In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import classification_report,confusion_matrix
```

In [2]:
```
df=pd.read_csv('data1.csv')
df.drop(columns='Unnamed: 0',inplace=True)
df.drop(columns='id',inplace=True)
df.drop(columns='origin_up',inplace=True)
df.drop(columns='activity_new',inplace=True)
df.drop(columns='channel_sales',inplace=True)
df.drop(columns='date_activ',inplace=True)
df.drop(columns='date_end',inplace=True)
df.drop(columns='date_first_activ',inplace=True)
df.drop(columns='date_modif_prod',inplace=True)
df.drop(columns='date_renewal',inplace=True)
df.head()
```

Out[2]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele	forecast_base_bill_year	forecast_
0	12.641989	0.0	9.212937	5.819619	5.819619	
1	5.485229	0.0	0.000000	5.819619	5.819619	
2	8.446985	0.0	0.000000	5.819619	5.819619	
3	6.300786	0.0	0.000000	5.819619	5.819619	
4	7.368340	0.0	0.000000	5.819619	5.819619	

5 rows × 41 columns

In [3]:
```
df.describe()
```

Out[3]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele	forecast_base_bill_year	f
count	16096.000000	16096.0	16096.000000	1.609500e+04	1.609500e+04	
mean	9.887019	0.0	5.432446	5.819619e+00	5.819619e+00	
std	1.836269	0.0	4.113587	1.769306e-12	1.769306e-12	
min	5.485229	0.0	0.000000	5.819619e+00	5.819619e+00	
25%	8.690642	0.0	0.000000	5.819619e+00	5.819619e+00	
50%	9.644263	0.0	6.804615	5.819619e+00	5.819619e+00	
75%	10.824218	0.0	8.325548	5.819619e+00	5.819619e+00	
max	14.029856	0.0	15.328156	5.819619e+00	5.819619e+00	

8 rows × 41 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16096 entries, 0 to 16095
Data columns (total 41 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   cons_12m                              16096 non-null  float64
 1   cons_gas_12m                          16096 non-null  float64
 2   cons_last_month                       16096 non-null  float64
 3   forecast_base_bill_ele                16095 non-null  float64
 4   forecast_base_bill_year               16095 non-null  float64
 5   forecast_bill_12m                     16094 non-null  float64
 6   forecast_cons                         16096 non-null  float64
 7   forecast_cons_12m                     16096 non-null  float64
 8   forecast_cons_year                    16096 non-null  float64
 9   forecast_discount_energy              16096 non-null  float64
 10  forecast_meter_rent_12m               16096 non-null  float64
 11  forecast_price_energy_p1              16096 non-null  float64
 12  forecast_price_energy_p2              16096 non-null  float64
 13  forecast_price_pow_p1                 16095 non-null  float64
 14  has_gas                               16096 non-null  int64  
 15  imp_cons                              16096 non-null  float64
 16  margin_gross_pow_ele                  16096 non-null  float64
 17  margin_net_pow_ele                    16096 non-null  float64
 18  nb_prod_act                           16096 non-null  float64
 19  net_margin                            16096 non-null  float64
 20  num_years_antig                       16096 non-null  float64
 21  pow_max                               16096 non-null  float64
 22  churn                                 16096 non-null  int64  
 23  apdekpcbwosbxepsfxclislboipuxpop_x    16096 non-null  int64  
 24  ckfxocssowaeipxueikxcmaxdmcduxsa_x    16096 non-null  int64  
 25  cluecxlameloamldmasudocsbmaoamdw_x    16096 non-null  int64  
 26  cwofmuicebbcmiaaxufmfimpowpacobu_x    16096 non-null  int64  
 27  fmwdwsxillemwbbwelxsampiuwwpcdcb_x    16096 non-null  int64  
 28  kkklcdamwfafdcfwofuscwfwadblfmce_x    16096 non-null  int64  
 29  kwuslieomapmswolewpobpplkaooaaew_x    16096 non-null  int64  
 30  sfisfxfcocfpcmckuekokxuseixdaoeu_x    16096 non-null  int64  
 31  wxemiwkumpibllwklfbcooafckufkdlm_x    16096 non-null  int64  
 32  apdekpcbwosbxepsfxclislboipuxpop_y    16096 non-null  int64  
 33  ckfxocssowaeipxueikxcmaxdmcduxsa_y    16096 non-null  int64  
 34  cluecxlameloamldmasudocsbmaoamdw_y    16096 non-null  int64  
 35  cwofmuicebbcmiaaxufmfimpowpacobu_y    16096 non-null  int64  
 36  fmwdwsxillemwbbwelxsampiuwwpcdcb_y    16096 non-null  int64  
 37  kkklcdamwfafdcfwofuscwfwadblfmce_y    16096 non-null  int64  
 38  kwuslieomapmswolewpobpplkaooaaew_y    16096 non-null  int64  
 39  sfisfxfcocfpcmckuekokxuseixdaoeu_y    16096 non-null  int64  
 40  wxemiwkumpibllwklfbcooafckufkdlm_y    16096 non-null  int64  
dtypes: float64(21), int64(20)
memory usage: 5.0 MB
```

