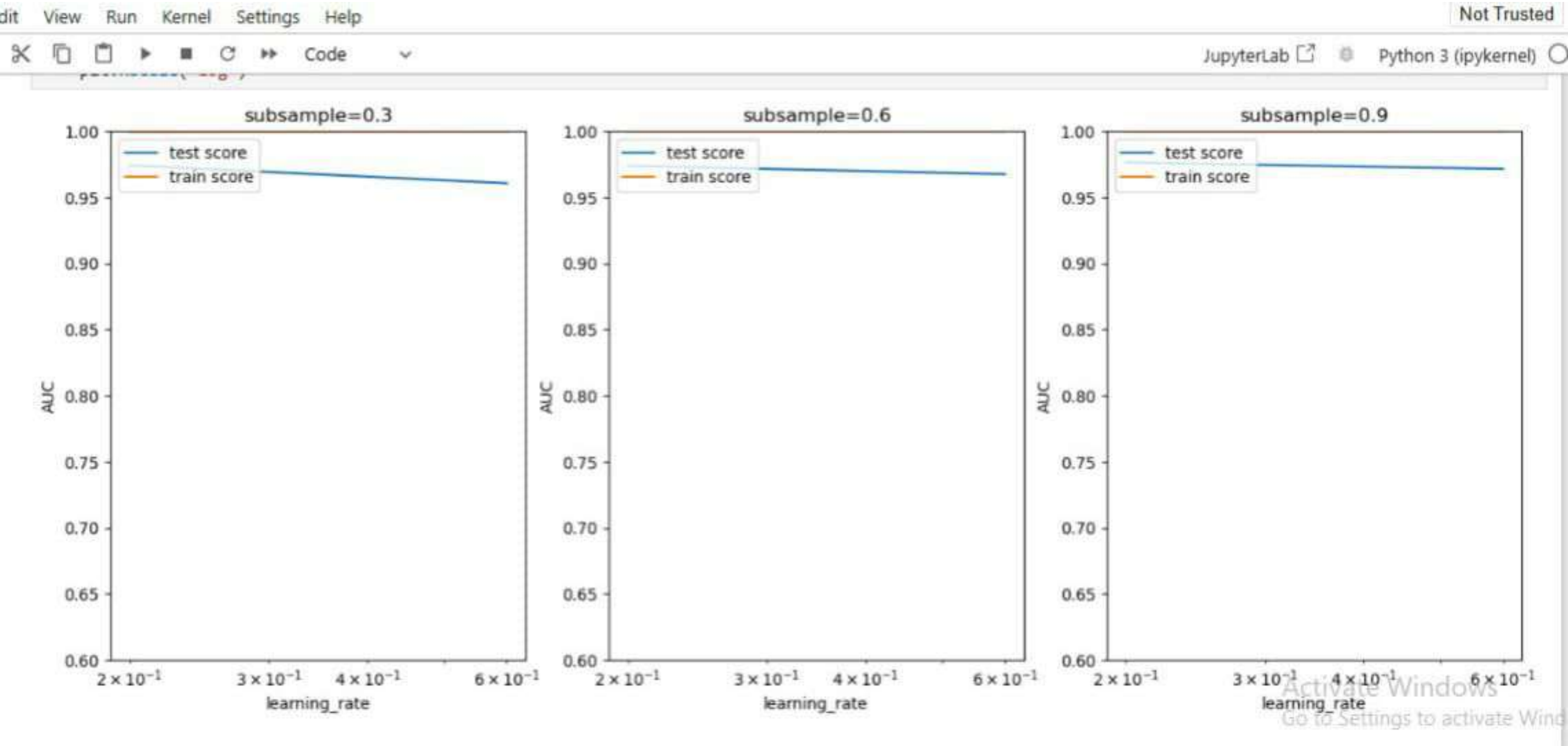

File
Edit
View
Run
Kernel
Settings
Help
Code
Receiver operating characteristic example
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate or [1 - True Negative Rate]
ROC curve (area = 0.92)

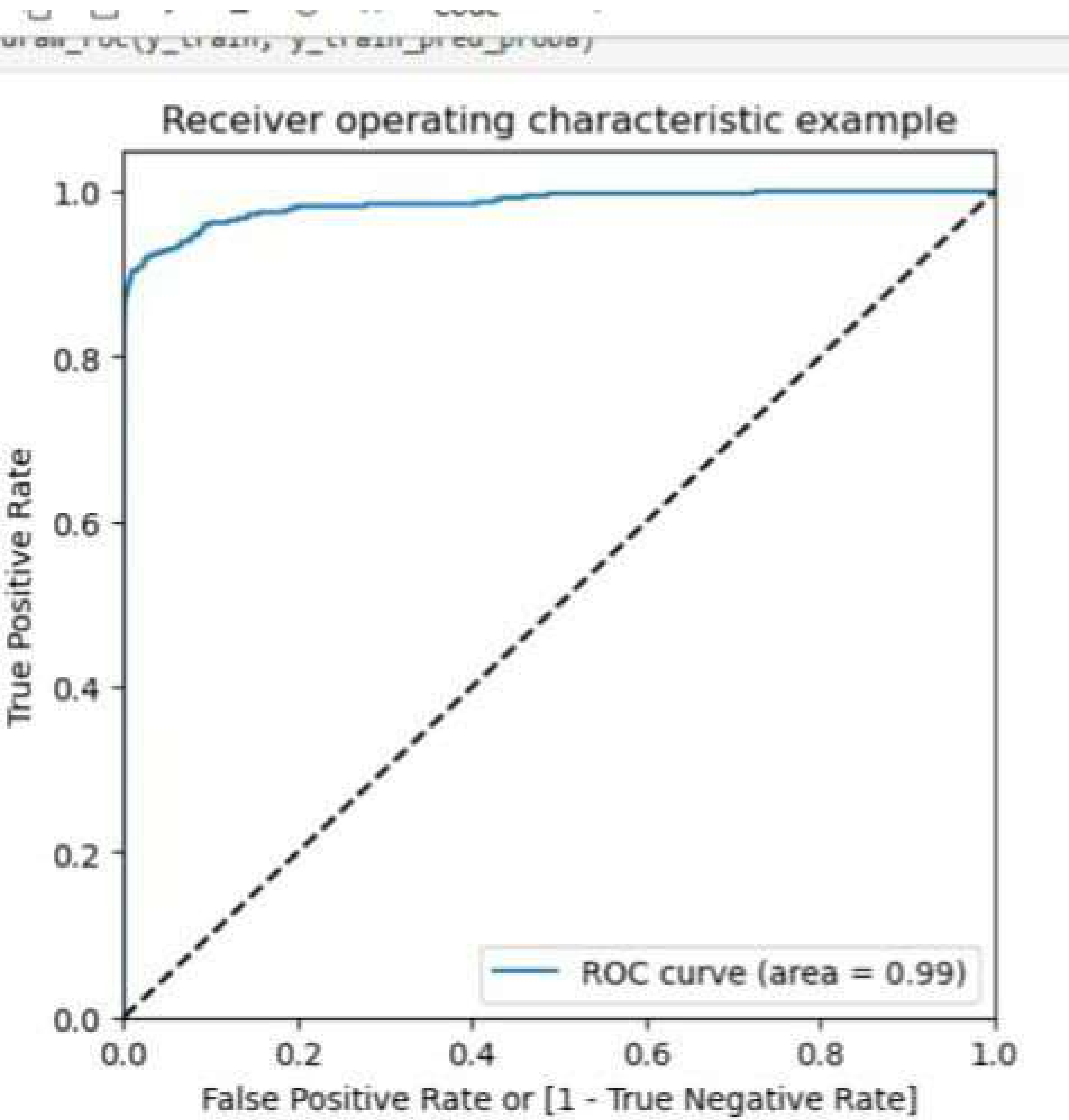

File
Edit
View
Run
Kernel
Settings
Help
Code
Receiver operating characteristic example
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate or [1 - True Negative Rate]
ROC curve (area = 1.00)


File
Edit
View
Run
Kernel
Settings
Help
Code
Receiver operating characteristic example
True Positive Rate
False Positive Rate or [1 - True Negative Rate]
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.97)

