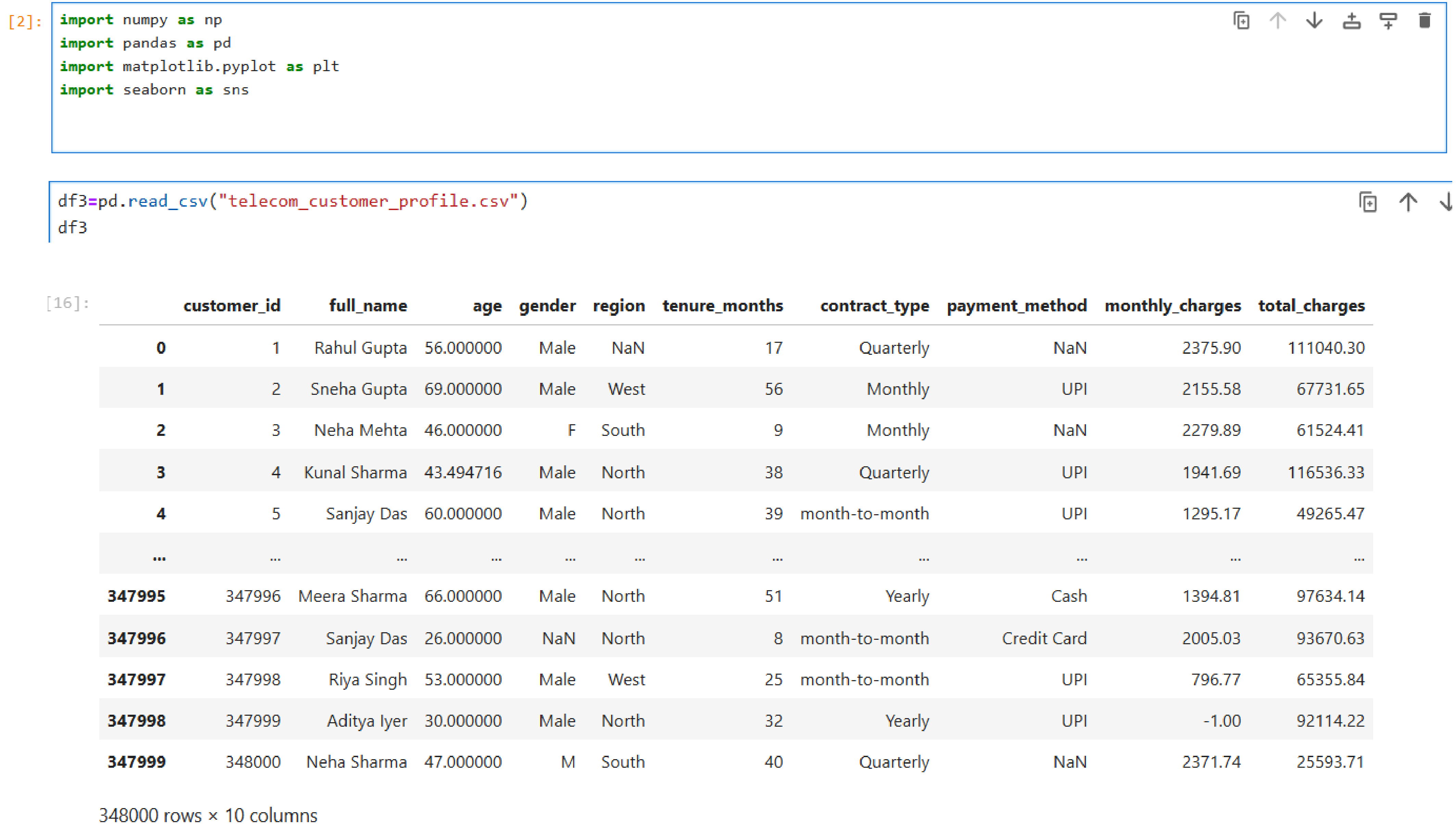


[2]:

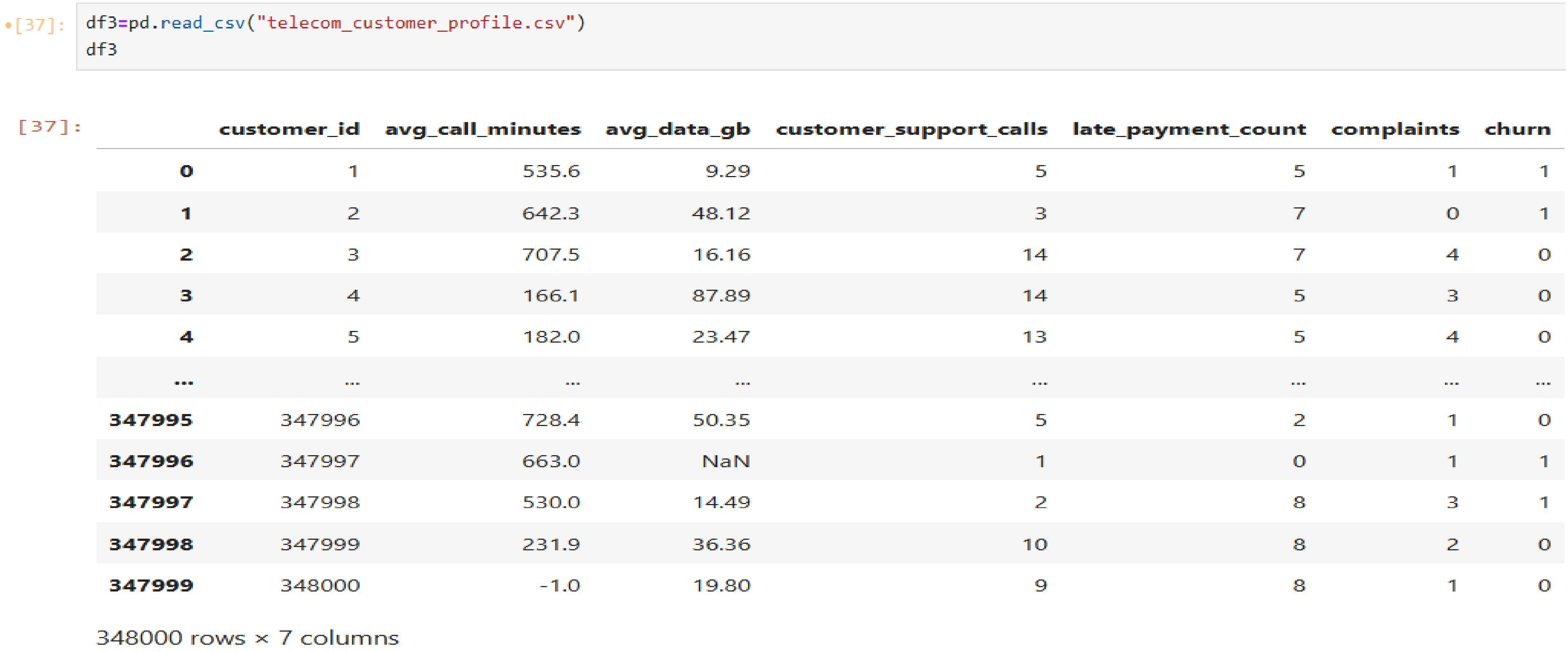
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
df3=pd.read_csv("telecom_customer_profile.csv")
df3
```

[16]:

	customer_id	full_name	age	gender	region	tenure_months	contract_type	payment_method	monthly_charges	total_charges
0	1	Rahul Gupta	56.000000	Male	NaN	17	Quarterly	NaN	2375.90	111040.30
1	2	Sneha Gupta	69.000000	Male	West	56	Monthly	UPI	2155.58	67731.65
2	3	Neha Mehta	46.000000	F	South	9	Monthly	NaN	2279.89	61524.41
3	4	Kunal Sharma	43.494716	Male	North	38	Quarterly	UPI	1941.69	116536.33
4	5	Sanjay Das	60.000000	Male	North	39	month-to-month	UPI	1295.17	49265.47
...	...	...	...	...	...	...	...	...	...	...
347995	347996	Meera Sharma	66.000000	Male	North	51	Yearly	Cash	1394.81	97634.14
347996	347997	Sanjay Das	26.000000	NaN	North	8	month-to-month	Credit Card	2005.03	93670.63
347997	347998	Riya Singh	53.000000	Male	West	25	month-to-month	UPI	796.77	65355.84
347998	347999	Aditya Iyer	30.000000	Male	North	32	Yearly	UPI	-1.00	92114.22
347999	348000	Neha Sharma	47.000000	M	South	40	Quarterly	NaN	2371.74	25593.71

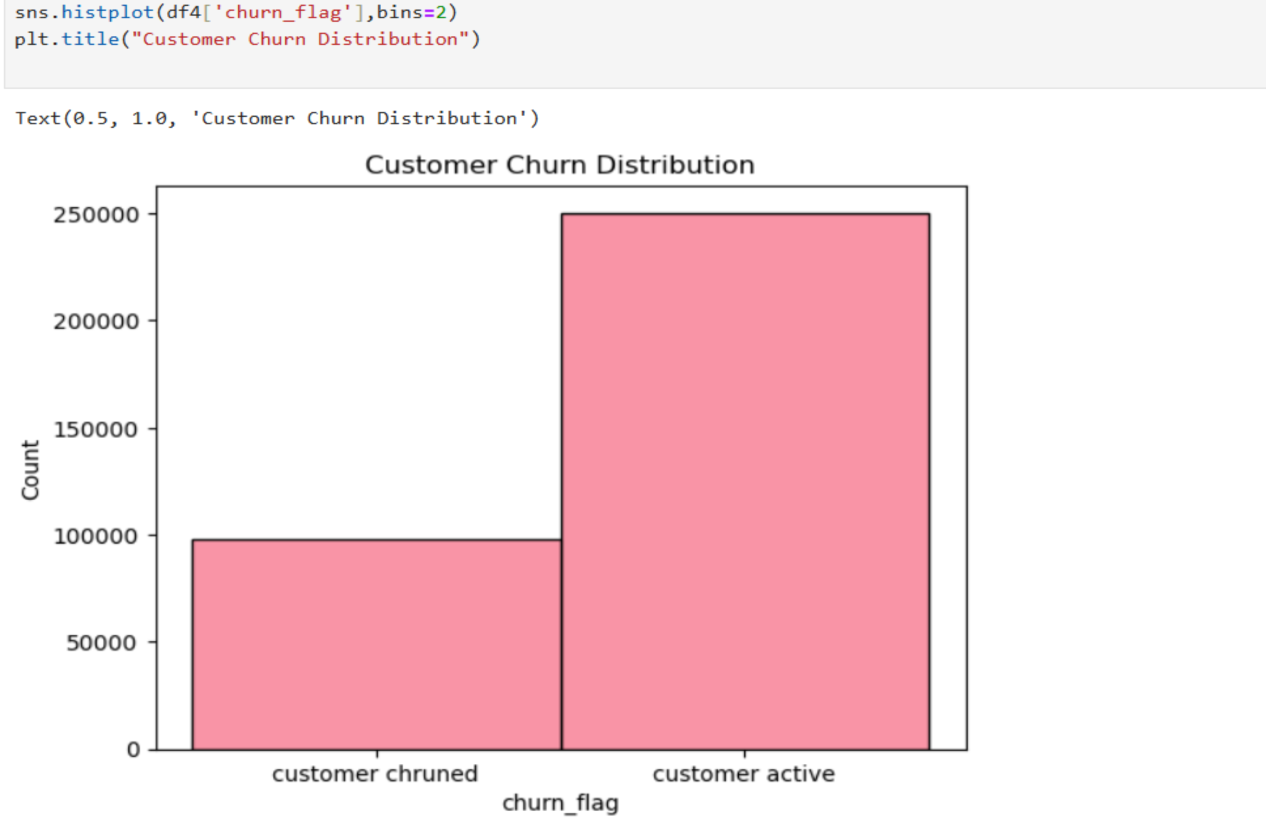
348000 rows × 10 columns