In [5]:

```
cols=['cons_12m','cons_gas_12m','cons_last_month','forecast_base_bill_ele','forecast
     'forecast_cons_year','forecast_discount_energy','forecast_meter_rent_12m','fore
     'forecast_price_energy_p2','forecast_price_pow_p1']
from sklearn.preprocessing import MinMaxScaler
scalar=MinMaxScaler()
df[cols]=scalar.fit_transform(df[cols])
```

In [6]:

```
x=df.drop('churn',axis='columns')
y=df['churn']
```

In [7]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```

In [8]:

```
x_train.shape
```

Out[8]: (12876, 40)

In [9]:
```
y_train.shape
```

Out[9]: (12876,)

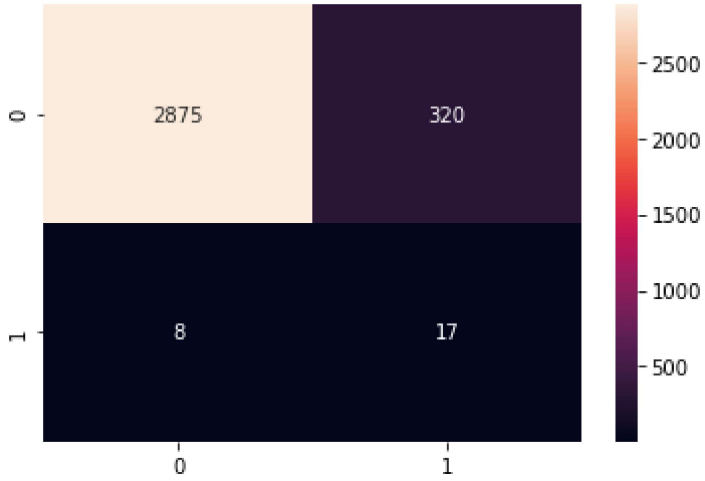
In [10]:
```
import xgboost as xgb
model=xgb.XGBClassifier(learning_rate=0.1,max_depth=6,estimators=500,n_jobs=-1)
model.fit(x_train,y_train)
```

```
C:\Users\vedan\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The
use of label encoder in XGBClassifier is deprecated and will be removed in a future
release. To remove this warning, do the following: 1) Pass option use_label_encoder=
False when constructing XGBClassifier object; and 2) Encode your labels (y) as integ
ers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[20:33:52] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/learner.cc:573:
Parameters: { "estimators" } might not be used.

  This may not be accurate due to some parameters are only used in language bindings
but
  passed down to XGBoost core.  Or some parameters are not used but slip through thi
s
  verification. Please open an issue if you find above cases.


[20:33:52] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used wit
h the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly
set eval_metric if you'd like to restore the old behavior.
```

Out[10]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, estimators=500, gamma=0,
              gpu_id=-1, importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [12]:
```
y_pred=model.predict(x_test)
```

In [13]:
```
cf=confusion_matrix(y_pred,y_test)
import seaborn as sb
sb.heatmap(cf,annot=True,fmt='d');
```



In [14]:
```
print(classification_report(y_pred,y_test));
```

```
              precision    recall  f1-score   support

           0       1.00      0.90      0.95      3195
           1       0.05      0.68      0.09        25

    accuracy                           0.90      3220
   macro avg       0.52      0.79      0.52      3220
weighted avg       0.99      0.90      0.94      3220
```

In confusion matrix we can see f1 score for both churn & not churn is 0.95 & 0.09 resp. So, that means our dataset is imbalanced.