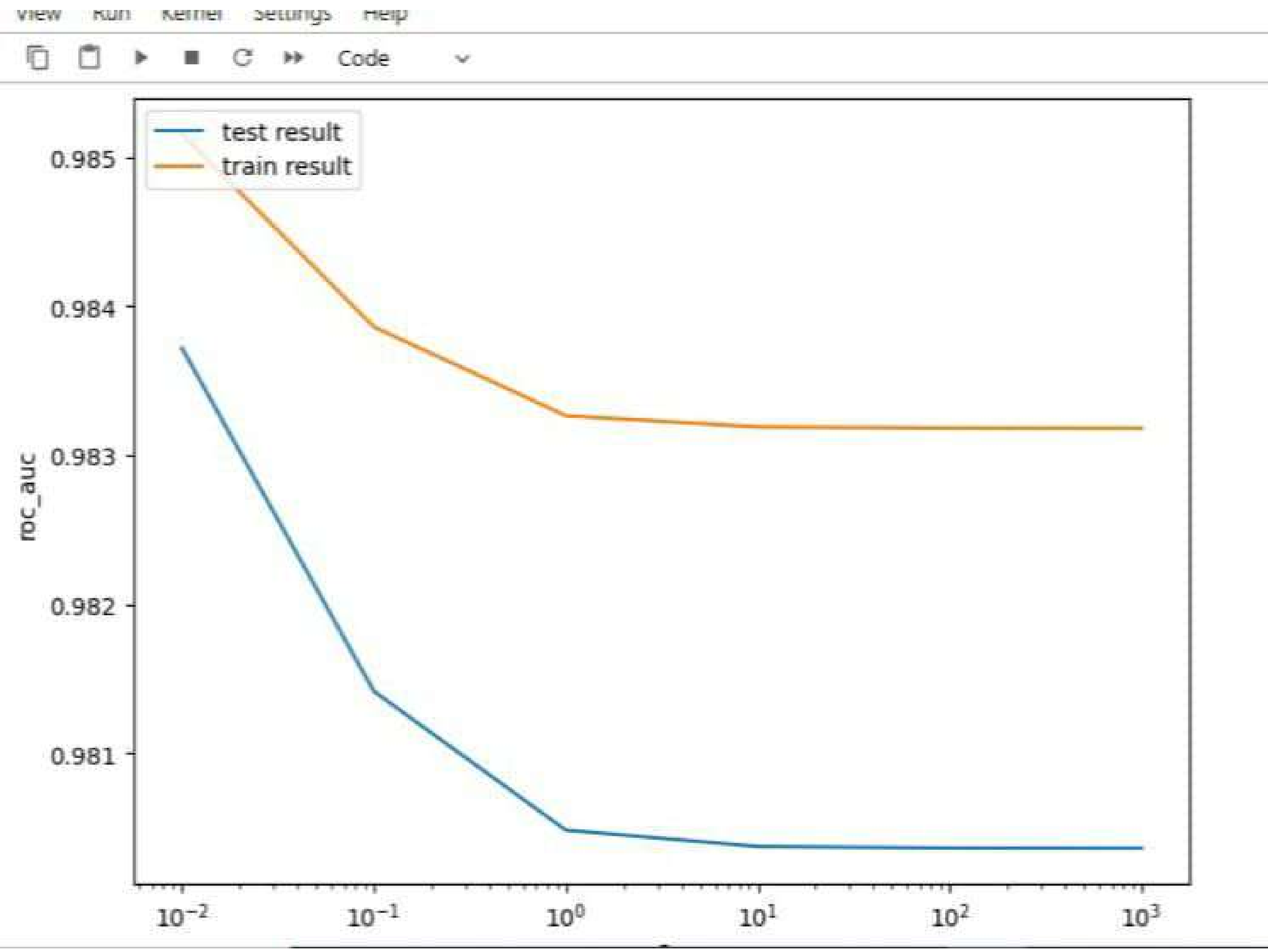


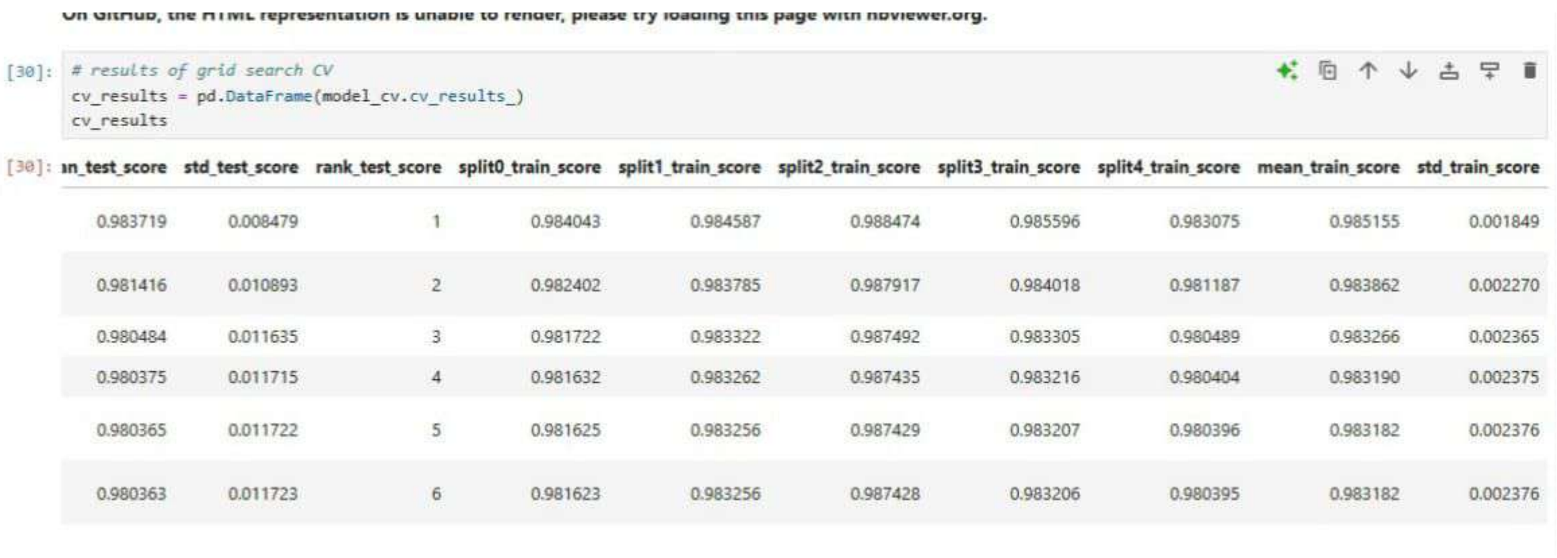
We can see that we have very good ROC on the test set 0.97, which is almost close to 1.


dit
View
Run
Kernel
Settings
Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)
subsample=0.3
subsample=0.6
subsample=0.9
test score
train score
AUC
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
2×10^{-1}
3×10^{-1}
4×10^{-1}
6×10^{-1}
learning_rate
Activate Windows
Go to Settings to activate Wind



```
draw_roc(y_train, y_train_pred_proba)
```


View
Run
Kernel
Settings
Help
Code
test result
train result
roc_auc
0.985
0.984
0.983
0.982
0.981
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}