```
df3=pd.read_csv("telecom_customer_profile.csv")
df3
```

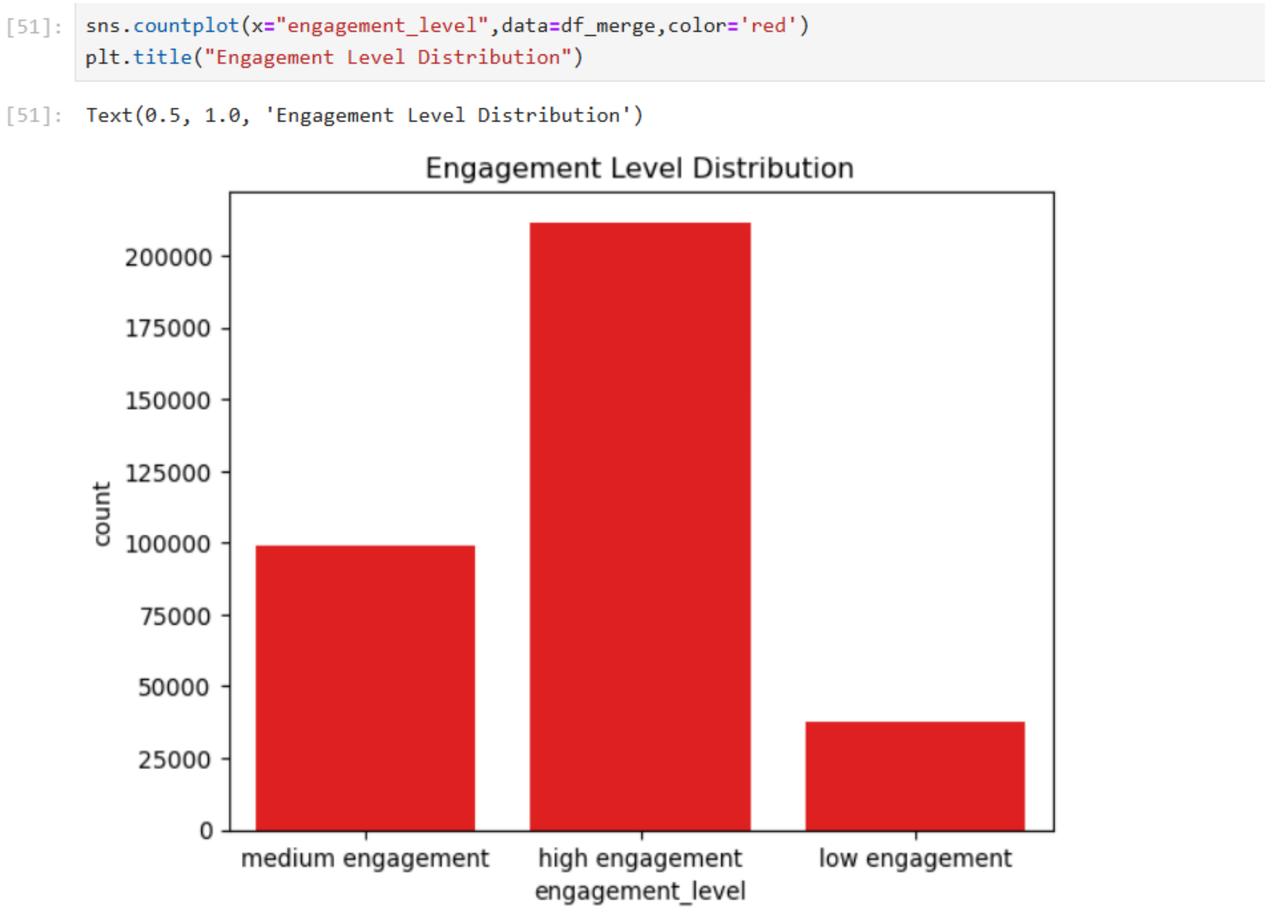
	customer_id	avg_call_minutes	avg_data_gb	customer_support_calls	late_payment_count	complaints	churn
0	1	535.6	9.29	5	5	1	1
1	2	642.3	48.12	3	7	0	1
2	3	707.5	16.16	14	7	4	0
3	4	166.1	87.89	14	5	3	0
4	5	182.0	23.47	13	5	4	0
...	...	...	...	...	...	...	...
347995	347996	728.4	50.35	5	2	1	0
347996	347997	663.0	NaN	1	0	1	1
347997	347998	530.0	14.49	2	8	3	1
347998	347999	231.9	36.36	10	8	2	0
347999	348000	-1.0	19.80	9	8	1	0

348000 rows × 7 columns



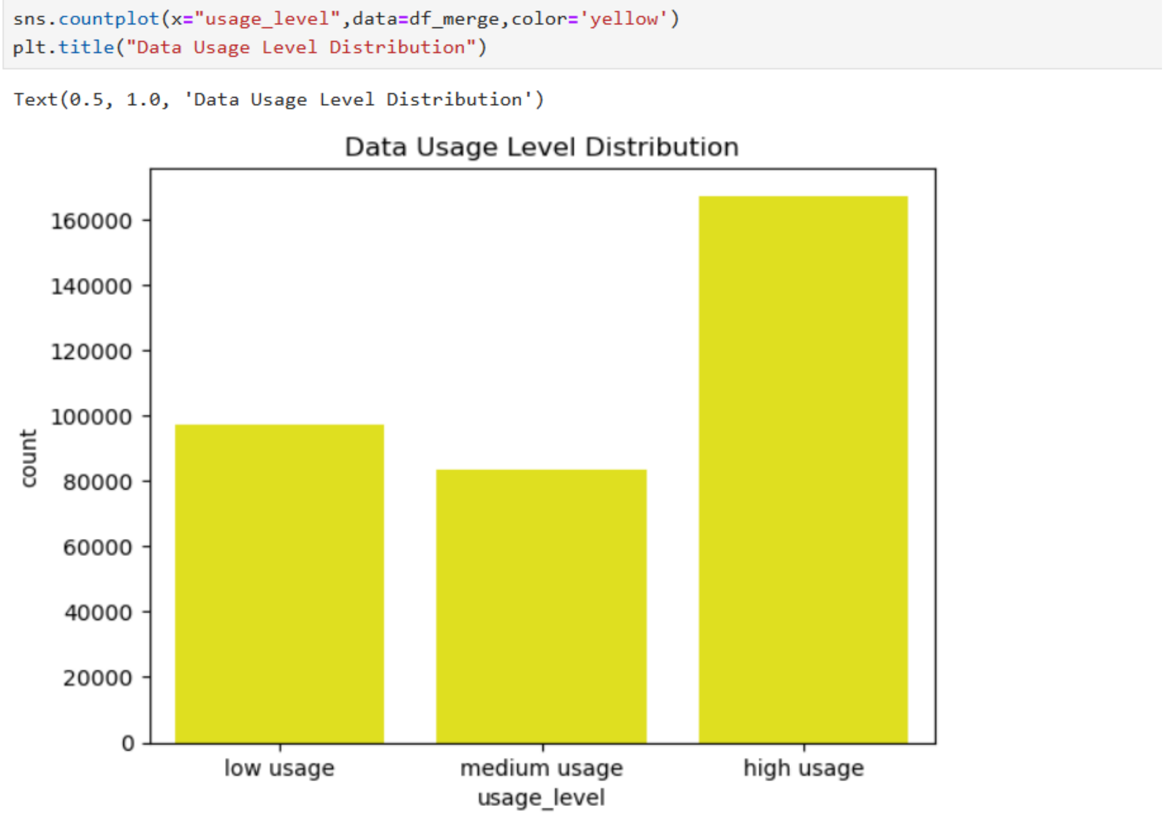
```
sns.histplot(df4['churn_flag'],bins=2)
plt.title("Customer Churn Distribution")
```