In [15]:
```
y_test.value_counts()
```

Out[15]:
```
0    2883
1     337
Name: churn, dtype: int64
```

We are having 2883 samples of zeroth class and 337 samples of first class.

We will oversample minority class which is churn or first class.

In [16]:
```
class_counts_0,class_counts_1=df.churn.value_counts()
class_counts_0,class_counts_1
```

Out[16]: (14501, 1595)

In [17]:
```
#divide by class
df_class_0=df[df['churn']==0]
df_class_1=df[df['churn']==1]
```

In [18]:
```
df_class_0.shape,df_class_1.shape
```

Out[18]: ((14501, 41), (1595, 41))

In [19]:
```
class_counts_0,class_counts_1
```

Out[19]: (14501, 1595)

In [20]:
```
#Oversampling 1595 to 14501 samples by using argument replace=True in sample functio
df_class_1_over=df_class_1.sample(class_counts_0,replace=True)
df_class_1_over.shape
```

Out[20]:
```
(14501, 41)
```

In [21]:
```
#combining oversampled dataframe of class 1 with class 0
df_test_over=pd.concat([df_class_0,df_class_1_over],axis=0)
df_test_over.shape
```

Out[21]:
```
(29002, 41)
```

In [22]:
```
df_test_over.churn.value_counts()
```

Out[22]:
```
0    14501
1    14501
Name: churn, dtype: int64
```

In [23]:
```
x=df_test_over.drop('churn',axis=1)
y=df_test_over['churn']


x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=15,test_size=0.2,str
```

In [24]:
```
import xgboost as xgb
model=xgb.XGBClassifier(learning_rate=0.1,max_depth=6,estimators=500,n_jobs=-1)
model.fit(x_train,y_train)
```

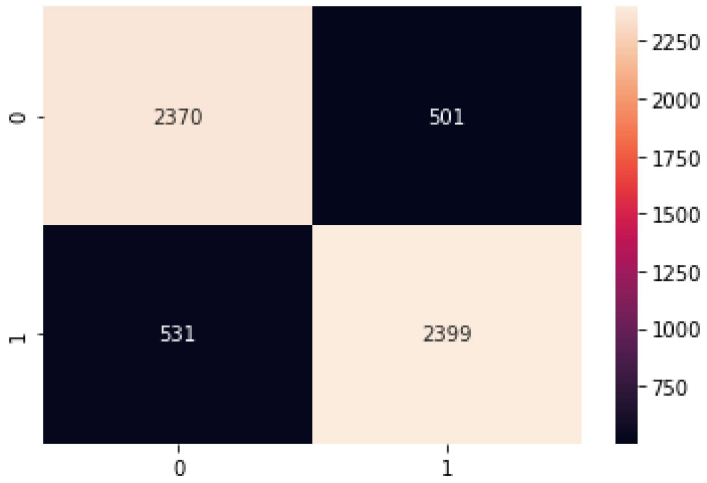
```
C:\Users\vedan\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The
use of label encoder in XGBClassifier is deprecated and will be removed in a future
release. To remove this warning, do the following: 1) Pass option use_label_encoder=
False when constructing XGBClassifier object; and 2) Encode your labels (y) as integ
ers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[20:37:53] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/learner.cc:573:
Parameters: { "estimators" } might not be used.

  This may not be accurate due to some parameters are only used in language bindings
but
  passed down to XGBoost core.  Or some parameters are not used but slip through thi
s
  verification. Please open an issue if you find above cases.


[20:37:53] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used wit
h the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly
set eval_metric if you'd like to restore the old behavior.
```

Out[24]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, estimators=500, gamma=0,
              gpu_id=-1, importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [25]:
```
y_pred=model.predict(x_test)
print(classification_report(y_pred,y_test));
```

```
              precision    recall  f1-score   support

           0       0.82      0.83      0.82      2871
           1       0.83      0.82      0.82      2930

    accuracy                           0.82      5801
   macro avg       0.82      0.82      0.82      5801
weighted avg       0.82      0.82      0.82      5801
```

In [26]:
```
cf=confusion_matrix(y_pred,y_test)
import seaborn as sb
sb.heatmap(cf,annot=True,fmt='d');
```