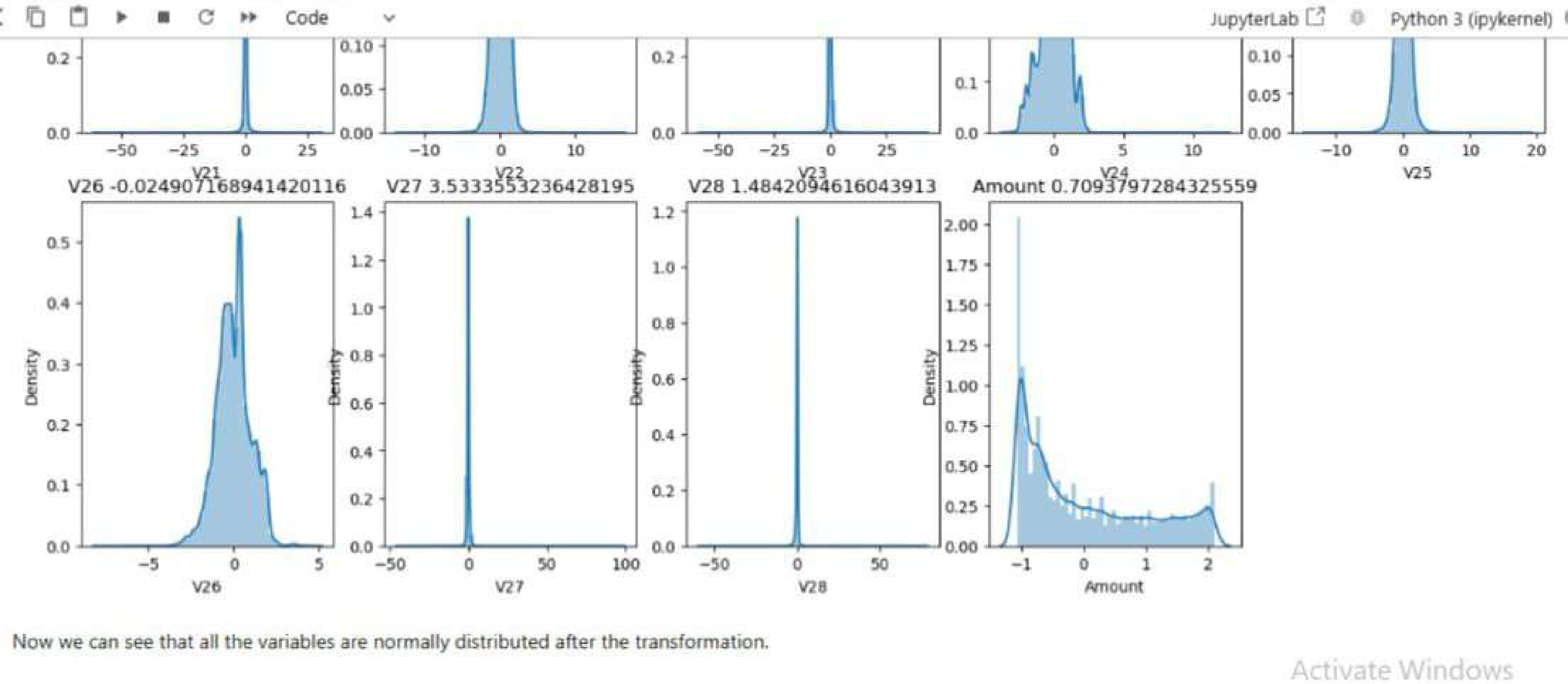


On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

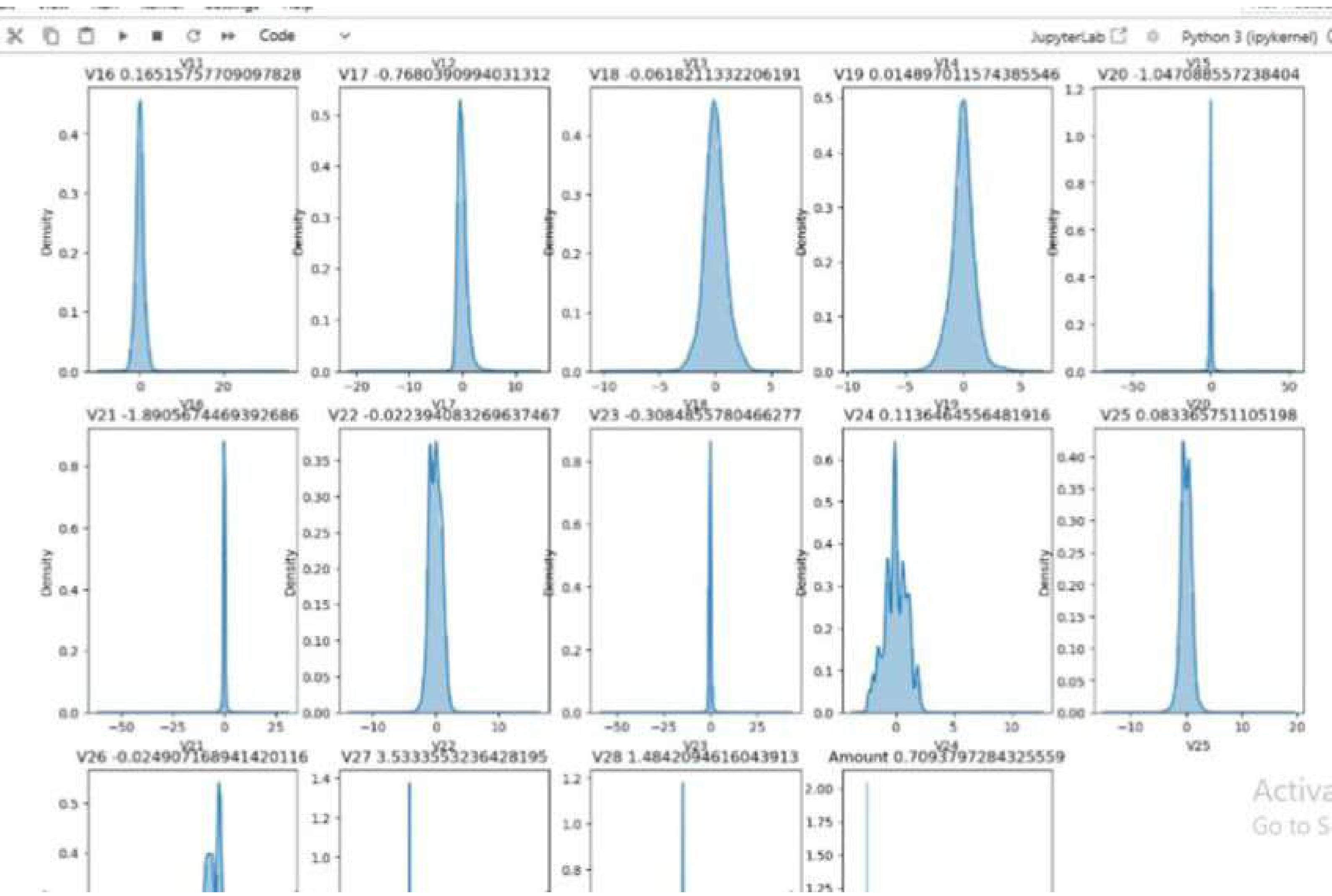
[30]:
```
# results of grid search CV
cv_results = pd.DataFrame(model_cv.cv_results_)
cv_results
```

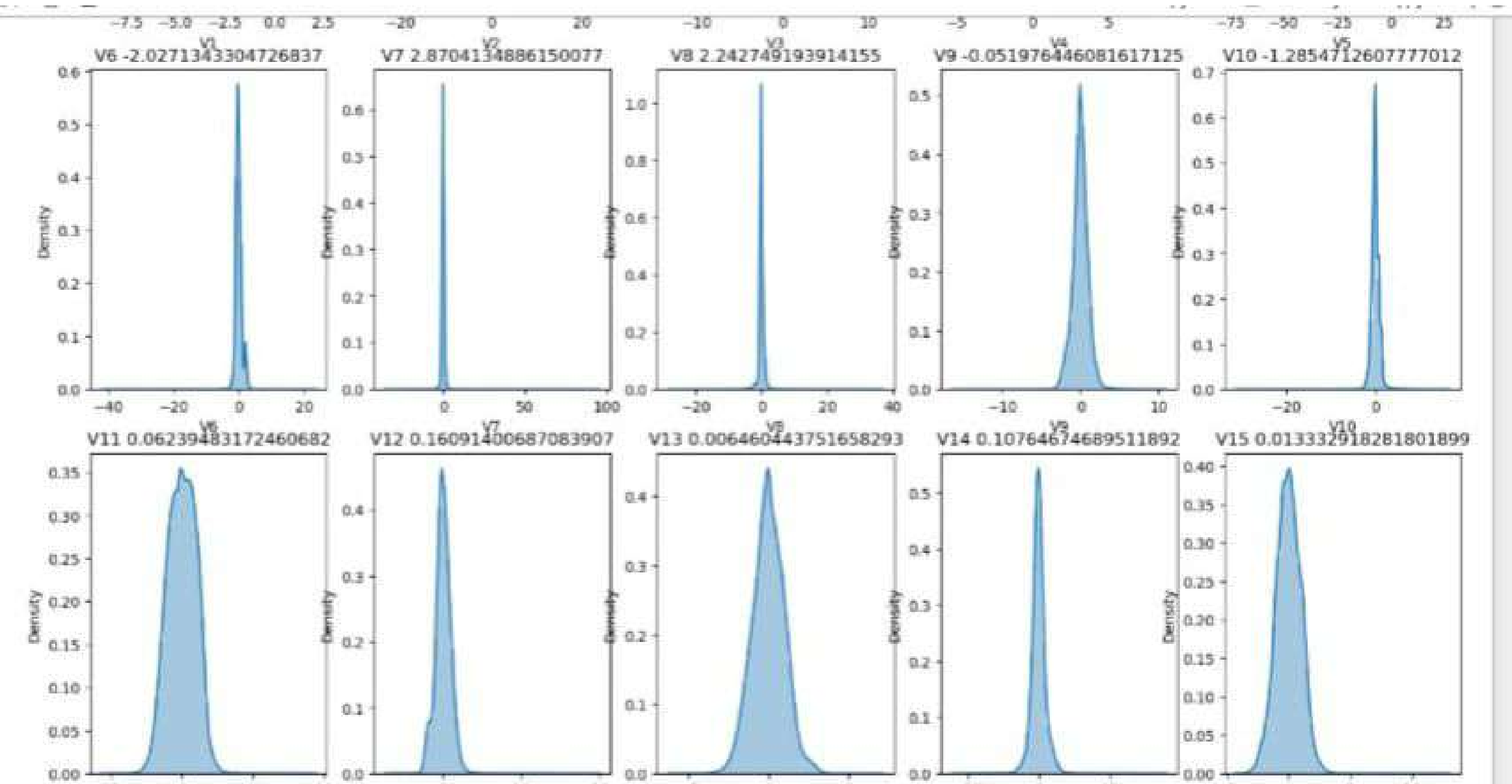
[30]:

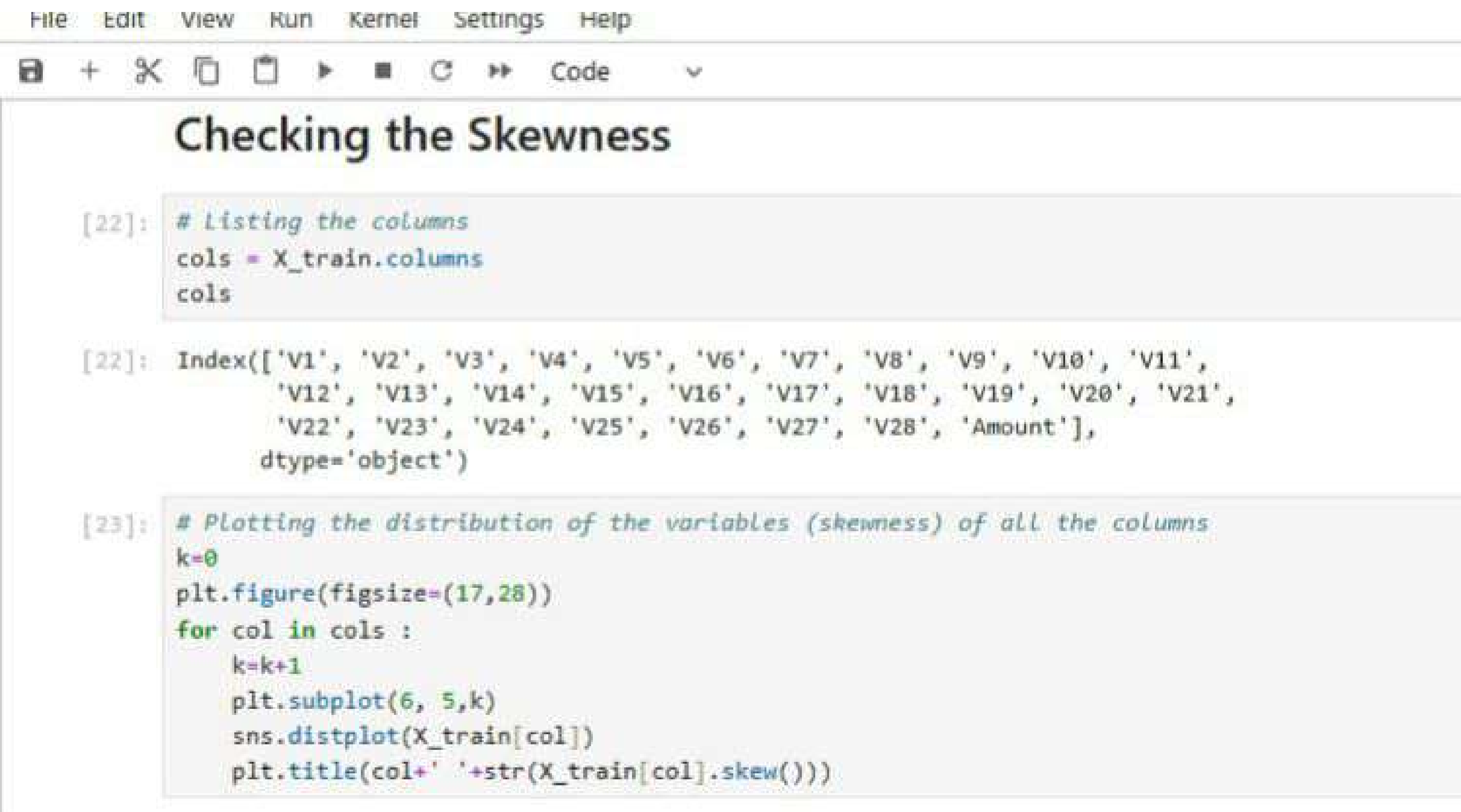
ın_test_score	std_test_score	rank_test_score	split0_train_score	split1_train_score	split2_train_score	split3_train_score	split4_train_score	mean_train_score	std_train_score
0.983719	0.008479	1	0.984043	0.984587	0.988474	0.985596	0.983075	0.985155	0.001849
0.981416	0.010893	2	0.982402	0.983785	0.987917	0.984018	0.981187	0.983862	0.002270
0.980484	0.011635	3	0.981722	0.983322	0.987492	0.983305	0.980489	0.983266	0.002365
0.980375	0.011715	4	0.981632	0.983262	0.987435	0.983216	0.980404	0.983190	0.002375
0.980365	0.011722	5	0.981625	0.983256	0.987429	0.983207	0.980396	0.983182	0.002376
0.980363	0.011723	6	0.981623	0.983256	0.987428	0.983206	0.980395	0.983182	0.002376



