customer-churn-prediction-1

August 24, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, confusion_matrix, precision_score,␣
↪f1_score, accuracy_score, classification_report
```

[3]:
```
#loading data
df=pd.read_excel("C:\\Users\\pavan\\Downloads\\customer.xlsx")
```

[4]:
```
df
```

[4]:
```
   CustomerID          Name  Age  Gender     Location  \
0           1    Customer_1   63    Male  Los Angeles
```

```
1              2        Customer_2   62  Female     New York
2              3        Customer_3   24  Female  Los Angeles
3              4        Customer_4   36  Female        Miami
4              5        Customer_5   46  Female        Miami
…            …           …   …       …          …
99995      99996    Customer_99996   33    Male      Houston
99996      99997    Customer_99997   62  Female     New York
99997      99998    Customer_99998   64    Male      Chicago
99998      99999    Customer_99999   51  Female     New York
99999     100000   Customer_100000   27  Female  Los Angeles

       Subscription_Length_Months  Monthly_Bill  Total_Usage_GB  Churn
0                              17         73.36             236      0
1                               1         48.76             172      0
2                               5         85.47             460      0
3                               3         97.94             297      1
4                              19         58.14             266      0
…                             …           …             …    …
99995                          23         55.13             226      1
99996                          19         61.65             351      0
99997                          17         96.11             251      1
99998                          20         49.25             434      1
99999                          19         76.57             173      1

[100000 rows x 9 columns]
```

[5]:
```python
df.head()
```

[5]:
```
   CustomerID        Name  Age  Gender     Location  \
0           1  Customer_1   63    Male  Los Angeles
1           2  Customer_2   62  Female     New York
2           3  Customer_3   24  Female  Los Angeles
3           4  Customer_4   36  Female        Miami
4           5  Customer_5   46  Female        Miami

   Subscription_Length_Months  Monthly_Bill  Total_Usage_GB  Churn
0                          17         73.36             236      0
1                           1         48.76             172      0
2                           5         85.47             460      0
3                           3         97.94             297      1
4                          19         58.14             266      0
```

[6]:
```python
df.shape
```

[6]:
```
(100000, 9)
```

[7]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   CustomerID                  100000 non-null  int64  
 1   Name                        100000 non-null  object 
 2   Age                         100000 non-null  int64  
 3   Gender                      100000 non-null  object 
 4   Location                    100000 non-null  object 
 5   Subscription_Length_Months  100000 non-null  int64  
 6   Monthly_Bill                100000 non-null  float64
 7   Total_Usage_GB              100000 non-null  int64  
 8   Churn                       100000 non-null  int64  
dtypes: float64(1), int64(5), object(3)
memory usage: 6.9+ MB
```

[8]:
```
df.columns.values
```

[8]:
```
array(['CustomerID', 'Name', 'Age', 'Gender', 'Location',
       'Subscription_Length_Months', 'Monthly_Bill', 'Total_Usage_GB',
       'Churn'], dtype=object)
```

[9]:
```
df.dtypes
```

[9]:
```
CustomerID                      int64
Name                           object
Age                             int64
Gender                         object
Location                       object
Subscription_Length_Months      int64
Monthly_Bill                  float64
Total_Usage_GB                  int64
Churn                           int64
dtype: object
```

[10]:
```
df = df.drop(['CustomerID'], axis = 1)
df.head()
```

[10]:
```
         Name  Age  Gender     Location  Subscription_Length_Months  \
0  Customer_1   63    Male  Los Angeles                          17   
1  Customer_2   62  Female     New York                           1   
2  Customer_3   24  Female  Los Angeles                           5   
3  Customer_4   36  Female        Miami                           3   
4  Customer_5   46  Female        Miami                          19   

   Monthly_Bill  Total_Usage_GB  Churn  
```

```
0          73.36              236       0
1          48.76              172       0
2          85.47              460       0
3          97.94              297       1
4          58.14              266       0
```

[11]:
```
df['Total_Usage_GB'] = pd.to_numeric(df.Total_Usage_GB, errors='coerce')
df.isnull().sum()
```

[11]:
```
Name                          0
Age                           0
Gender                        0
Location                      0
Subscription_Length_Months    0
Monthly_Bill                  0
Total_Usage_GB                0
Churn                         0
dtype: int64
```

[12]:
```
df[np.isnan(df['Total_Usage_GB'])]
```

[12]:
```
Empty DataFrame
Columns: [Name, Age, Gender, Location, Subscription_Length_Months, Monthly_Bill,
Total_Usage_GB, Churn]
Index: []
```

[13]:
```
df[df['Age'] == 0].index
```

[13]:
```
Int64Index([], dtype='int64')
```

[14]:
```
df.drop(labels=df[df['Age'] == 0].index, axis=0, inplace=True)
df[df['Age'] == 0].index
```

[14]:
```
Int64Index([], dtype='int64')
```

[15]:
```
df.fillna(df["Total_Usage_GB"].mean())
```

[15]:
```
                 Name  Age  Gender     Location  Subscription_Length_Months  \
0          Customer_1   63    Male  Los Angeles                          17
1          Customer_2   62  Female     New York                           1
2          Customer_3   24  Female  Los Angeles                           5
3          Customer_4   36  Female        Miami                           3
4          Customer_5   46  Female        Miami                          19
…                 …  …    …             …                         …
99995  Customer_99996   33    Male      Houston                          23
99996  Customer_99997   62  Female     New York                          19
99997  Customer_99998   64    Male      Chicago                          17
```

```
99998   Customer_99999   51  Female     New York                        20
99999  Customer_100000   27  Female  Los Angeles                        19

       Monthly_Bill  Total_Usage_GB  Churn
0             73.36             236      0
1             48.76             172      0
2             85.47             460      0
3             97.94             297      1
4             58.14             266      0
…                 …           …   …
99995         55.13             226      1
99996         61.65             351      0
99997         96.11             251      1
99998         49.25             434      1
99999         76.57             173      1

[100000 rows x 8 columns]
```