```
Text(0.5, 1.0, 'Customer Churn Distribution')
```



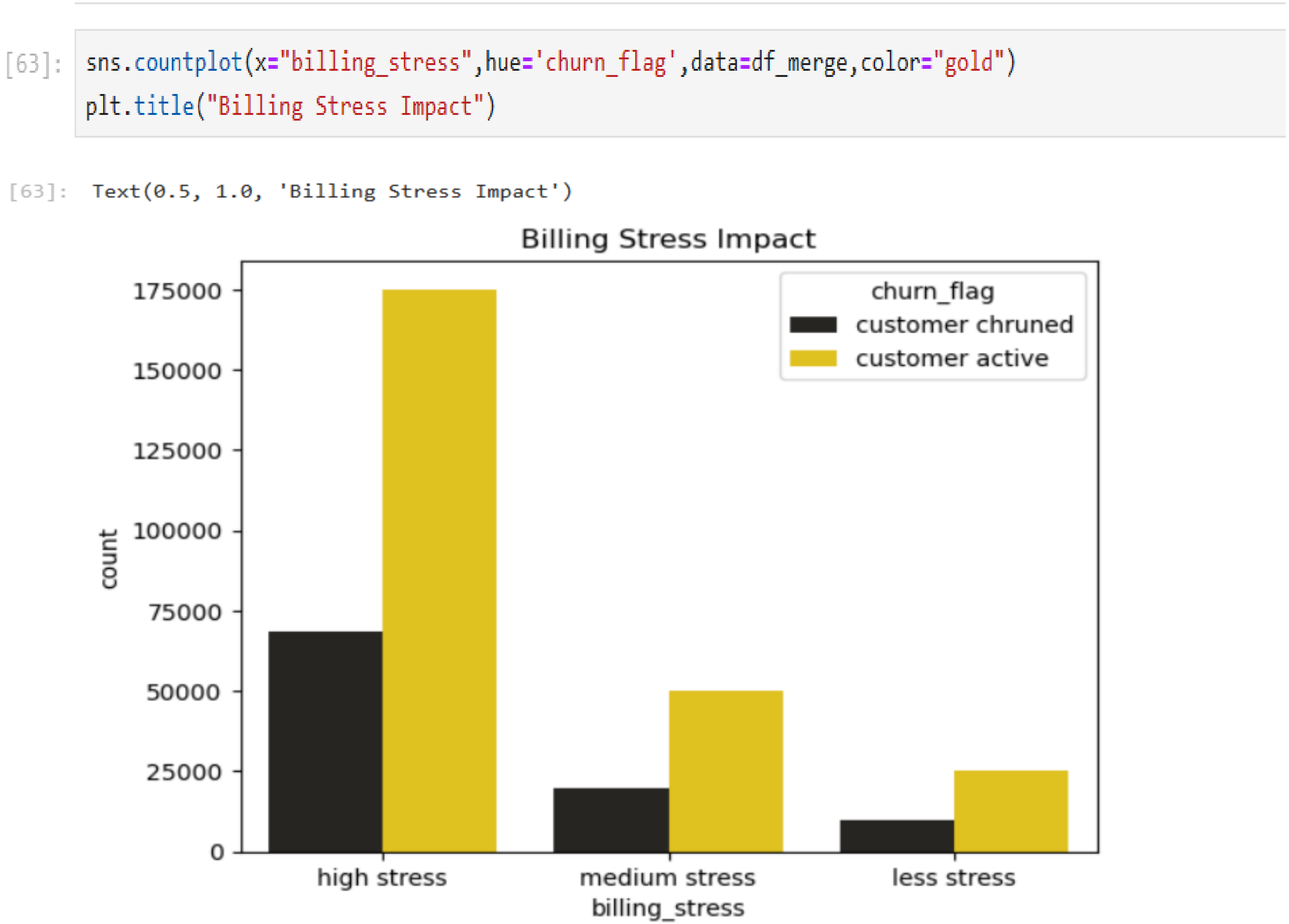
```
[51]: sns.countplot(x="engagement_level",data=df_merge,color='red')
      plt.title("Engagement Level Distribution")
```