Now we can see that all the variables are normally distributed after the transformation.

Code
JupyterLab
Python 3 (ipykernel)
V16 0.16515757709097828
V17 -0.7680390994031312
V18 -0.0618211332206191
V19 0.014897011574385546
V20 -1.047088557238404
Density
V16
V17
V18
V19
V20
V21 -1.8905674469392686
V22 -0.022394083269637467
V23 -0.3084855780466277
V24 0.1136464556481916
V25 0.083365751105198
V21
V22
V23
V24
V25
V26 -0.024907168941420116
V27 3.5333553236428195
V28 1.4842094616043913
Amount 0.7093797284325559
Activa
Go to S


−7.5 −5.0 −2.5 0.0 2.5
V1
−20 0 20
V2
−10 0 10
V3
−5 0 5
V4
−75 −50 −25 0 25
V5
V6 -2.0271343304726837
Density
0.0 0.1 0.2 0.3 0.4 0.5 0.6
−40 −20 0 20
V6
V7 2.8704134886150077
Density
0.0 0.1 0.2 0.3 0.4 0.5 0.6
0 50 100
V7
V8 2.242749193914155
Density
0.0 0.2 0.4 0.6 0.8 1.0
−20 0 20 40
V8
V9 -0.051976446081617125
Density
0.0 0.1 0.2 0.3 0.4 0.5
−10 0 10
V9
V10 -1.2854712607777012
Density
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
−20 0
V10
V11 0.06239483172460682
Density
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35
−5 0 5 10
V12 0.16091400687083907
Density
0.0 0.1 0.2 0.3 0.4
0 10 20
V13 0.006460443751658293
Density
0.0 0.1 0.2 0.3 0.4
−5 0 5
V14 0.10764674689511892
Density
0.0 0.1 0.2 0.3 0.4 0.5
−10 0 10
V15 0.013332918281801899
Density
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40
−5 0 5 10

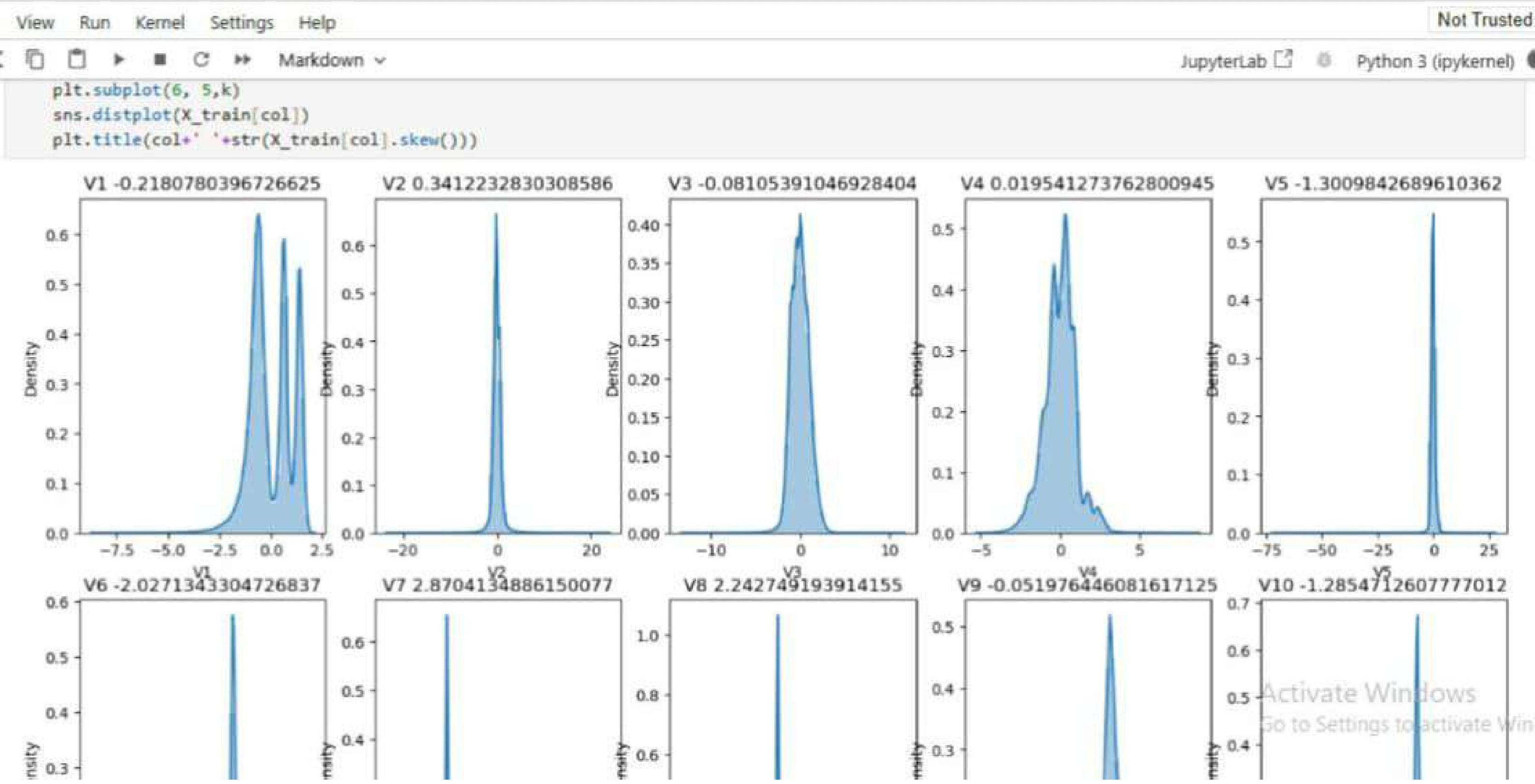


Checking the Skewness

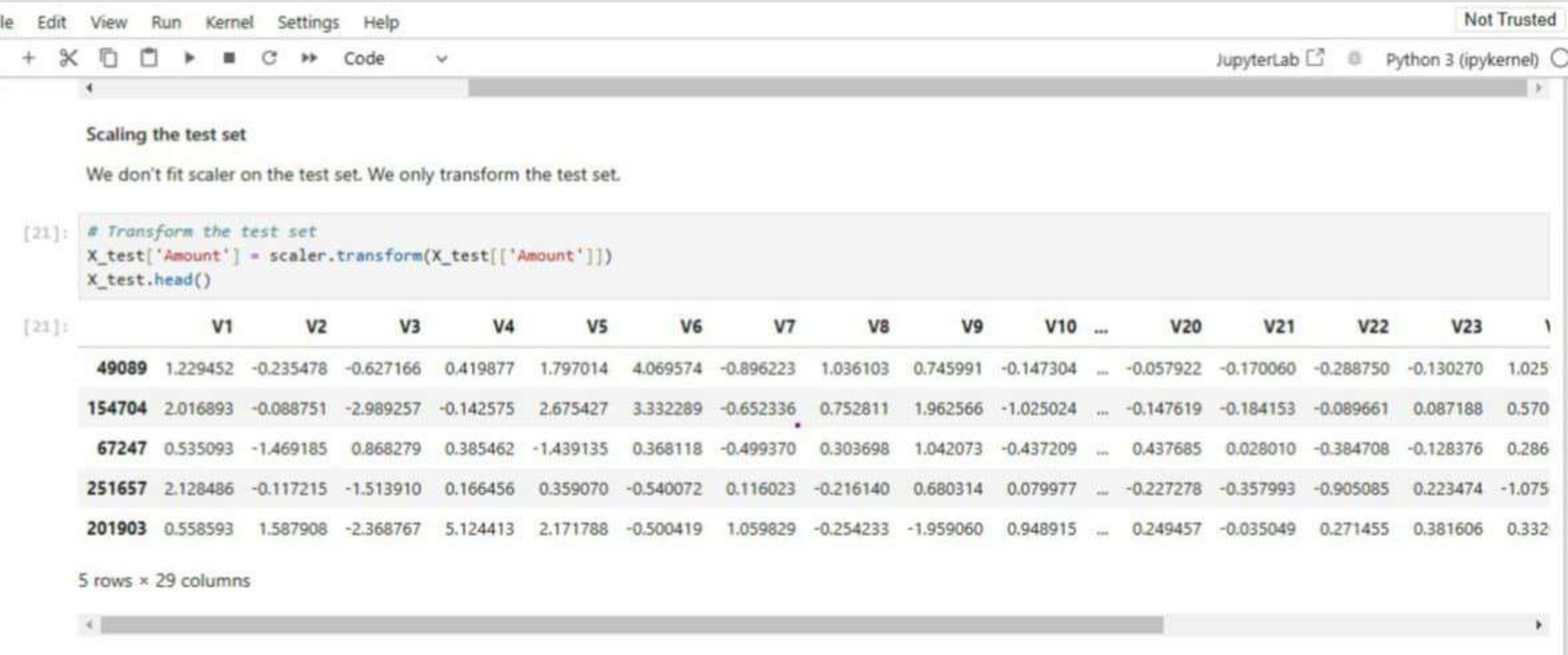
```
[22]: # Listing the columns
      cols = X_train.columns
      cols
```

```
[22]: Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11',
             'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21',
             'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount'],
            dtype='object')
```

```
[23]: # Plotting the distribution of the variables (skewness) of all the columns
      k=0
      plt.figure(figsize=(17,28))
      for col in cols :
          k=k+1
          plt.subplot(6, 5,k)
          sns.distplot(X_train[col])
          plt.title(col+' '+str(X_train[col].skew()))
```



```
    plt.subplot(6, 5,k)
    sns.distplot(X_train[col])
    plt.title(col+' '+str(X_train[col].skew()))
```