[16]:
```
df.isnull().sum()
```

[16]:
```
Name                          0
Age                           0
Gender                        0
Location                      0
Subscription_Length_Months    0
Monthly_Bill                  0
Total_Usage_GB                0
Churn                         0
dtype: int64
```

[17]:
```
df["Monthly_Bill"].describe(include=['object', 'bool'])
```

[17]:
```
count    100000.000000
mean         65.053197
std          20.230696
min          30.000000
25%          47.540000
50%          65.010000
75%          82.640000
max         100.000000
Name: Monthly_Bill, dtype: float64
```

[18]:
```
numerical_cols = ['Age', 'Monthly_Bill', 'Total_Usage_GB']
df[numerical_cols].describe()
```

[18]:
```
                 Age   Monthly_Bill  Total_Usage_GB
count  100000.000000  100000.000000   100000.000000
```

```
mean       44.027020      65.053197      274.393650
std        15.280283      20.230696      130.463063
min        18.000000      30.000000       50.000000
25%        31.000000      47.540000      161.000000
50%        44.000000      65.010000      274.000000
75%        57.000000      82.640000      387.000000
max        70.000000     100.000000      500.000000
```

[19]:
```
g_labels = ['Male', 'Female']
c_labels = ['No', 'Yes']
# Create subplots: use 'domain' type for Pie subplot
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':
 ↪'domain'}]])
fig.add_trace(go.Pie(labels=g_labels, values=df['Gender'].value_counts(),␣
 ↪name="Gender"),
              1, 1)
fig.add_trace(go.Pie(labels=c_labels, values=df['Age'].value_counts(),␣
 ↪name="Age"),
              1, 2)

# Use `hole` to create a donut-like pie chart
fig.update_traces(hole=.4, hoverinfo="label+percent+name", textfont_size=16)

fig.update_layout(
    title_text="customer_churn_prediction",
    # Add annotations in the center of the donut pies.
    annotations=[dict(text='Gender', x=0.16, y=0.5, font_size=20,␣
 ↪showarrow=False),
                 dict(text='Age', x=0.84, y=0.5, font_size=20,␣
 ↪showarrow=False)])
fig.show()
```

[20]:
```
df["Gender"][df["Gender"]=="No"].groupby(by=df["Age"]).count()
```

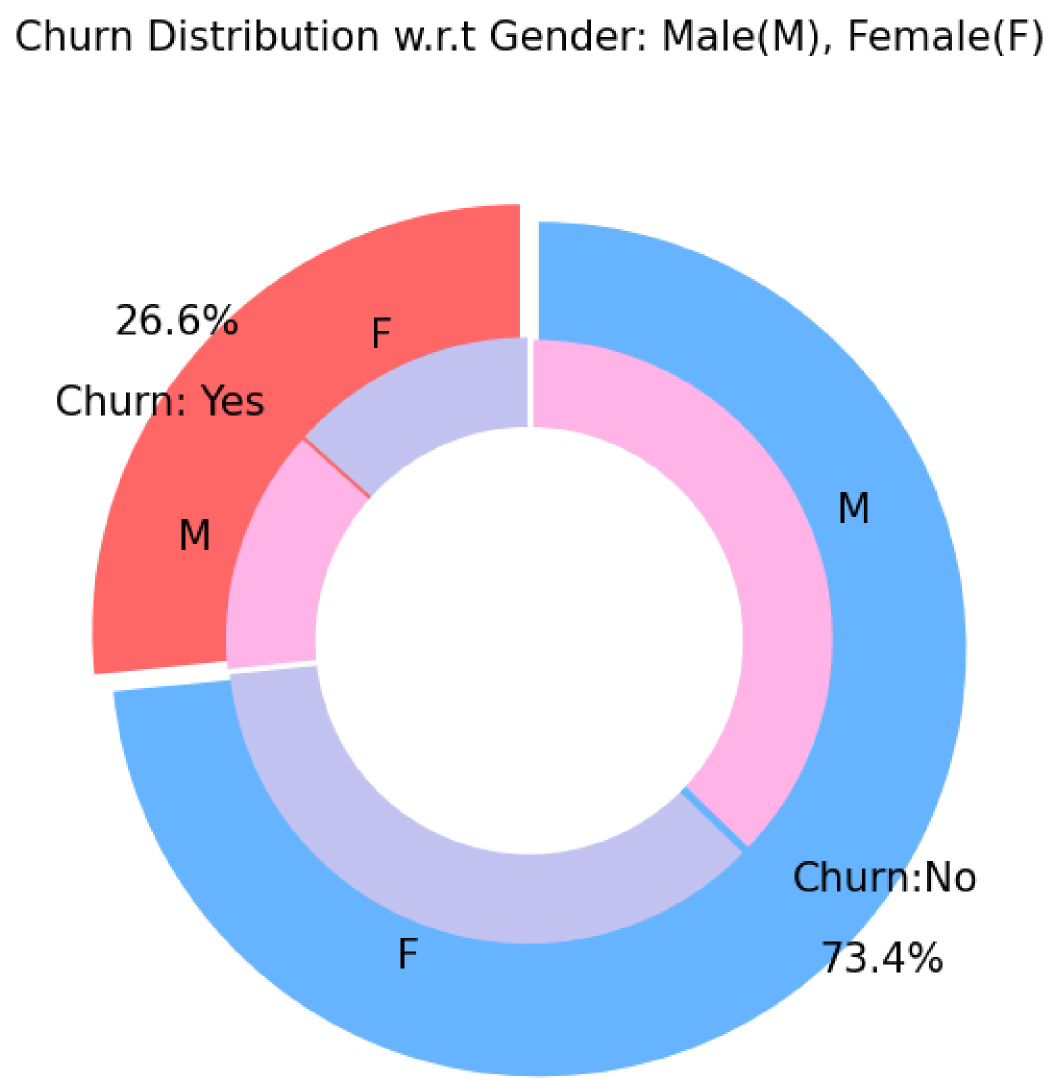
[20]:
```
Series([], Name: Gender, dtype: int64)
```