```
[51]: Text(0.5, 1.0, 'Engagement Level Distribution')
```



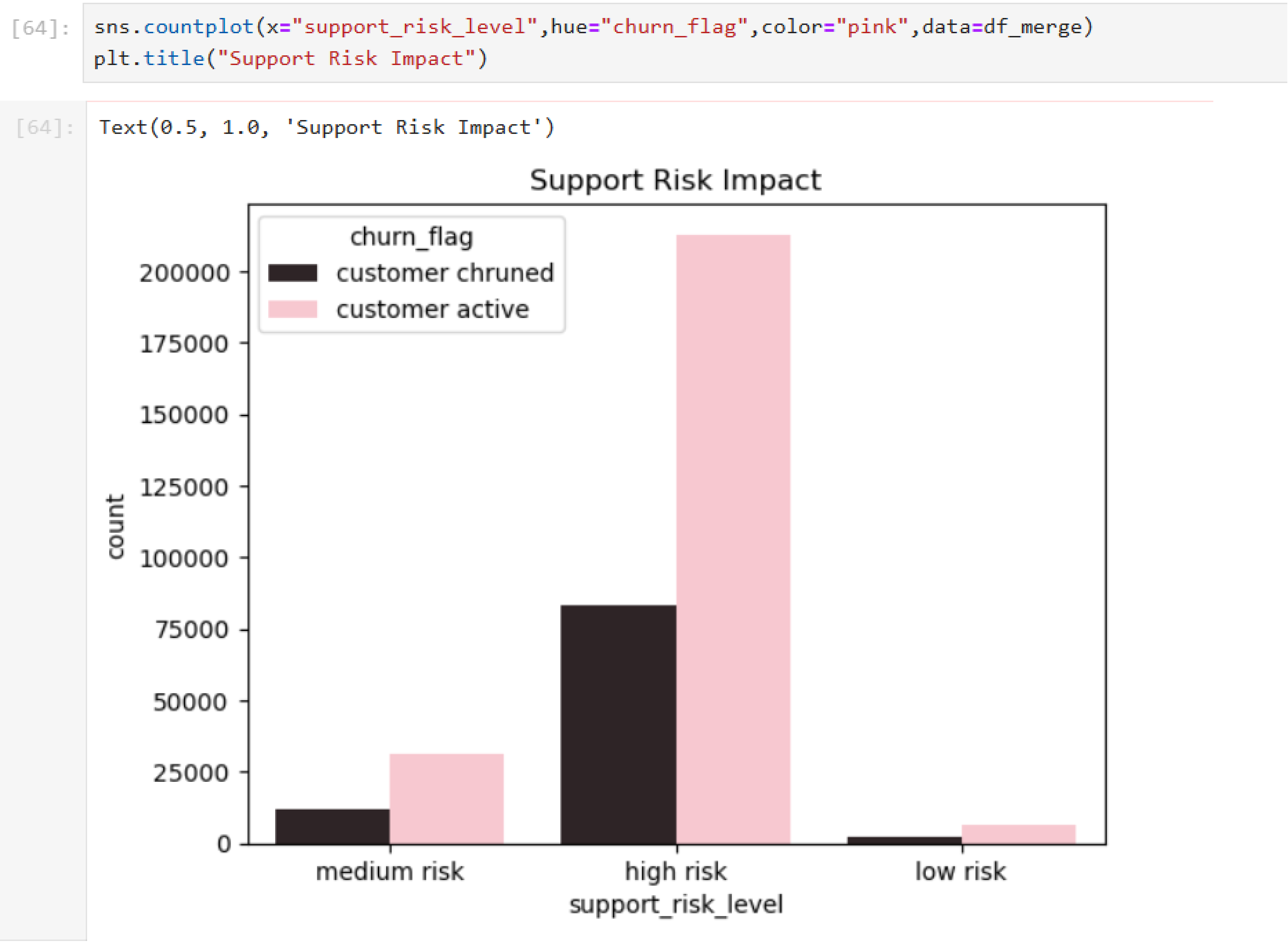
```
sns.countplot(x="usage_level",data=df_merge,color='yellow')
plt.title("Data Usage Level Distribution")
```

```
Text(0.5, 1.0, 'Data Usage Level Distribution')
```



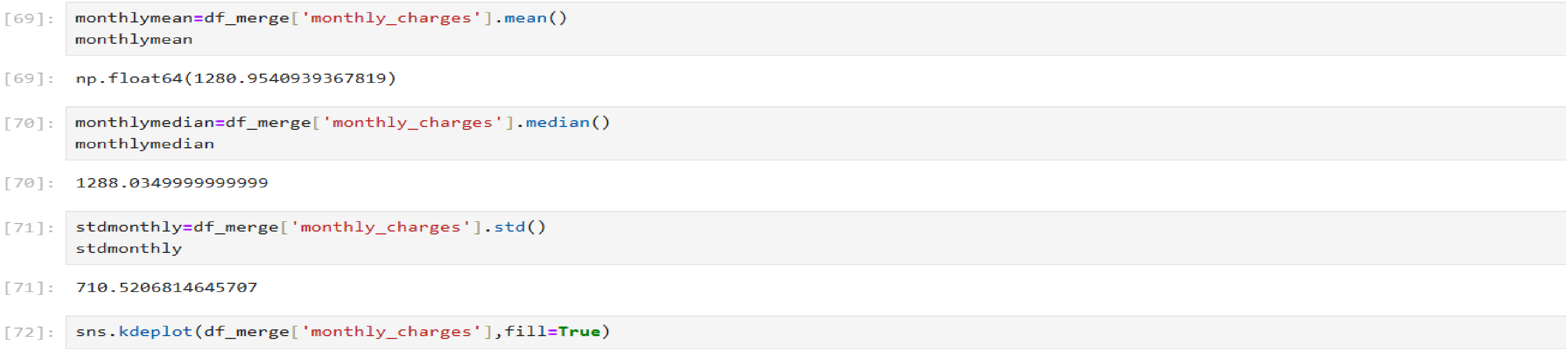
```
[63]: sns.countplot(x="billing_stress",hue='churn_flag',data=df_merge,color="gold")
      plt.title("Billing Stress Impact")
```

```
[63]: Text(0.5, 1.0, 'Billing Stress Impact')
```

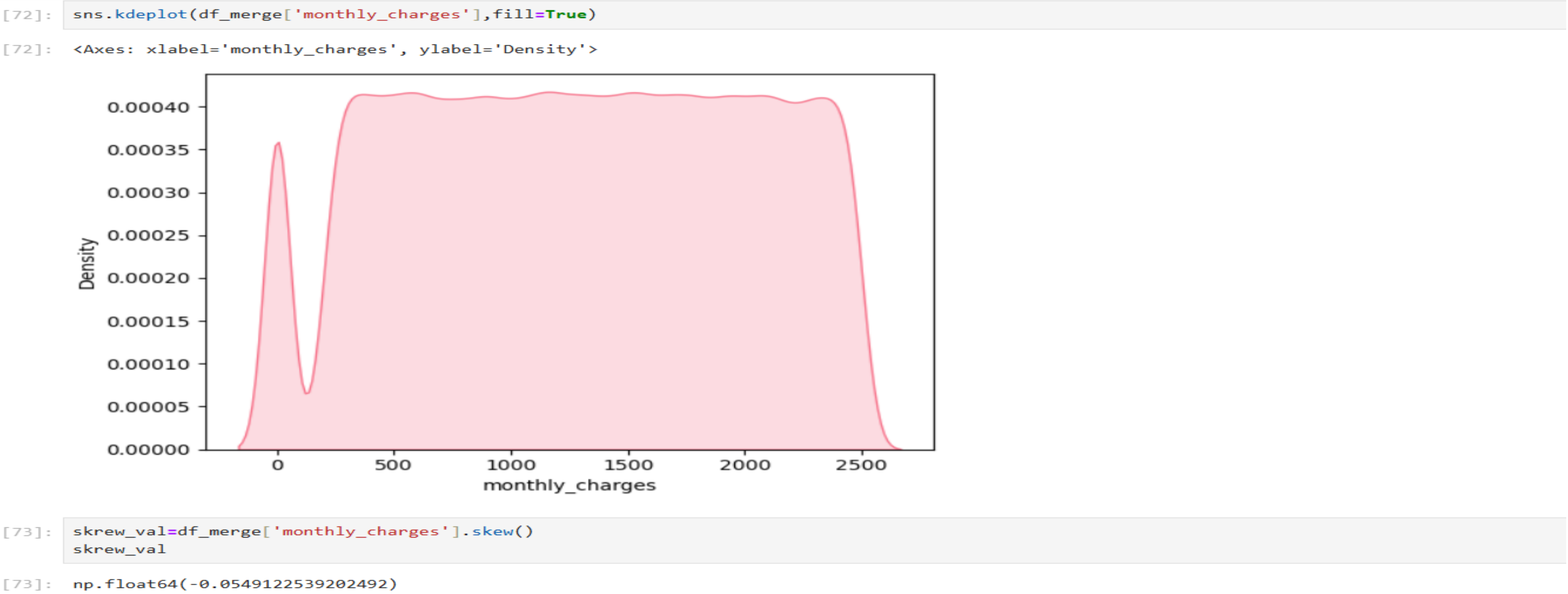