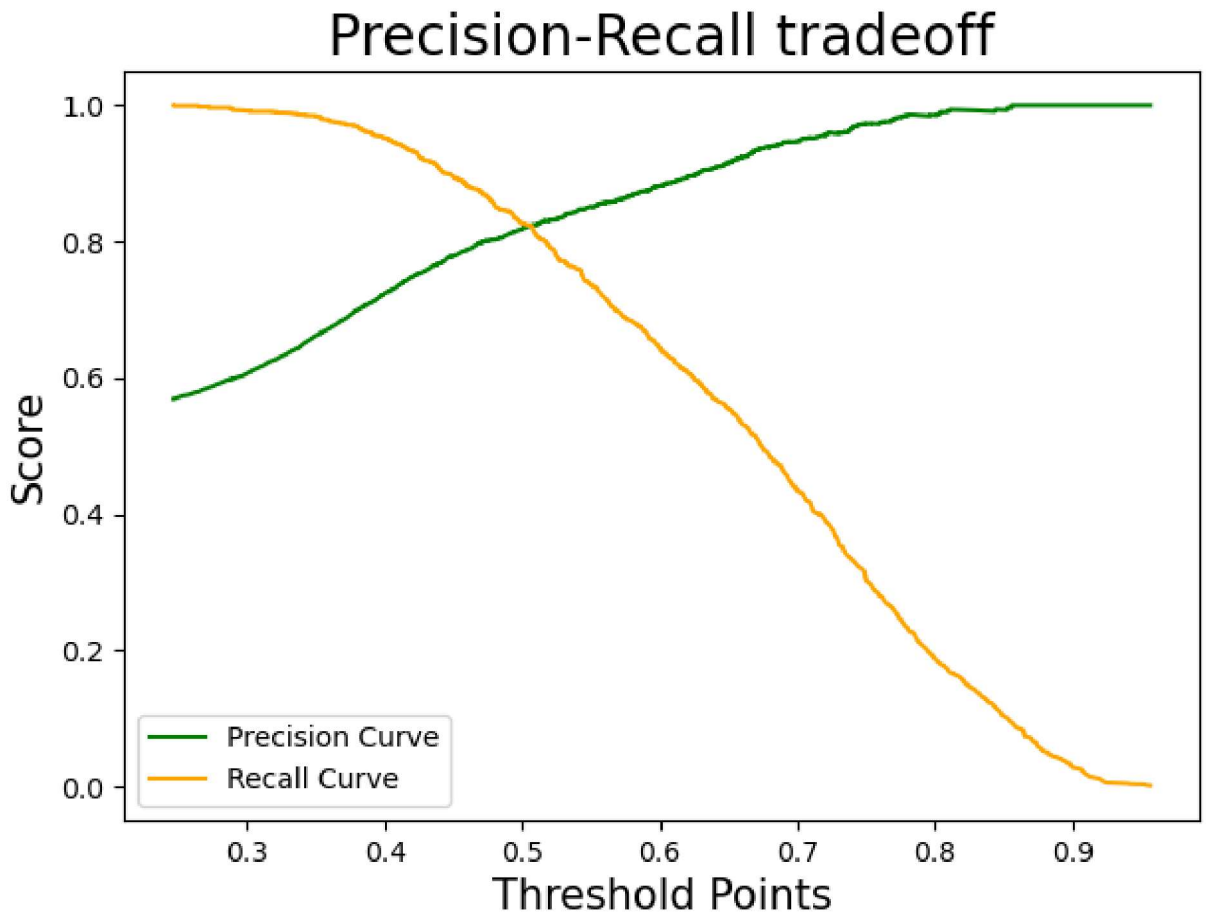
In [27]:
```
predicted_probabilities = model.predict_proba(x_test)
```

In [28]:
```
from sklearn.metrics import precision_recall_curve
precision_points, recall_points, threshold_points = precision_recall_curve(y_test, p
precision_points.shape, recall_points.shape, threshold_points.shape
```

Out[28]: ((3516,), (3516,), (3515,))