[21]:
```
df["Age"][df["Age"]=="Yes"].groupby(by=df["Gender"]).count()
```

[21]:
```
Series([], Name: Age, dtype: int64)
```

[22]:
```
plt.figure(figsize=(6, 6))
labels =["Churn: Yes","Churn:No"]
values = [1869,5163]
labels_gender = ["F","M","F","M"]
sizes_gender = [939,930 , 2544,2619]
colors = ['#ff6666', '#66b3ff']
colors_gender = ['#c2c2f0','#ffb3e6', '#c2c2f0','#ffb3e6']
```

```
explode = (0.3,0.3)
explode_gender = (0.1,0.1,0.1,0.1)
textprops = {"fontsize":15}
#Plot
plt.pie(values, labels=labels,autopct='%1.1f%%',pctdistance=1.08,␣
 ↪labeldistance=0.8,colors=colors, startangle=90,frame=True,␣
 ↪explode=explode,radius=10, textprops =textprops, counterclock = True, )
plt.pie(sizes_gender,labels=labels_gender,colors=colors_gender,startangle=90,␣
 ↪explode=explode_gender,radius=7, textprops =textprops, counterclock = True, )
#Draw circle
centre_circle = plt.Circle((0,0),5,color='black', fc='white',linewidth=0)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title('Churn Distribution w.r.t Gender: Male(M), Female(F)', fontsize=15,␣
 ↪y=1.1)

# show plot

plt.axis('equal')
plt.tight_layout()
plt.show()
```



[24]:
```
labels = df['Monthly_Bill'].unique()
values = df['Monthly_Bill'].value_counts()

fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.update_layout(title_text="<b>Payment Method Distribution</b>")
fig.show()
```

[25]:
```
fig = px.histogram(df, x="Churn", color="Monthly_Bill", title="<b>Customer␣
↪Payment Method distribution w.r.t. Churn</b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

[26]:
```
df["Subscription_Length_Months"].unique()
```

[26]:
```
array([17,  1,  5,  3, 19, 15, 10, 12, 20, 13,  8, 23,  2,  4, 18,  9, 14,
       16,  6,  7, 24, 22, 11, 21], dtype=int64)
```

[27]:
```
df[df["Gender"]=="Male"][["Subscription_Length_Months", "Churn"]].value_counts()
```

[27]:
```
Subscription_Length_Months  Churn
1                           1        1109
2                           1        1089
14                          0        1088
21                          0        1088
20                          0        1087
22                          0        1083
15                          0        1079
22                          1        1067
10                          1        1067
2                           0        1059
20                          1        1058
13                          1        1058
5                           0        1056
14                          1        1056
11                          1        1056
                            0        1050
16                          1        1049
18                          1        1046
7                           1        1044
9                           0        1042
10                          0        1040
12                          1        1040
16                          0        1038
3                           1        1037
24                          0        1037
17                          0        1034
7                           0        1034
4                           0        1030
8                           1        1027
19                          1        1025
9                           1        1024
5                           1        1024
24                          1        1024
8                           0        1022
6                           0        1020
18                          0        1018
23                          0        1016
19                          0        1012
1                           0        1011
```

```
3                            0         1010
23                           1         1006
12                           0         1001
6                            1         1000
4                            1         1000
13                           0          994
15                           1          980
21                           1          977
17                           1          972
dtype: int64
```

[28]:
```python
df[df["Gender"]=="Female"][["Subscription_Length_Months", "Churn"]].
 ↪value_counts()
```

[28]:
```
Subscription_Length_Months  Churn
6                            0         1124
5                            0         1099
1                            0         1098
20                           0         1081
18                           0         1081
16                           1         1080
20                           1         1077
22                           1         1072
7                            1         1071
12                           0         1069
16                           0         1062
7                            0         1062
21                           1         1061
2                            0         1060
24                           1         1059
11                           0         1058
9                            0         1055
13                           0         1053
14                           0         1052
4                            0         1050
3                            1         1049
17                           1         1049
13                           1         1049
19                           1         1047
12                           1         1045
22                           0         1045
8                            0         1044
23                           1         1042
6                            1         1040
3                            0         1040
11                           1         1036
10                           0         1034
```

```
15                            0         1034
                              1         1029
1                             1         1029
21                            0         1028
18                            1         1026
19                            0         1022
2                             1         1020
23                            0         1019
4                             1         1018
14                            1         1017
8                             1         1013
9                             1         1013
10                            1         1010
17                            0         1009
24                            0          993
5                             1          992
dtype: int64
```

[29]:

```
fig = go.Figure()

fig.add_trace(go.Bar(
  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
       ["Female", "Male", "Female", "Male"]],
  y = [965, 992, 219, 240],
  name = 'DSL',
))

fig.add_trace(go.Bar(
  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
       ["Female", "Male", "Female", "Male"]],
  y = [889, 910, 664, 633],
  name = 'Fiber optic',
))

fig.add_trace(go.Bar(
  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
       ["Female", "Male", "Female", "Male"]],
  y = [690, 717, 56, 57],
  name = 'No Internet',
))

fig.update_layout(title_text="<b>Churn Distribution w.r.t. Internet Service and␣
↪Gender</b>")

fig.show()
```

[30]:
```
color_map = {"Yes": "#FF97FF", "No": "#AB63FA"}
fig = px.histogram(df, x="Churn", color="Subscription_Length_Months",␣
↪barmode="group", title="<b>Dependents distribution</b>",␣
↪color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