Scaling the test set

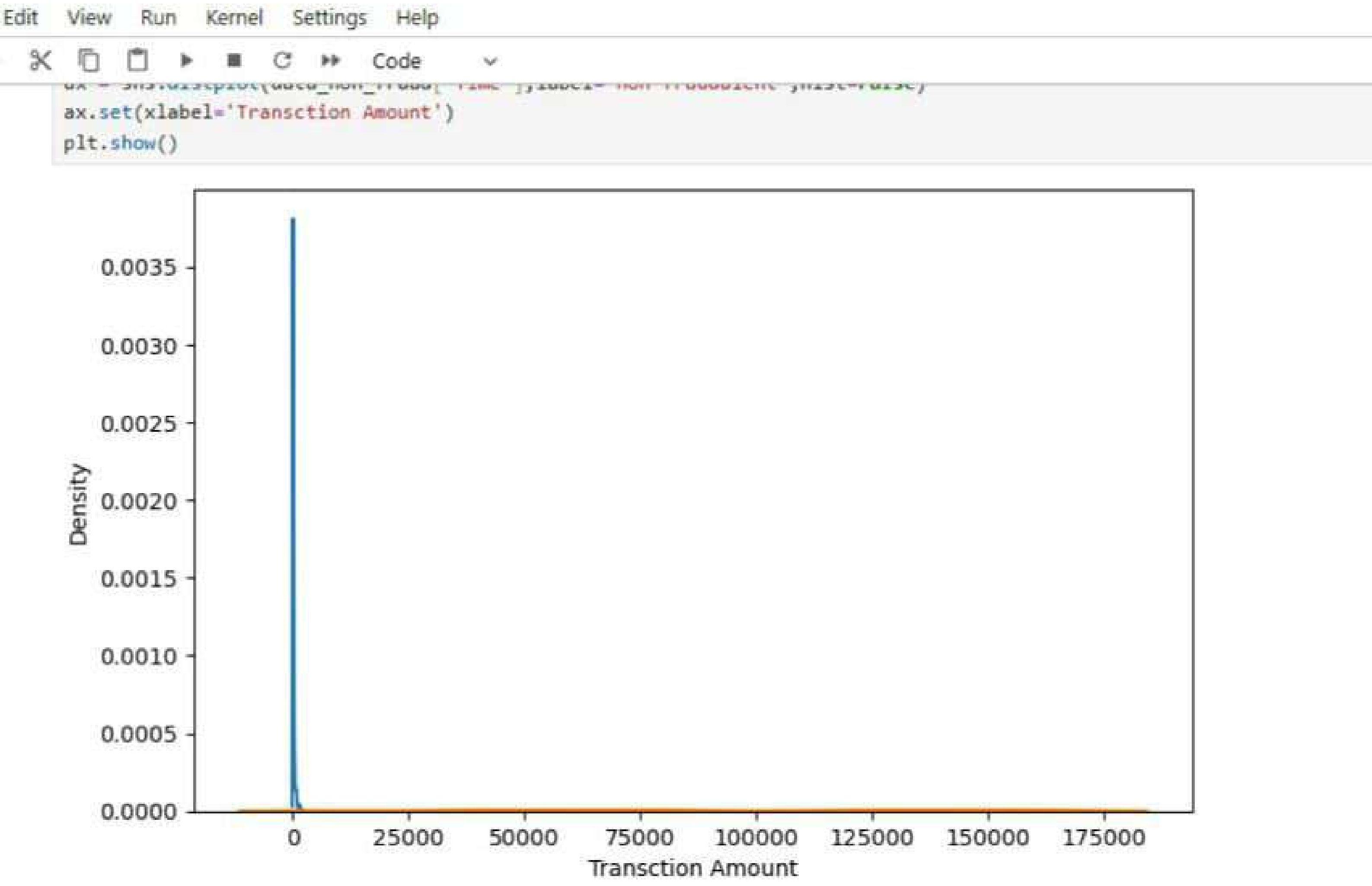
We don't fit scaler on the test set. We only transform the test set.

```
[21]: # Transform the test set
      X_test['Amount'] = scaler.transform(X_test[['Amount']])
      X_test.head()
```

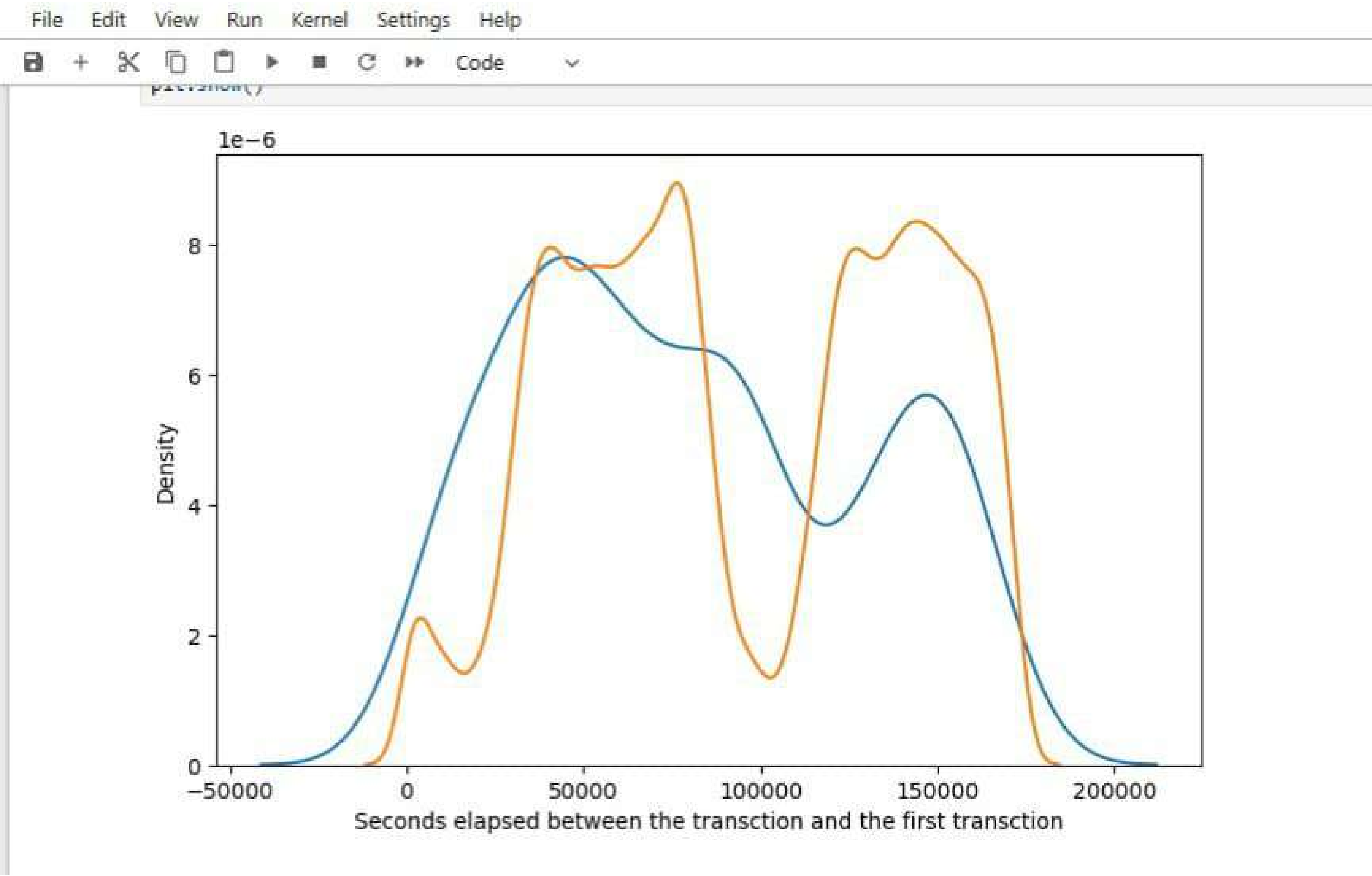
[21]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V20	V21	V22	V23	V
49089	1.229452	-0.235478	-0.627166	0.419877	1.797014	4.069574	-0.896223	1.036103	0.745991	-0.147304	...	-0.057922	-0.170060	-0.288750	-0.130270	1.025
154704	2.016893	-0.088751	-2.989257	-0.142575	2.675427	3.332289	-0.652336	0.752811	1.962566	-1.025024	...	-0.147619	-0.184153	-0.089661	0.087188	0.570
67247	0.535093	-1.469185	0.868279	0.385462	-1.439135	0.368118	-0.499370	0.303698	1.042073	-0.437209	...	0.437685	0.028010	-0.384708	-0.128376	0.286
251657	2.128486	-0.117215	-1.513910	0.166456	0.359070	-0.540072	0.116023	-0.216140	0.680314	0.079977	...	-0.227278	-0.357993	-0.905085	0.223474	-1.075
201903	0.558593	1.587908	-2.368767	5.124413	2.171788	-0.500419	1.059829	-0.254233	-1.959060	0.948915	...	0.249457	-0.035049	0.271455	0.381606	0.332