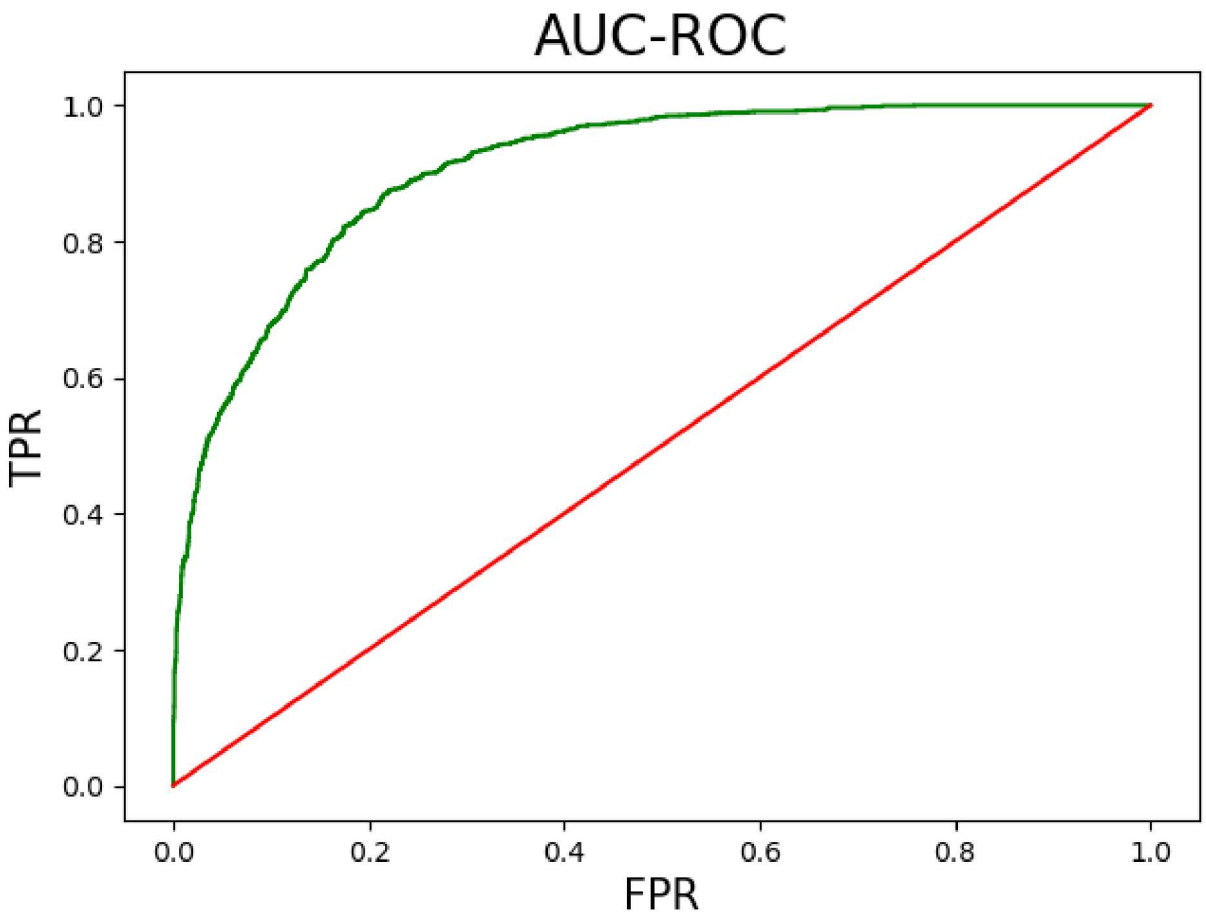
In [29]:
```
plt.figure(figsize = (7,5), dpi = 100)
plt.plot( threshold_points, precision_points[:-1], color = 'green', label = 'Precisi
plt.plot( threshold_points, recall_points[:-1], color = 'orange', label = 'Recall Cu
plt.xlabel('Threshold Points', fontsize = 15)
plt.ylabel('Score', fontsize = 15)
plt.title('Precision-Recall tradeoff', fontsize = 20)
plt.legend()
```

Out[29]: <matplotlib.legend.Legend at 0x23525e4c790>



In [30]:
```
from sklearn.metrics import roc_curve , roc_auc_score
fpr, tpr, threshold = roc_curve(y_test, predicted_probabilities[:,1])
```

In [31]:
```
plt.figure(figsize = (7,5), dpi = 100)
plt.plot( fpr, tpr, color = 'green')
plt.plot( [0,1], [0,1], label = 'baseline', color = 'red')
plt.xlabel('FPR', fontsize = 15)
plt.ylabel('TPR', fontsize = 15)
plt.title('AUC-ROC', fontsize = 20)
plt.show()
roc_auc_score(y_test, predicted_probabilities[:,1])
```



Out[31]: 0.9076058196341333

AUC-ROC score is 0.9076058196341333