[31]:
```
color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
fig = px.histogram(df, x="Churn", color="Monthly_Bill", barmode="group",␣
↪title="<b>Chrun distribution w.r.t. Partners</b>",␣
↪color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

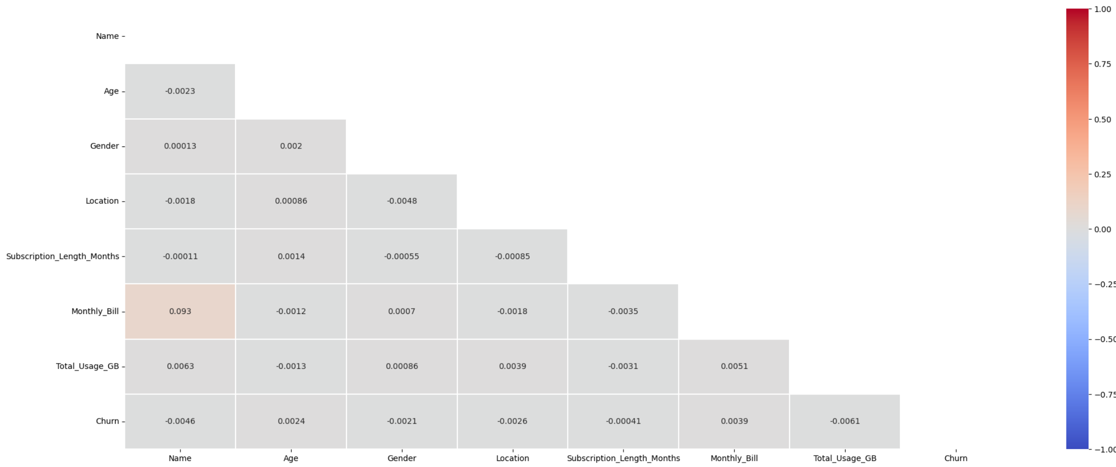
[32]:
```
color_map = {"Yes": '#00CC96', "No": '#B6E880'}
fig = px.histogram(df, x="Churn", color="Age", title="<b>Chrun distribution w.r.
↪t. Senior Citizen</b>", color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

[33]:
```
color_map = {"Yes": "#FF97FF", "No": "#AB63FA"}
fig = px.histogram(df, x="Churn", color="Age", barmode="group", title="<b>Churn␣
↪w.r.t Online Security</b>", color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

[34]:
```
color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
fig = px.histogram(df, x="Churn", color="Total_Usage_GB",  title="<b>Chrun␣
↪distribution w.r.t. Paperless Billing</b>", color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

[35]:
```
fig = px.histogram(df, x="Churn", color="Gender",barmode="group", ␣
↪title="<b>Chrun distribution w.r.t. TechSupport</b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

[36]:
```
plt.figure(figsize=(25, 10))

corr = df.apply(lambda x: pd.factorize(x)[0]).corr()

mask = np.triu(np.ones_like(corr, dtype=bool))

ax = sns.heatmap(corr, mask=mask, xticklabels=corr.columns, yticklabels=corr.
↪columns, annot=True, linewidths=.2, cmap='coolwarm', vmin=-1, vmax=1)
```



[37]:
```
def object_to_int(dataframe_series):
    if dataframe_series.dtype=='object':
        dataframe_series = LabelEncoder().fit_transform(dataframe_series)
    return dataframe_series
```

[38]:
```
df = df.apply(lambda x: object_to_int(x))
df.head()
```

[38]:
```
    Name  Age  Gender  Location  Subscription_Length_Months  Monthly_Bill  \
0      0   63       1         2                          17         73.36
1  11112   62       0         4                           1         48.76
2  22223   24       0         2                           5         85.47
3  33334   36       0         3                           3         97.94
4  44445   46       0         3                          19         58.14

   Total_Usage_GB  Churn
0             236      0
1             172      0
2             460      0
3             297      1
4             266      0
```

[39]:
```
plt.figure(figsize=(14,7))
df.corr()['Churn'].sort_values(ascending = False)
```

[39]:
```
Churn                         1.000000
Location                      0.006405
Subscription_Length_Months    0.002328
Gender                        0.002121
Age                           0.001559
Monthly_Bill                 -0.000211
```