5 rows × 29 columns

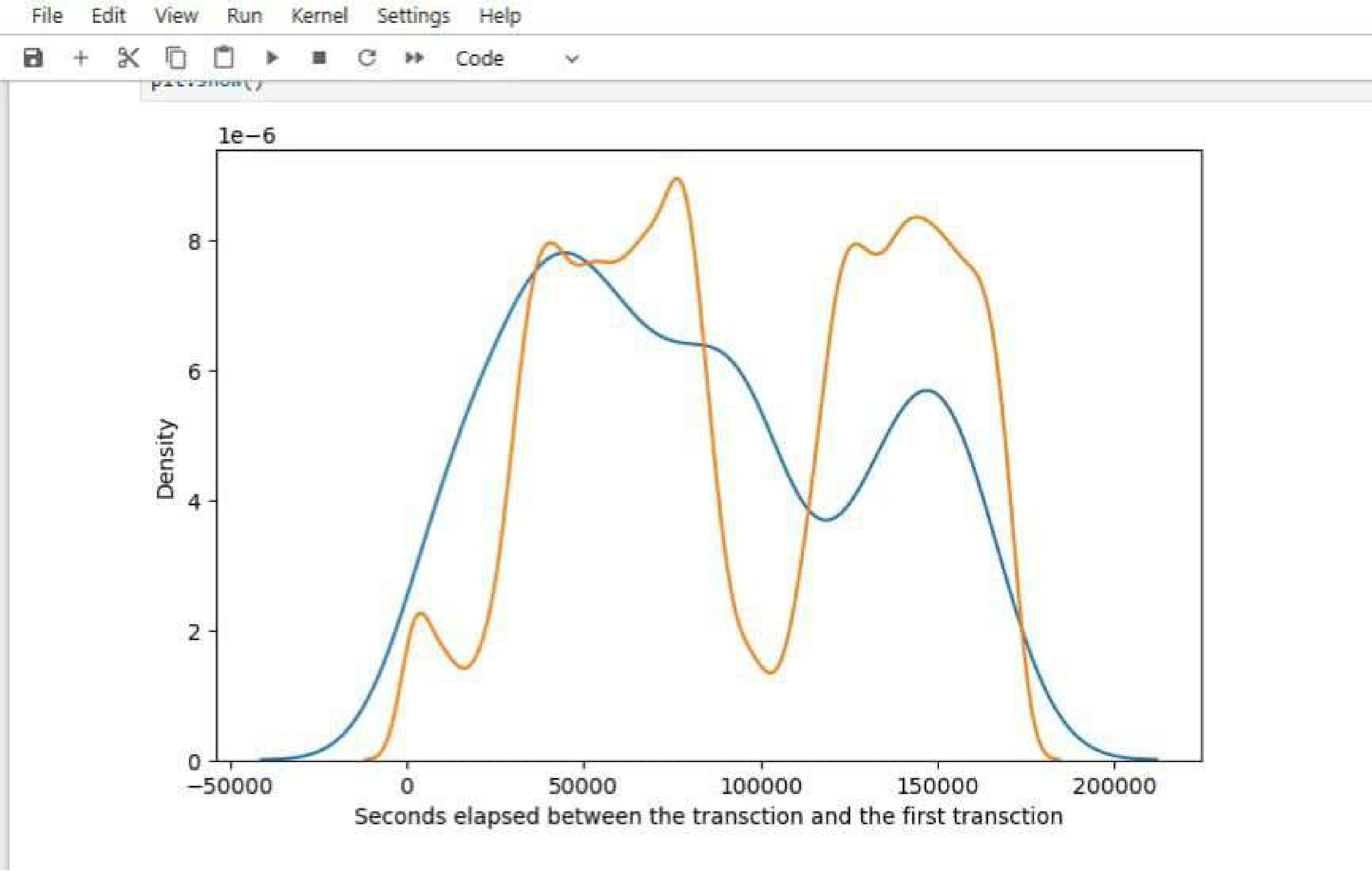


```
ax = sns.distplot(data_non_fraud['Time'],label='Non Fraudulent',hist=False)
ax.set(xlabel='Transction Amount')
plt.show()
```



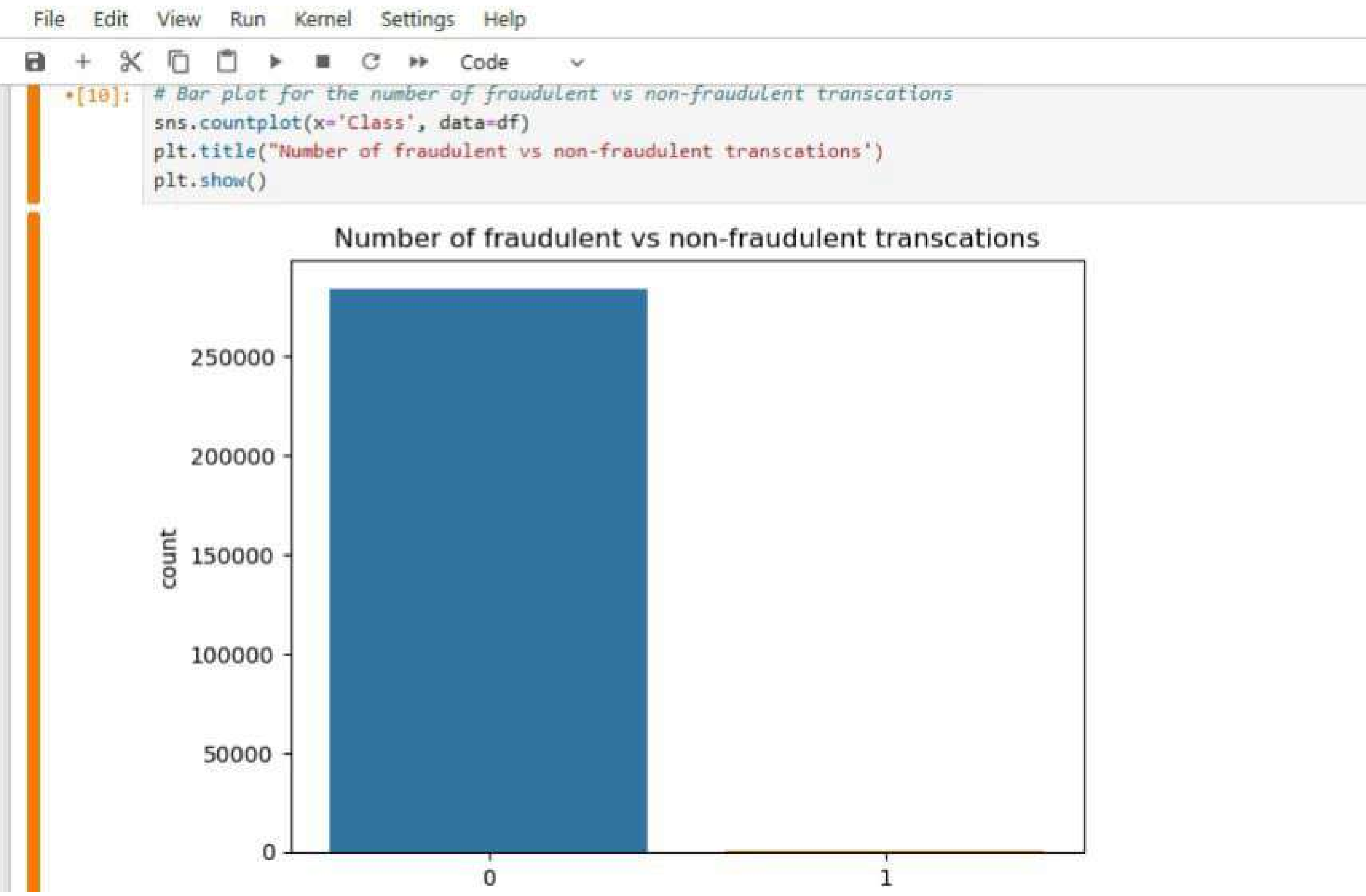
```
plt.show()
```

File
Edit
View
Run
Kernel
Settings
Help
Code
1e−6
Density
8
6
4
2
0
−50000
0
50000
100000
150000
200000
Seconds elapsed between the transction and the first transction