```
[64]: sns.countplot(x="support_risk_level",hue="churn_flag",color="pink",data=df_merge)
      plt.title("Support Risk Impact")
```

```
[64]: Text(0.5, 1.0, 'Support Risk Impact')
```

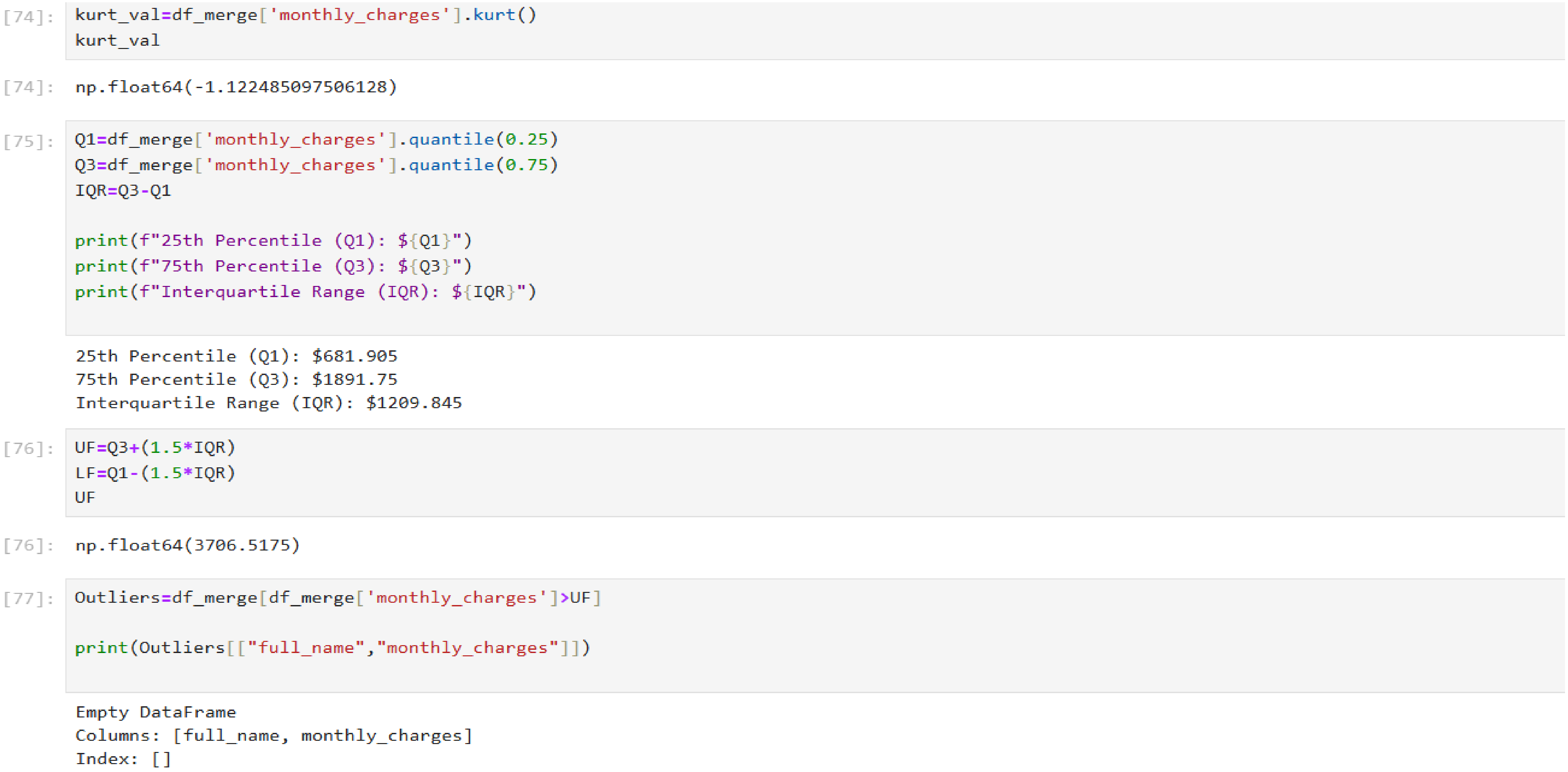


```
[69]: monthlymean=df_merge['monthly_charges'].mean()
      monthlymean

[69]: np.float64(1280.9540939367819)

[70]: monthlymedian=df_merge['monthly_charges'].median()
      monthlymedian

[70]: 1288.0349999999999

[71]: stdmonthly=df_merge['monthly_charges'].std()
      stdmonthly

[71]: 710.5206814645707

[72]: sns.kdeplot(df_merge['monthly_charges'],fill=True)
```



```
[72]: sns.kdeplot(df_merge['monthly_charges'],fill=True)

[72]: <Axes: xlabel='monthly_charges', ylabel='Density'>
```

```
[73]: skrew_val=df_merge['monthly_charges'].skew()
      skrew_val

[73]: np.float64(-0.0549122539202492)
```



```
[74]: kurt_val=df_merge['monthly_charges'].kurt()
      kurt_val
```

```
[74]: np.float64(-1.122485097506128)
```

```
[75]: Q1=df_merge['monthly_charges'].quantile(0.25)
      Q3=df_merge['monthly_charges'].quantile(0.75)
      IQR=Q3-Q1

      print(f"25th Percentile (Q1): ${Q1}")
      print(f"75th Percentile (Q3): ${Q3}")
      print(f"Interquartile Range (IQR): ${IQR}")
```

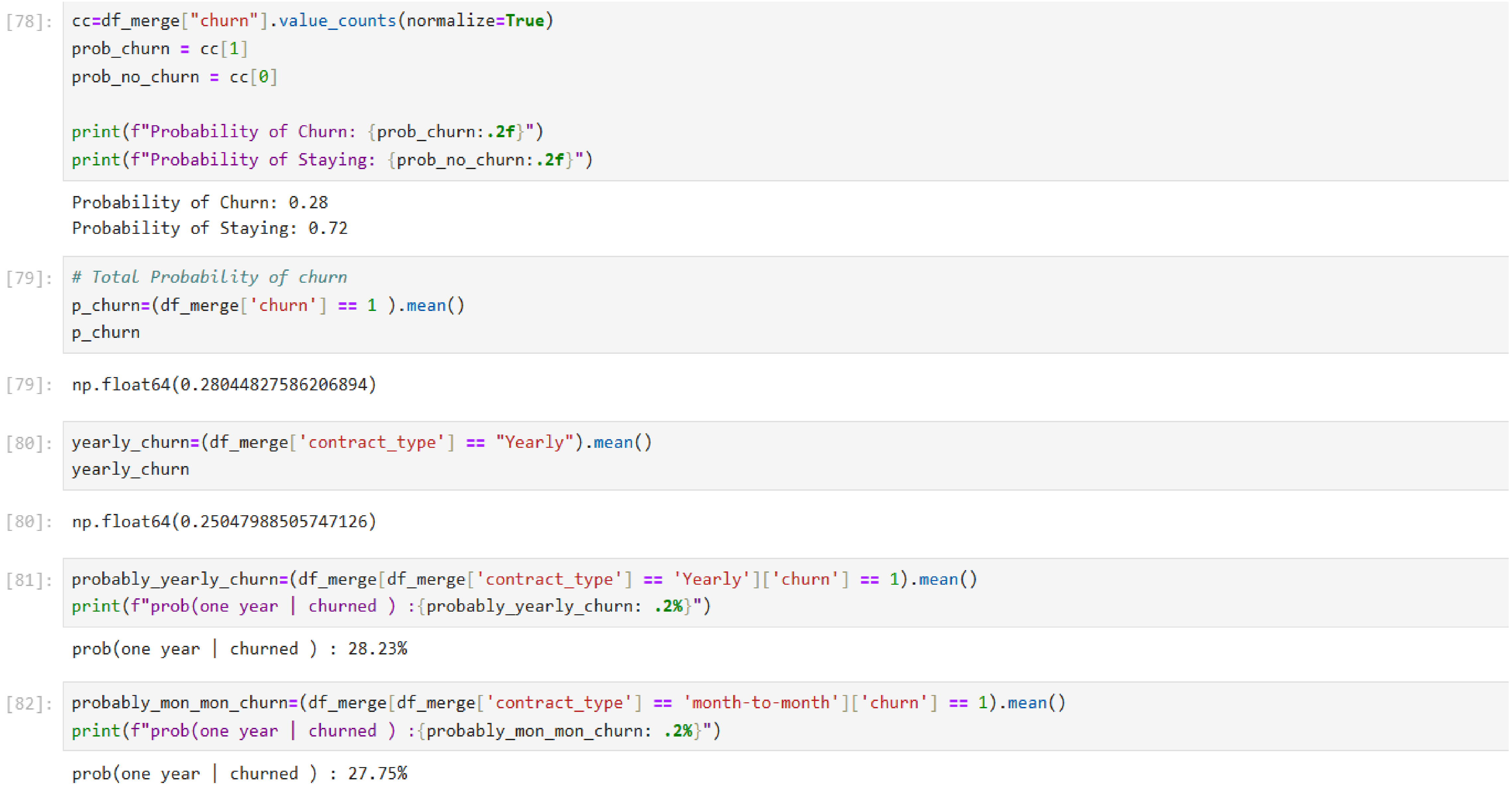
```
25th Percentile (Q1): $681.905
75th Percentile (Q3): $1891.75
Interquartile Range (IQR): $1209.845
```