```
Name                             -0.001418
Total_Usage_GB                   -0.002842
Name: Churn, dtype: float64
```

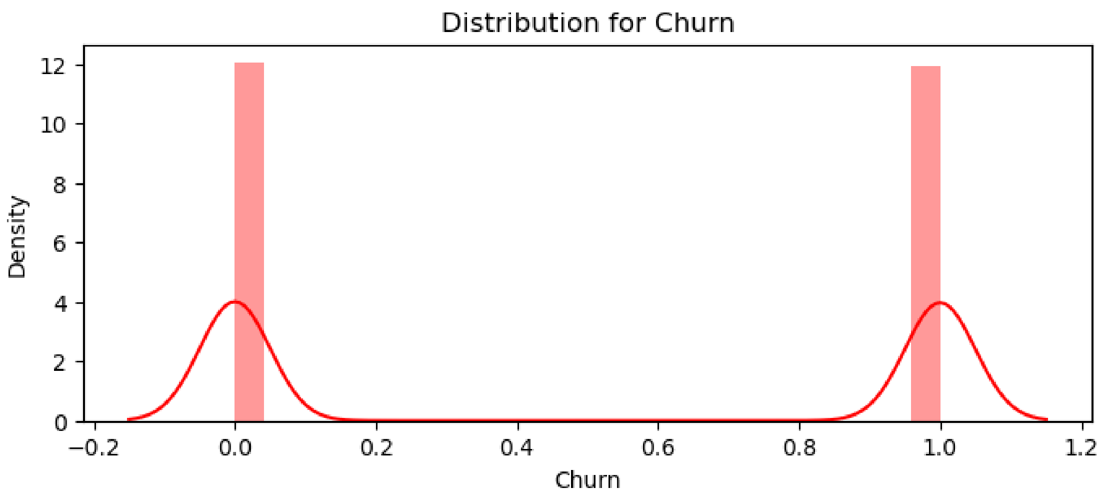
```
<Figure size 1400x700 with 0 Axes>
```

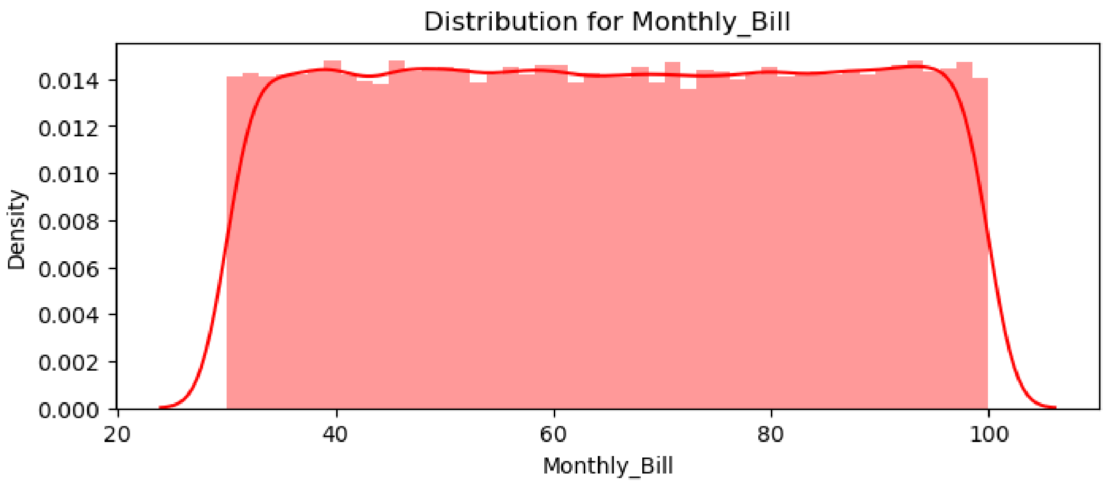
[40]:
```
X = df.drop(columns = ['Churn'])
y = df['Churn'].values
```

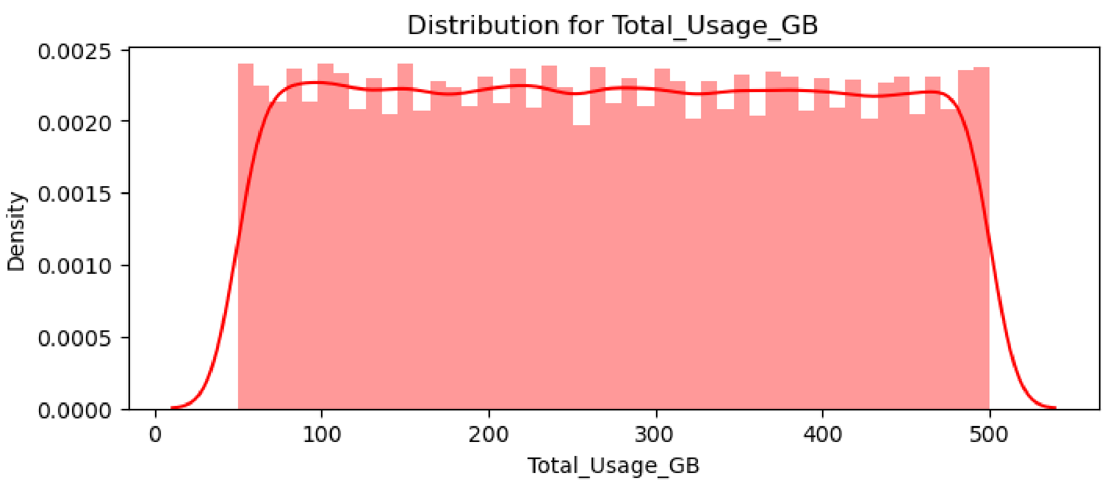
[41]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.30,␣
 ↪random_state = 40, stratify=y)
```

[42]:
```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(8,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color= color)
```

[43]:
```
num_cols = ["Churn", 'Monthly_Bill', 'Total_Usage_GB']
for feat in num_cols: distplot(feat, df)
```







[44]:
```
knn_model = KNeighborsClassifier(n_neighbors = 11)
knn_model.fit(X_train,y_train)
predicted_y = knn_model.predict(X_test)
accuracy_knn = knn_model.score(X_test,y_test)
print("KNN accuracy:",accuracy_knn)
```

```
KNN accuracy: 0.49773333333333336
```

[45]:
```
print(classification_report(y_test, predicted_y))
```

```
              precision    recall  f1-score   support

           0       0.50      0.50      0.50     15066
           1       0.50      0.49      0.49     14934
```

```
    accuracy                           0.50     30000
   macro avg       0.50      0.50      0.50     30000
weighted avg       0.50      0.50      0.50     30000
```