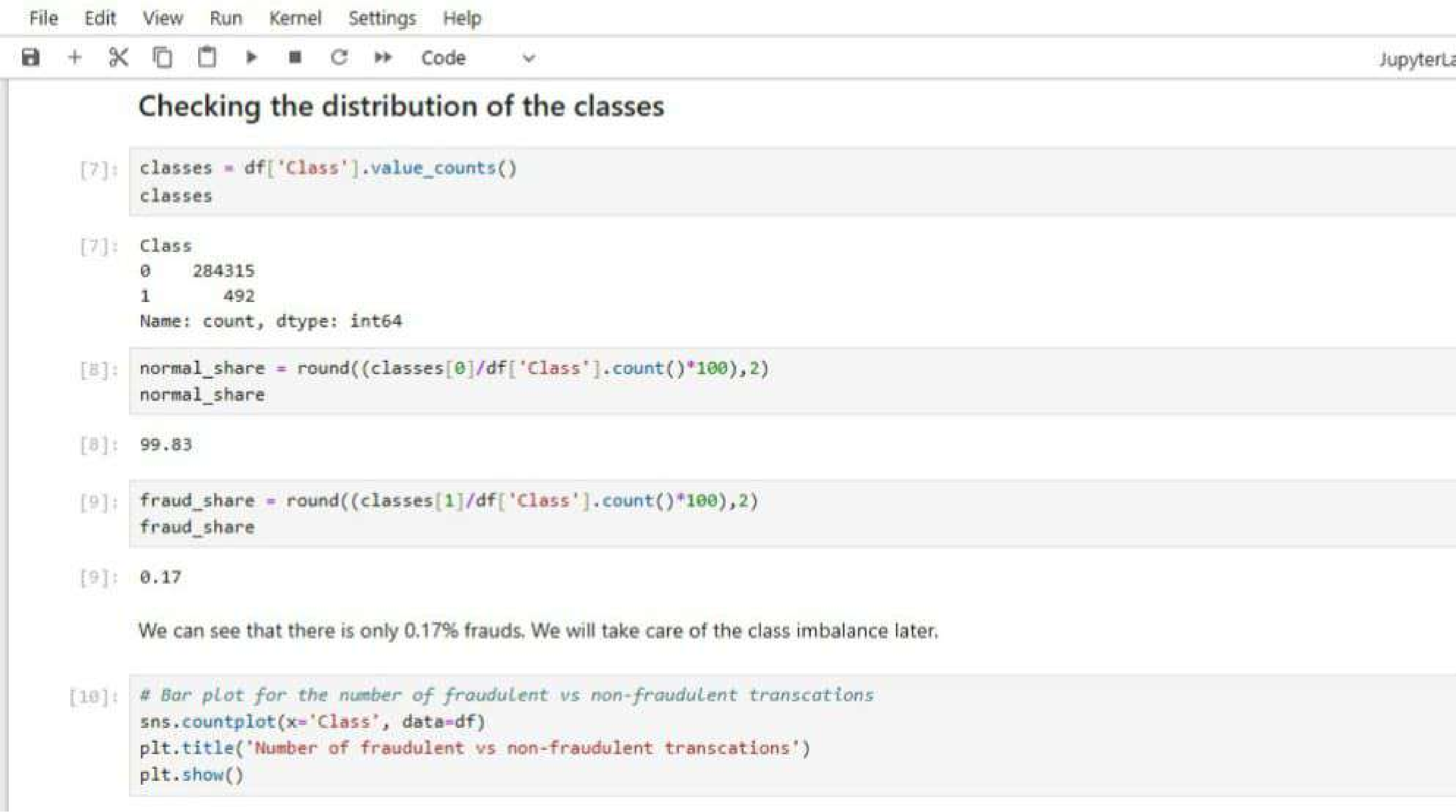


File
Edit
View
Run
Kernel
Settings
Help
Code
1e−6
Density
0
2
4
6
8
−50000
0
50000
100000
150000
200000
Seconds elapsed between the transction and the first transction

```
plt.show()
```



```
# Bar plot for the number of fraudulent vs non-fraudulent transcations
sns.countplot(x='Class', data=df)
plt.title("Number of fraudulent vs non-fraudulent transcations')
plt.show()
```



Checking the distribution of the classes

```
[7]:  classes = df['Class'].value_counts()
      classes
```

```
[7]:  Class
      0    284315
      1       492
      Name: count, dtype: int64
```

```
[8]:  normal_share = round((classes[0]/df['Class'].count()*100),2)
      normal_share
```

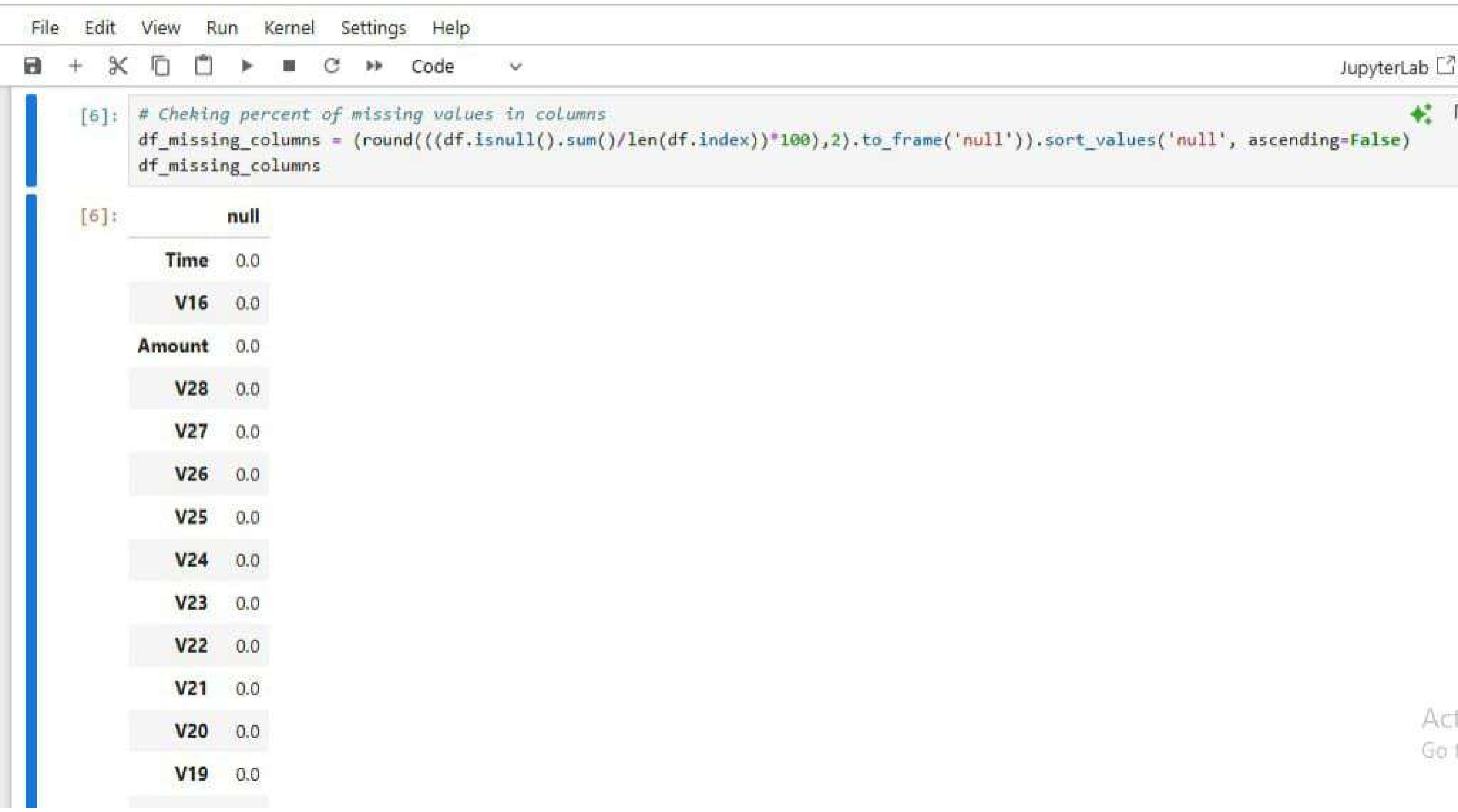
```
[8]:  99.83
```

```
[9]:  fraud_share = round((classes[1]/df['Class'].count()*100),2)
      fraud_share
```

```
[9]:  0.17
```

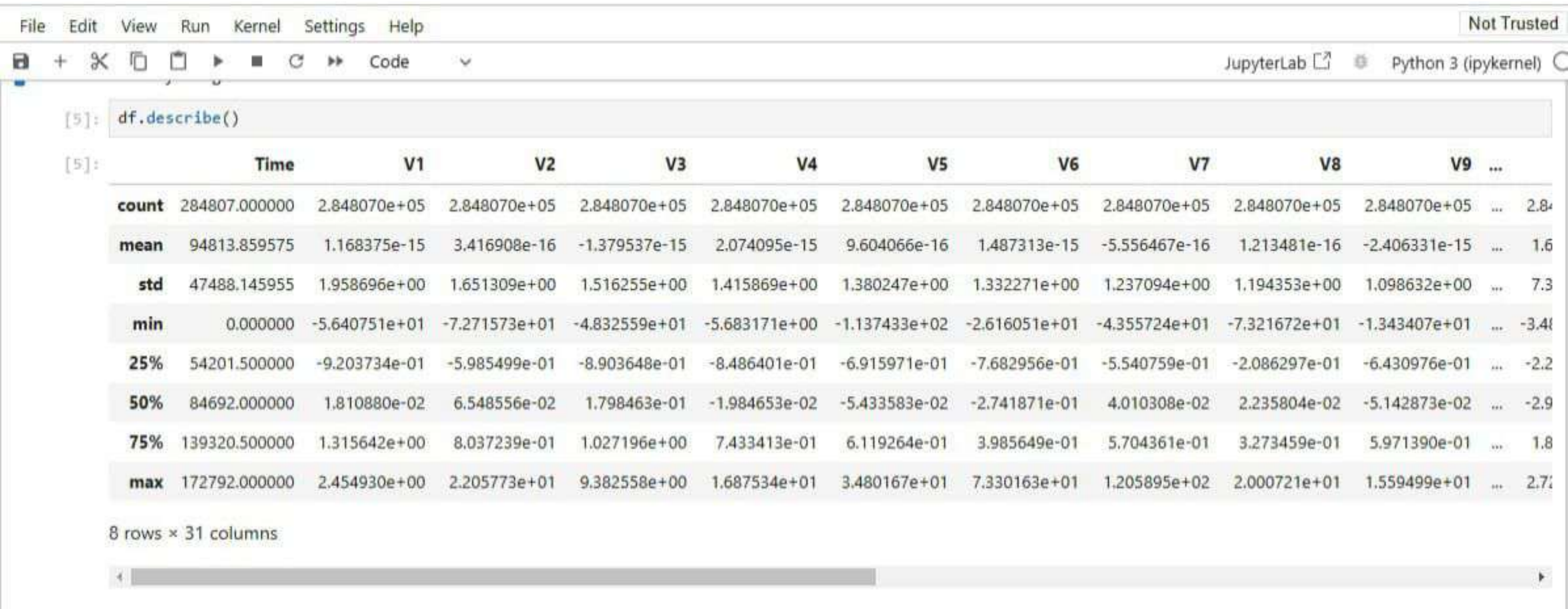
We can see that there is only 0.17% frauds. We will take care of the class imbalance later.

```
[10]:  # Bar plot for the number of fraudulent vs non-fraudulent transcations
       sns.countplot(x='Class', data=df)
       plt.title('Number of fraudulent vs non-fraudulent transcations')
       plt.show()
```



```
[6]: # Cheking percent of missing values in columns
     df_missing_columns = (round(((df.isnull().sum()/len(df.index))*100),2).to_frame('null')).sort_values('null', ascending=False)
     df_missing_columns
```

```
[6]:          null
       Time    0.0
        V16    0.0
     Amount    0.0
        V28    0.0
        V27    0.0
        V26    0.0
        V25    0.0
        V24    0.0
        V23    0.0
        V22    0.0
        V21    0.0
        V20    0.0
        V19    0.0
```