```
[76]: UF=Q3+(1.5*IQR)
      LF=Q1-(1.5*IQR)
      UF
```

```
[76]: np.float64(3706.5175)
```

```
[77]: Outliers=df_merge[df_merge['monthly_charges']>UF]

      print(Outliers[["full_name","monthly_charges"]])
```

```
Empty DataFrame
Columns: [full_name, monthly_charges]
Index: []
```



```
[78]: cc=df_merge["churn"].value_counts(normalize=True)
      prob_churn = cc[1]
      prob_no_churn = cc[0]

      print(f"Probability of Churn: {prob_churn:.2f}")
      print(f"Probability of Staying: {prob_no_churn:.2f}")
```

```
      Probability of Churn: 0.28
      Probability of Staying: 0.72
```

```
[79]: # Total Probability of churn
      p_churn=(df_merge['churn'] == 1 ).mean()
      p_churn
```

```
[79]: np.float64(0.28044827586206894)
```

```
[80]: yearly_churn=(df_merge['contract_type'] == "Yearly").mean()
      yearly_churn
```

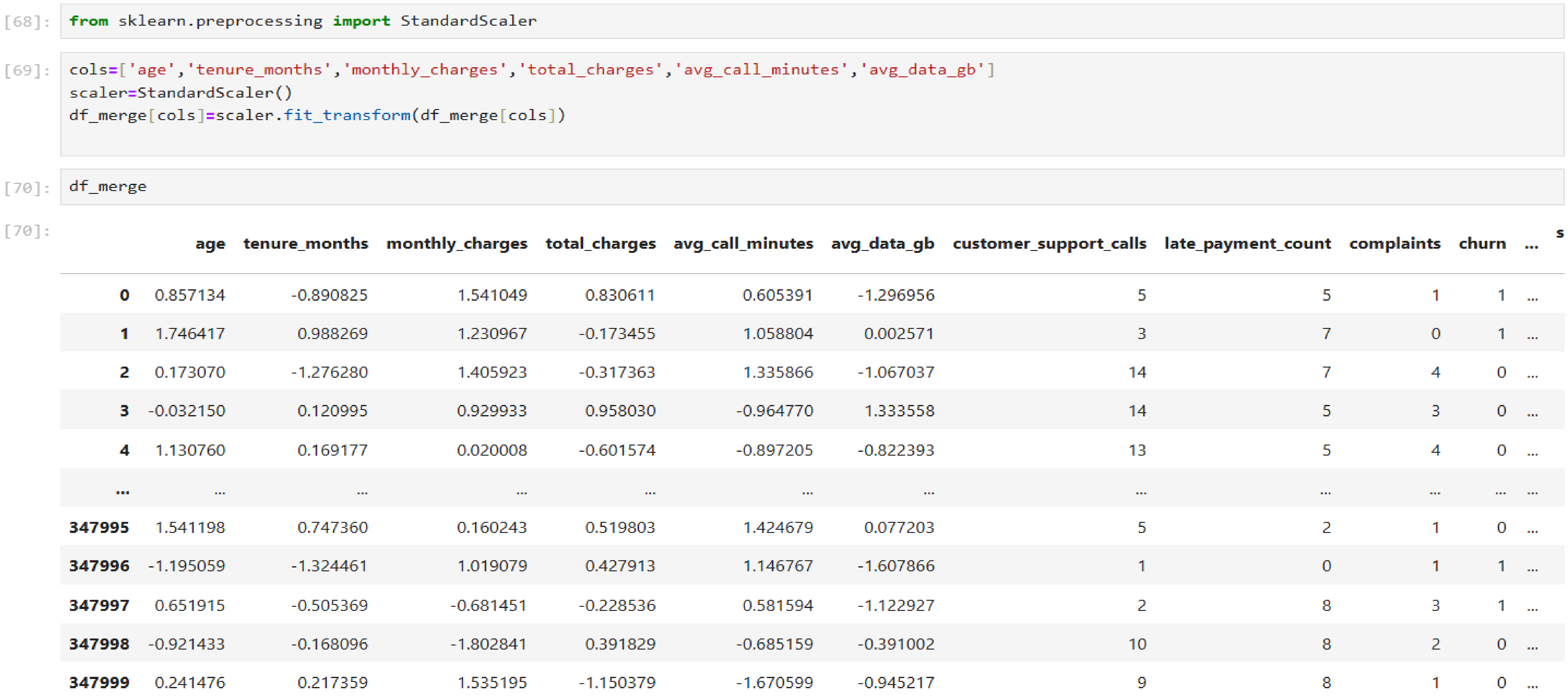
```
[80]: np.float64(0.25047988505747126)
```

```
[81]: probably_yearly_churn=(df_merge[df_merge['contract_type'] == 'Yearly']['churn'] == 1).mean()
      print(f"prob(one year | churned ) :{probably_yearly_churn: .2%}")
```

```
      prob(one year | churned ) : 28.23%
```

```
[82]: probably_mon_mon_churn=(df_merge[df_merge['contract_type'] == 'month-to-month']['churn'] == 1).mean()
      print(f"prob(one year | churned ) :{probably_mon_mon_churn: .2%}")
```

```
      prob(one year | churned ) : 27.75%
```



```
[68]: from sklearn.preprocessing import StandardScaler
```

```
[69]: cols=['age','tenure_months','monthly_charges','total_charges','avg_call_minutes','avg_data_gb']
      scaler=StandardScaler()
      df_merge[cols]=scaler.fit_transform(df_merge[cols])
```

```
[70]: df_merge
```

[70]:

	age	tenure_months	monthly_charges	total_charges	avg_call_minutes	avg_data_gb	customer_support_calls	late_payment_count	complaints	churn	...	s
0	0.857134	-0.890825	1.541049	0.830611	0.605391	-1.296956	5	5	1	1	...	
1	1.746417	0.988269	1.230967	-0.173455	1.058804	0.002571	3	7	0	1	...	
2	0.173070	-1.276280	1.405923	-0.317363	1.335866	-1.067037	14	7	4	0	...	
3	-0.032150	0.120995	0.929933	0.958030	-0.964770	1.333558	14	5	3	0	...	
4	1.130760	0.169177	0.020008	-0.601574	-0.897205	-0.822393	13	5	4	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
347995	1.541198	0.747360	0.160243	0.519803	1.424679	0.077203	5	2	1	0	...	
347996	-1.195059	-1.324461	1.019079	0.427913	1.146767	-1.607866	1	0	1	1	...	
347997	0.651915	-0.505369	-0.681451	-0.228536	0.581594	-1.122927	2	8	3	1	...	
347998	-0.921433	-0.168096	-1.802841	0.391829	-0.685159	-0.391002	10	8	2	0	...	
347999	0.241476	0.217359	1.535195	-1.150379	-1.670599	-0.945217	9	8	1	0	...	

[73]:

```
X=df_merge.drop(columns=['churn'])
y=df_merge['churn']
```

[76]:

```
from sklearn.model_selection import  train_test_split
```

[80]:

```

X_train,X_test,y_train,y_test= train_test_split(X,y, test_size=0.2,random_state=42,stratify=y)
```

[81]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)
```

```
C:\Users\Hp\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:465: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

[81]:

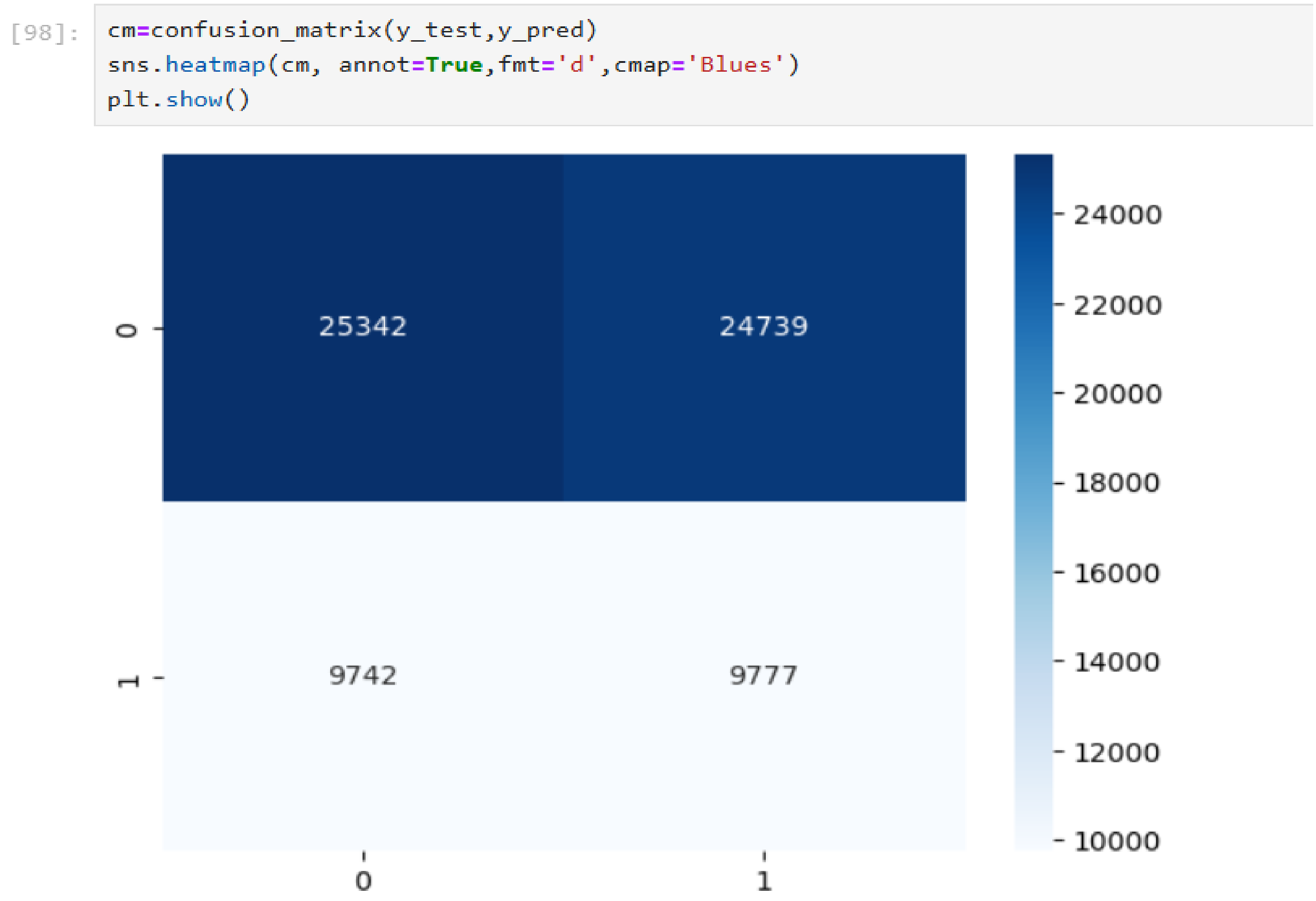
```
▾ LogisticRegression
LogisticRegression()
```

[82]:

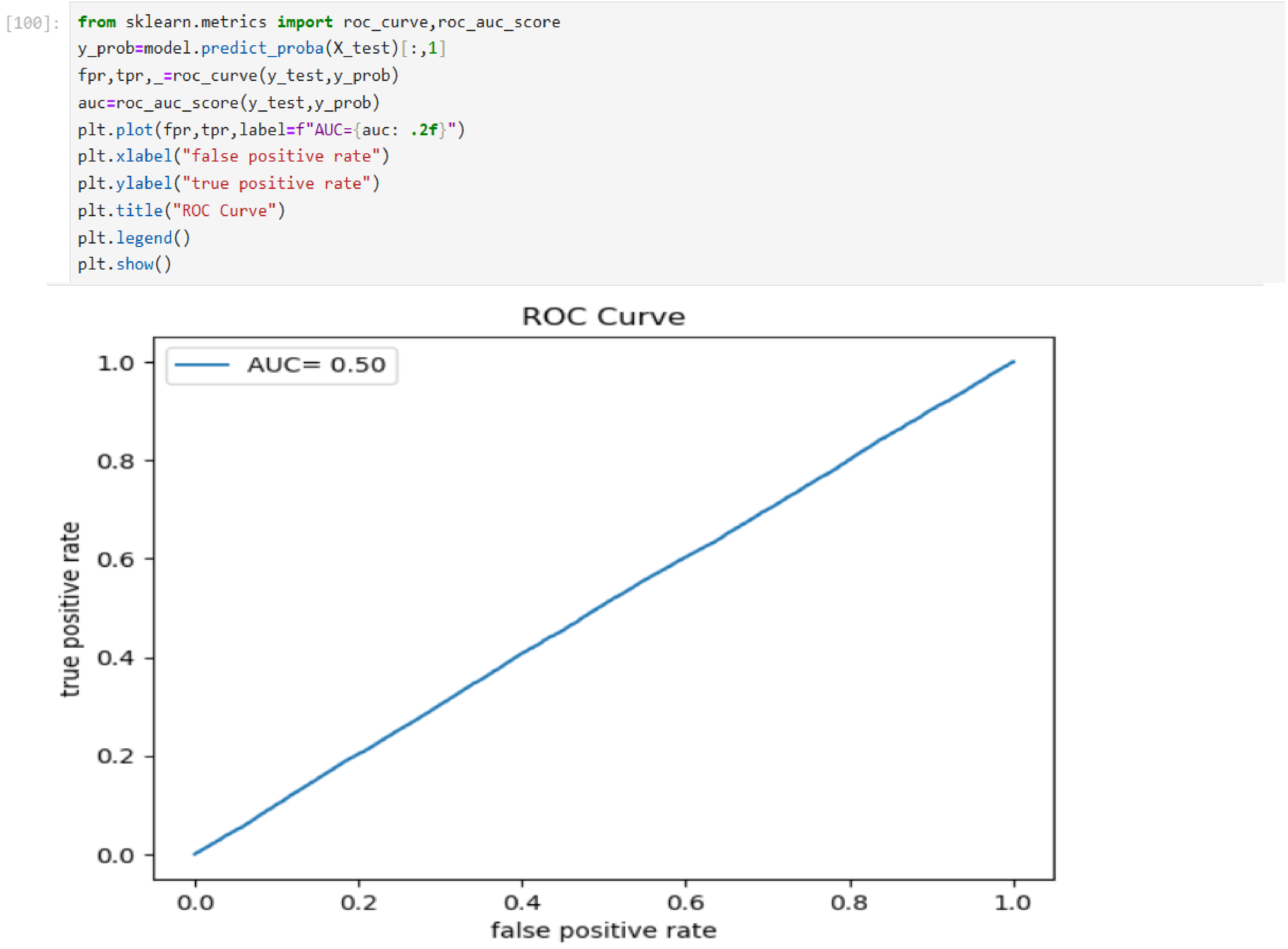
```
y_pred=model.predict(X_test)
```

[83]:

```
y_pred
```

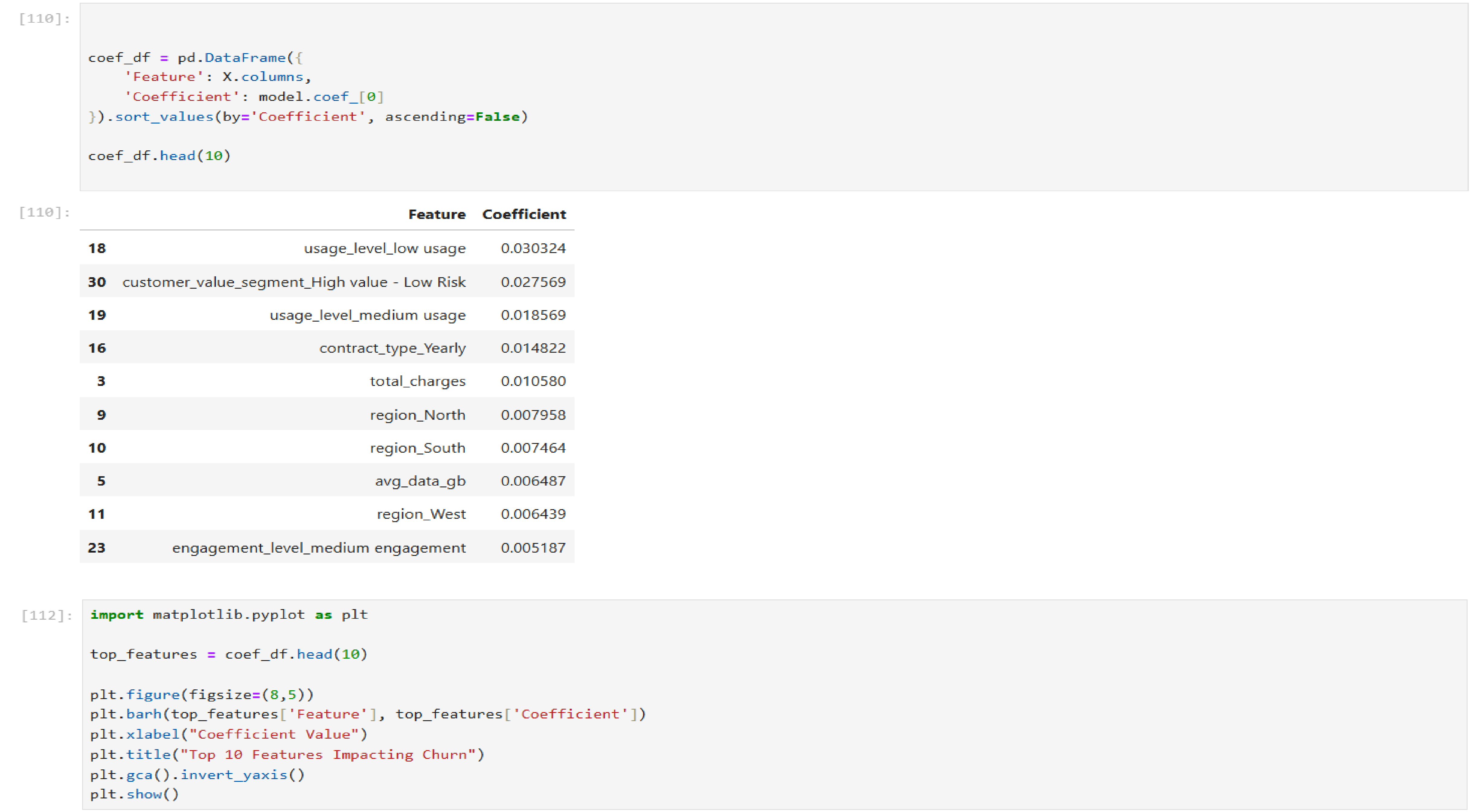


```
[98]: cm=confusion_matrix(y_test,y_pred)
      sns.heatmap(cm, annot=True,fmt='d',cmap='Blues')
      plt.show()
```



[100]:

```
from sklearn.metrics import roc_curve,roc_auc_score
y_prob=model.predict_proba(X_test)[:,1]
fpr,tpr,_=roc_curve(y_test,y_prob)
auc=roc_auc_score(y_test,y_prob)
plt.plot(fpr,tpr,label=f"AUC={auc: .2f}")
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```



[110]:

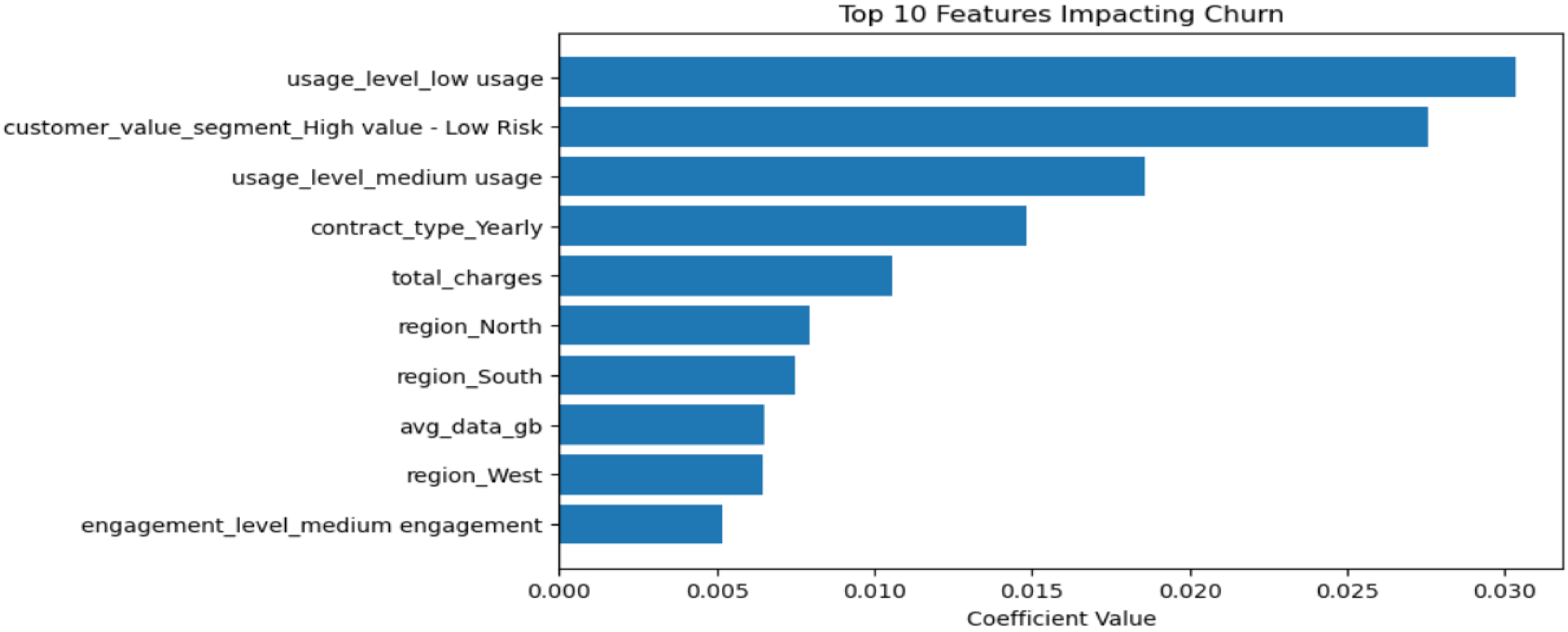
```python
coef_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.coef_[0]
}).sort_values(by='Coefficient', ascending=False)

coef_df.head(10)
```

[110]:

	Feature	Coefficient
18	usage_level_low usage	0.030324
30	customer_value_segment_High value - Low Risk	0.027569
19	usage_level_medium usage	0.018569
16	contract_type_Yearly	0.014822
3	total_charges	0.010580
9	region_North	0.007958
10	region_South	0.007464
5	avg_data_gb	0.006487
11	region_West	0.006439
23	engagement_level_medium engagement	0.005187

[112]:

```python
import matplotlib.pyplot as plt

top_features = coef_df.head(10)

plt.figure(figsize=(8,5))
plt.barh(top_features['Feature'], top_features['Coefficient'])
plt.xlabel("Coefficient Value")
plt.title("Top 10 Features Impacting Churn")
plt.gca().invert_yaxis()
plt.show()
```



Top 10 Features Impacting Churn
usage_level_low usage
customer_value_segment_High value - Low Risk
usage_level_medium usage
contract_type_Yearly
total_charges
region_North
region_South
avg_data_gb
region_West
engagement_level_medium engagement
0.000
0.005
0.010
0.015
0.020
0.025
0.030
Coefficient Value