[46]:
```
svc_model = SVC(random_state = 1)
svc_model.fit(X_train,y_train)
predict_y = svc_model.predict(X_test)
accuracy_svc = svc_model.score(X_test,y_test)
print("SVM accuracy is :",accuracy_svc)
```

```
SVM accuracy is : 0.5022
```

[47]:
```
print(classification_report(y_test, predict_y))
```

```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67     15066
           1       0.00      0.00      0.00     14934

    accuracy                           0.50     30000
   macro avg       0.25      0.50      0.33     30000
weighted avg       0.25      0.50      0.34     30000
```

[48]:
```
model_rf = RandomForestClassifier(n_estimators=500 , oob_score = True, n_jobs =␣
 ↪-1,
                                  random_state =50, max_features = "auto",
                                  max_leaf_nodes = 30)
model_rf.fit(X_train, y_train)

# Make predictions
prediction_test = model_rf.predict(X_test)
print (metrics.accuracy_score(y_test, prediction_test))
```

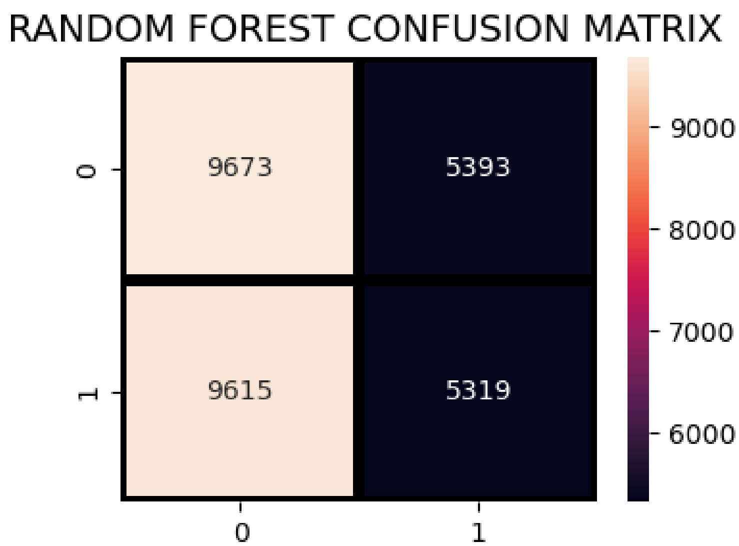
```
0.4997333333333333
```

[49]:
```
print(classification_report(y_test, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.50      0.64      0.56     15066
           1       0.50      0.36      0.41     14934

    accuracy                           0.50     30000
   macro avg       0.50      0.50      0.49     30000
weighted avg       0.50      0.50      0.49     30000
```

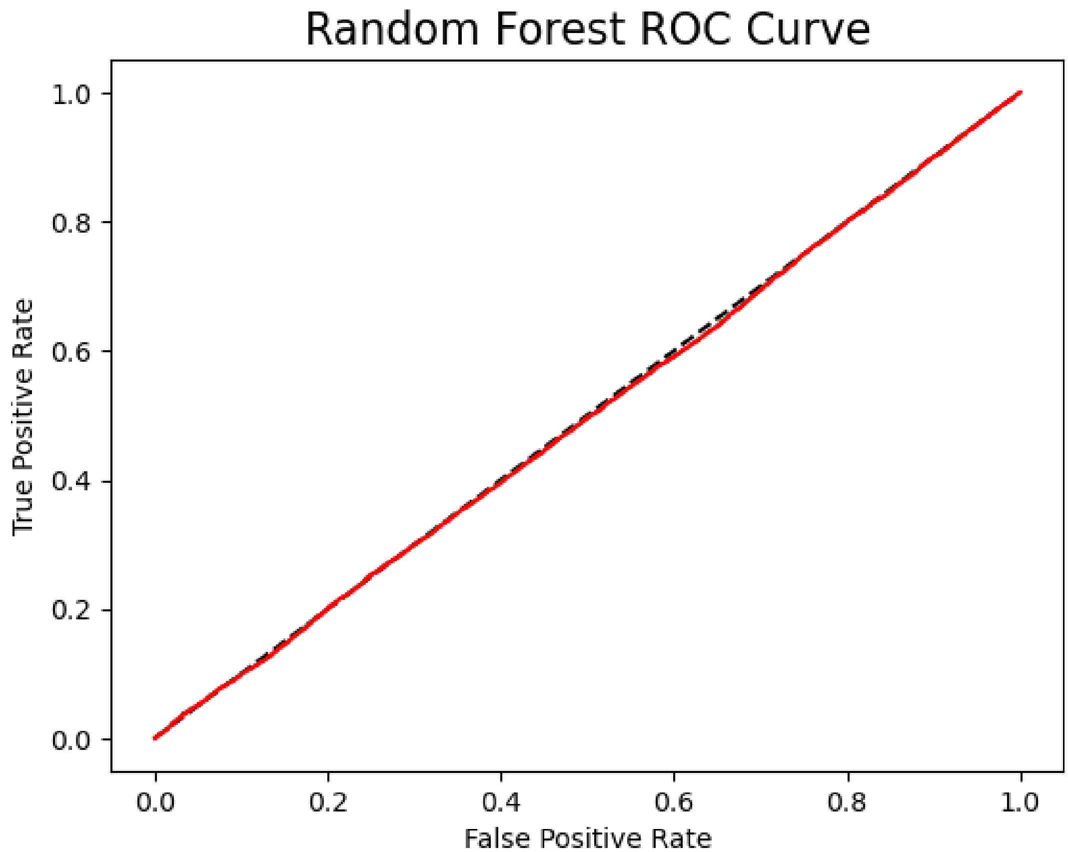
[50]:

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, prediction_test),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title(" RANDOM FOREST CONFUSION MATRIX",fontsize=14)
plt.show()
```



[51]:

```
y_rfpred_prob = model_rf.predict_proba(X_test)[:,1]
fpr_rf, tpr_rf, thresholds = roc_curve(y_test, y_rfpred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr_rf, tpr_rf, label='Random Forest',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve',fontsize=16)
plt.show();
```



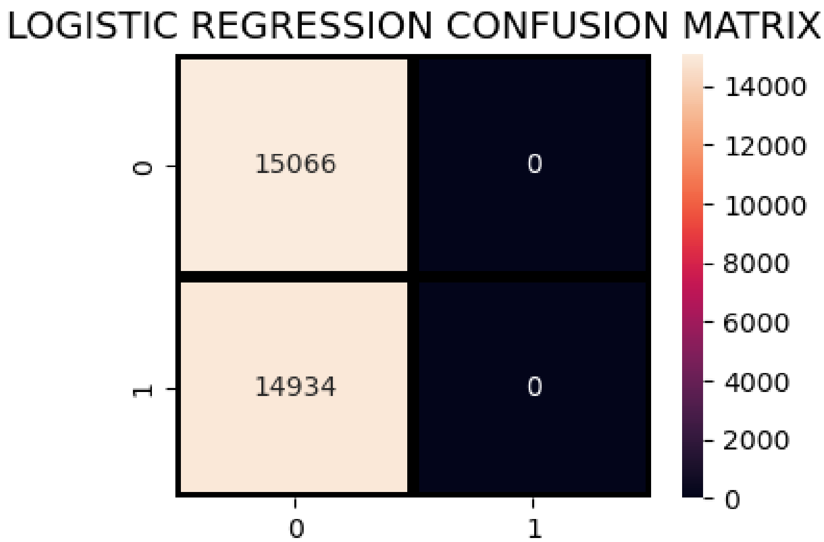
[52]:
```
lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
accuracy_lr = lr_model.score(X_test,y_test)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.5022
```

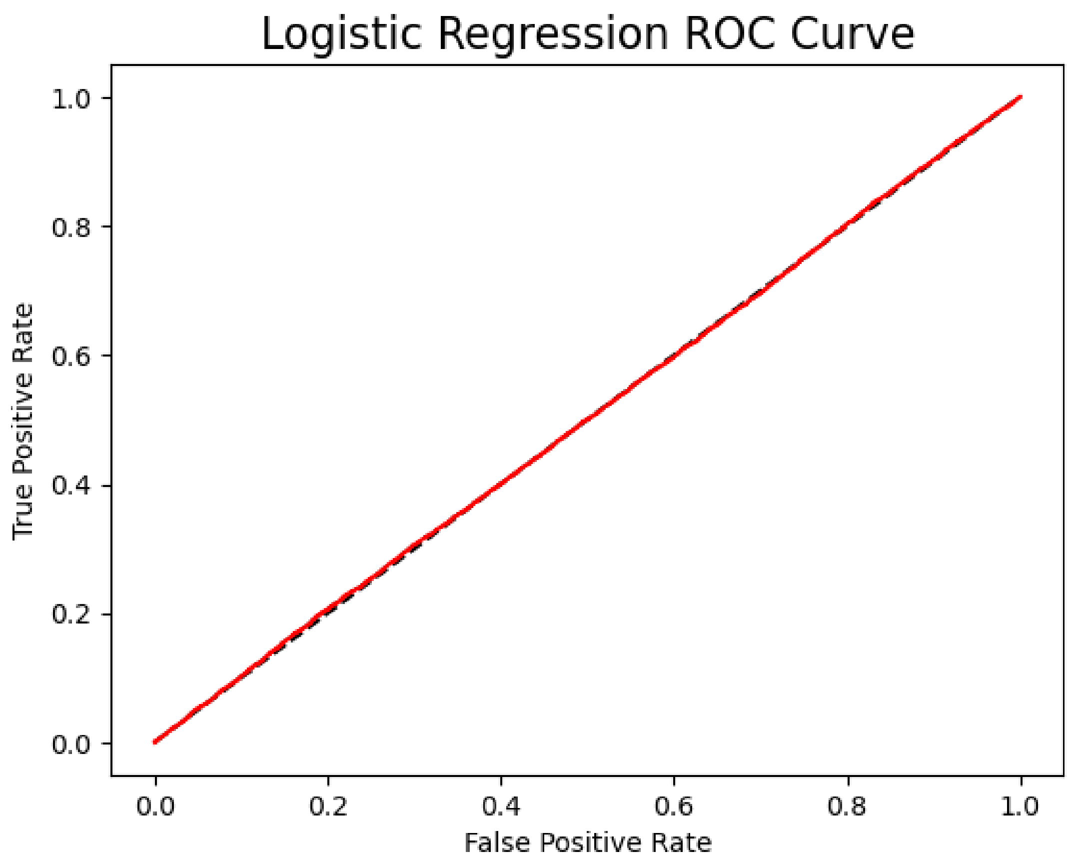
[53]:
```
lr_pred= lr_model.predict(X_test)
report = classification_report(y_test,lr_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67     15066
           1       0.00      0.00      0.00     14934

    accuracy                           0.50     30000
   macro avg       0.25      0.50      0.33     30000
weighted avg       0.25      0.50      0.34     30000
```

[54]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, lr_pred),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("LOGISTIC REGRESSION CONFUSION MATRIX",fontsize=14)
plt.show()
```



[55]:
```
y_pred_prob = lr_model.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr, tpr, label='Logistic Regression',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve',fontsize=16)
plt.show();
```



[56]:
```
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train,y_train)
predictdt_y = dt_model.predict(X_test)
accuracy_dt = dt_model.score(X_test,y_test)
print("Decision Tree accuracy is :",accuracy_dt)
```

```
Decision Tree accuracy is : 0.4982
```