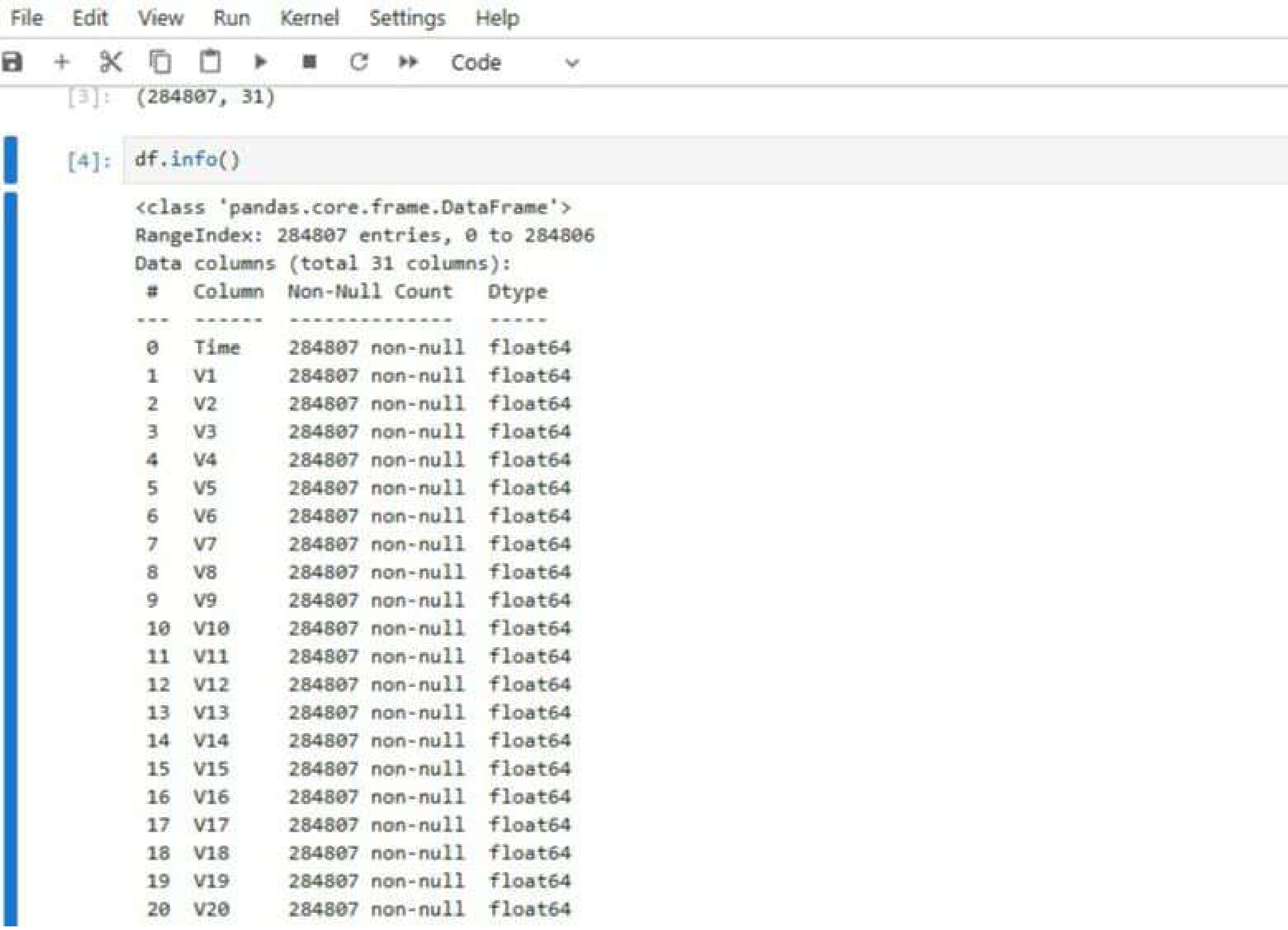


[5]:
```
df.describe()
```

[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.84
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	1.6
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.3
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.48
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.2
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.9
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.8
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.72

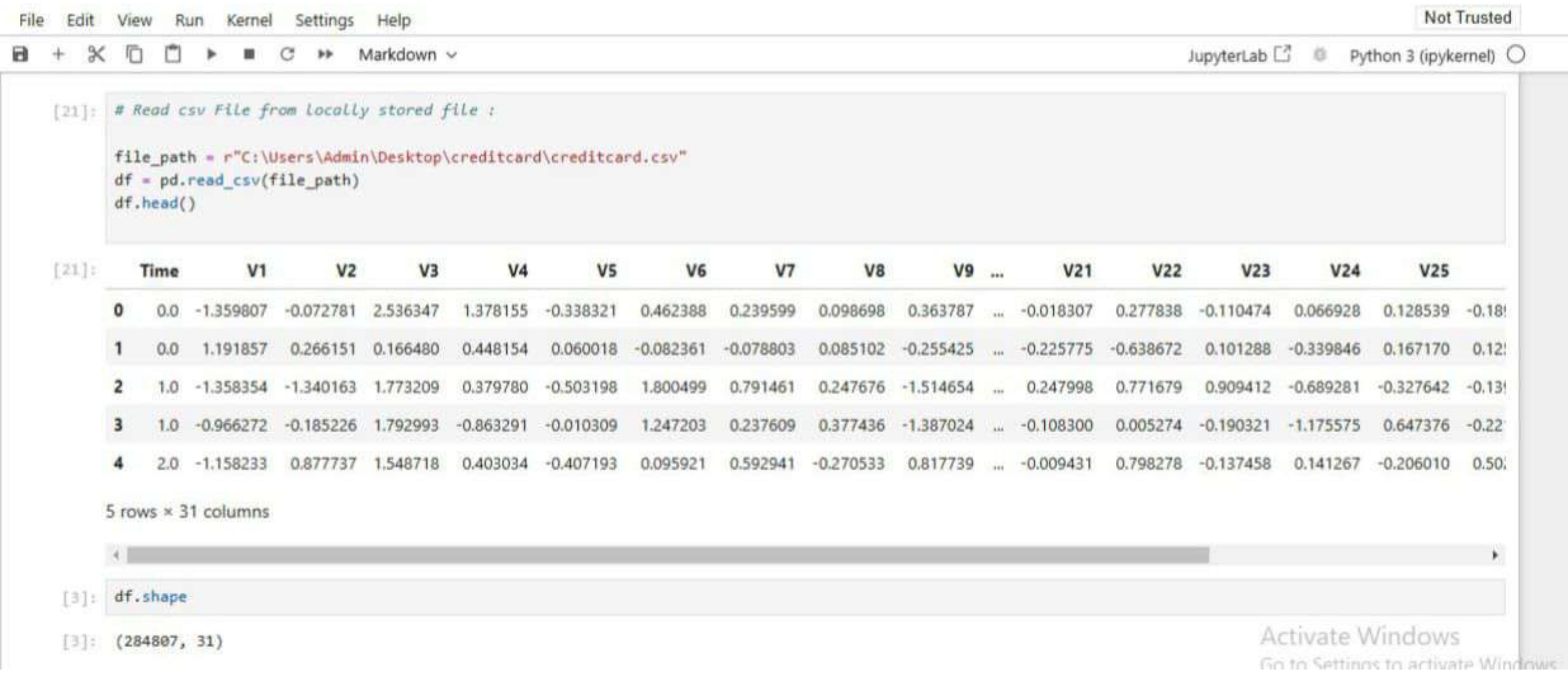
8 rows × 31 columns



```
[3]: (284807, 31)
```

```
[4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
```



```
[21]: # Read csv File from locally stored file :

      file_path = r"C:\Users\Admin\Desktop\creditcard\creditcard.csv"
      df = pd.read_csv(file_path)
      df.head()
```

[21]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.18
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.12
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.13
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.22
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.50

5 rows × 31 columns

```
[3]: df.shape
```

```
[3]: (284807, 31)
```