[57]:
```
print(classification_report(y_test, predictdt_y))
```

```
              precision    recall  f1-score   support

           0       0.50      0.50      0.50     15066
           1       0.50      0.49      0.50     14934

    accuracy                           0.50     30000
   macro avg       0.50      0.50      0.50     30000
weighted avg       0.50      0.50      0.50     30000
```

[58]:
```
a_model = AdaBoostClassifier()
a_model.fit(X_train,y_train)
a_preds = a_model.predict(X_test)
print("AdaBoost Classifier accuracy")
metrics.accuracy_score(y_test, a_preds)
```

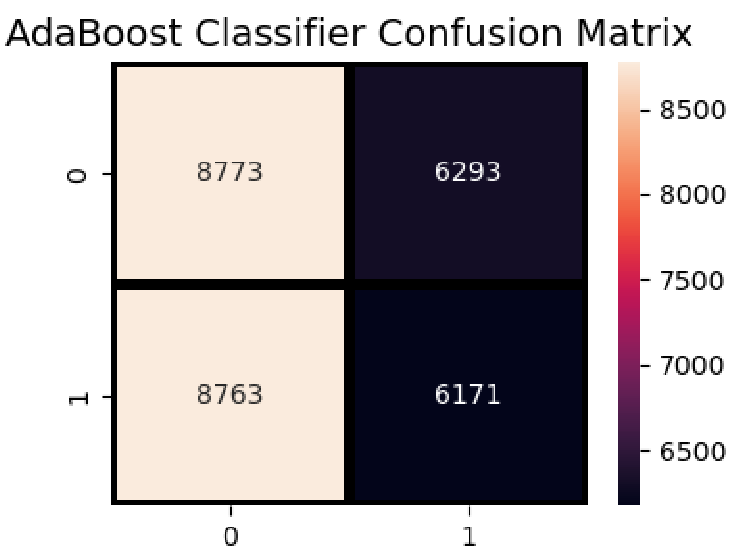
```
AdaBoost Classifier accuracy
```

[58]: 0.4981333333333333

[59]:
```
print(classification_report(y_test, a_preds))
```

```
              precision    recall  f1-score   support

           0       0.50      0.58      0.54     15066
           1       0.50      0.41      0.45     14934

    accuracy                           0.50     30000
   macro avg       0.50      0.50      0.49     30000
weighted avg       0.50      0.50      0.49     30000
```

[60]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, a_preds),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("AdaBoost Classifier Confusion Matrix",fontsize=14)
plt.show()
```



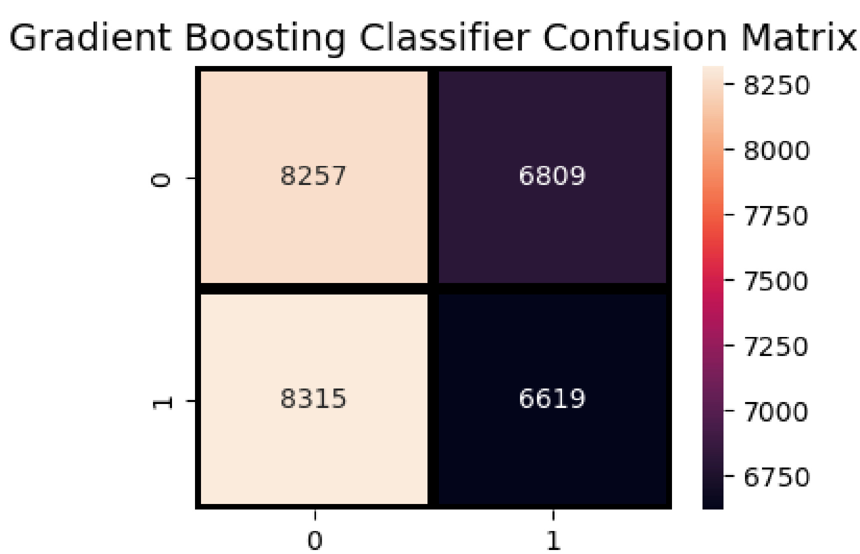
[61]:
```
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)
gb_pred = gb.predict(X_test)
print("Gradient Boosting Classifier", accuracy_score(y_test, gb_pred))
```

```
Gradient Boosting Classifier 0.4958666666666667
```

[62]:
```
print(classification_report(y_test, gb_pred))
```

```
              precision    recall  f1-score   support

           0       0.50      0.55      0.52     15066
           1       0.49      0.44      0.47     14934

    accuracy                           0.50     30000
   macro avg       0.50      0.50      0.49     30000
weighted avg       0.50      0.50      0.49     30000
```

[63]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, gb_pred),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("Gradient Boosting Classifier Confusion Matrix",fontsize=14)
plt.show()
```



[64]:
```
from sklearn.ensemble import VotingClassifier
clf1 = GradientBoostingClassifier()
```

```
clf2 = LogisticRegression()
clf3 = AdaBoostClassifier()
eclf1 = VotingClassifier(estimators=[('gbc', clf1), ('lr', clf2), ('abc',␣
 ↪clf3)], voting='soft')
eclf1.fit(X_train, y_train)
predictions = eclf1.predict(X_test)
print("Final Accuracy Score ")
print(accuracy_score(y_test, predictions))
```

```
Final Accuracy Score
0.4968
```

[65]:
```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.50      0.61      0.55     15066
           1       0.49      0.38      0.43     14934

    accuracy                           0.50     30000
   macro avg       0.50      0.50      0.49     30000
weighted avg       0.50      0.50      0.49     30000
```

[66]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, predictions),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("FINAL CONFUSION MATRIX",fontsize=14)
plt.show()
```