Building the final model for Churn prediction

In this module, our final task is to build a random forest model for predicting whether a customer will churn away or not. To improve model performance hyperparameter tuning will be performed and the best parameters that give best accuracy will be chosen to build final model.

In [64]:
```
#getting the packages
import pandas as pd
import numpy as np
import seaborn as sns
from datetime import datetime
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
%matplotlib inline
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
sns.set(color_codes=True)
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingGridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

In [65]:
```
#obtaining the dataset
df = pd.read_csv('C:\\Users\\sujoydutta\\Downloads\\modifiedenergydata.csv')
df.head()
```

Out[65]:

	id	cons_12m	cons_gas_12m	cons_last_month	date_activ	date_end	date_mod
0	24011ae4ebbe3035111d65fa7c15bc57	0.000000	0.92993	0.00000	2013-06-15	2016-06-15	201
1	d29c2c54acc38ff3c0614d0a653813dd	4.704260	-0.00000	0.00000	2009-08-21	2016-08-30	200
2	764c75f661154dac3a6c254cd082ea7d	3.451355	-0.00000	0.00000	2010-04-16	2016-04-16	201
3	bba03439a292a1e166f80264c16191cb	4.062661	-0.00000	0.00000	2010-03-30	2016-03-30	201
4	149d57cf92fc41cf94415803a877cb4b	4.672976	-0.00000	3.55748	2010-01-13	2016-03-07	201

5 rows × 64 columns

In [66]:
```
#seeing df info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 64 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   id                                          14606 non-null  object
 1   cons_12m                                    14606 non-null  float64
 2   cons_gas_12m                                14606 non-null  float64
 3   cons_last_month                             14606 non-null  float64
```

```
 4   date_activ                                  14606 non-null  object
 5   date_end                                    14606 non-null  object
 6   date_modif_prod                             14606 non-null  object
 7   date_renewal                                14606 non-null  object
 8   forecast_cons_12m                           14606 non-null  float64
 9   forecast_cons_year                          14606 non-null  float64
 10  forecast_discount_energy                    14606 non-null  float64
 11  forecast_meter_rent_12m                     14606 non-null  float64
 12  forecast_price_energy_off_peak              14606 non-null  float64
 13  forecast_price_energy_peak                  14606 non-null  float64
 14  forecast_price_pow_off_peak                 14606 non-null  float64
 15  has_gas                                     14606 non-null  float64
 16  imp_cons                                    14606 non-null  float64
 17  margin_gross_pow_ele                        14606 non-null  float64
 18  margin_net_pow_ele                          14606 non-null  float64
 19  nb_prod_act                                 14606 non-null  float64
 20  net_margin                                  14606 non-null  float64
 21  num_years_antig                             14606 non-null  float64
 22  pow_max                                     14606 non-null  float64
 23  var_year_price_off_peak_var                 14606 non-null  float64
 24  var_year_price_peak_var                     14606 non-null  float64
 25  var_year_price_mid_peak_var                 14606 non-null  float64
 26  var_year_price_off_peak_fix                 14606 non-null  float64
 27  var_year_price_peak_fix                     14606 non-null  float64
 28  var_year_price_mid_peak_fix                 14606 non-null  float64
 29  var_year_price_off_peak                     14606 non-null  float64
 30  var_year_price_peak                         14606 non-null  float64
 31  var_year_price_mid_peak                     14606 non-null  float64
 32  var_6m_price_off_peak_var                   14606 non-null  float64
 33  var_6m_price_peak_var                       14606 non-null  float64
 34  var_6m_price_mid_peak_var                   14606 non-null  float64
 35  var_6m_price_off_peak_fix                   14606 non-null  float64
 36  var_6m_price_peak_fix                       14606 non-null  float64
 37  var_6m_price_mid_peak_fix                   14606 non-null  float64
 38  var_6m_price_off_peak                       14606 non-null  float64
 39  var_6m_price_peak                           14606 non-null  float64
 40  var_6m_price_mid_peak                       14606 non-null  float64
 41  churn                                       14606 non-null  float64
 42  offpeak_diff_dec_january_energy             14606 non-null  float64
 43  offpeak_diff_dec_january_power              14606 non-null  float64
 44  off_peak_peak_var_max_monthly_diff          14606 non-null  float64
 45  peak_mid_peak_var_max_monthly_diff          14606 non-null  float64
 46  off_peak_mid_peak_var_max_monthly_diff      14606 non-null  float64
 47  off_peak_peak_fix_max_monthly_diff          14606 non-null  float64
 48  peak_mid_peak_fix_max_monthly_diff          14606 non-null  float64
 49  off_peak_mid_peak_fix_max_monthly_diff      14606 non-null  float64
 50  customer_tenure_days                        14606 non-null  float64
 51  time_to_renewal_days                        14606 non-null  int64
 52  avg_monthly_consumption                     14606 non-null  float64
 53  forecast_error                              14606 non-null  float64
 54  consumption_change                          14606 non-null  float64
 55  product_diversity                           14606 non-null  float64
 56  channel_MISSING                             14606 non-null  bool
 57  channel_ewpakwlliwisiwduibdlfmalxowmwpci    14606 non-null  bool
 58  channel_foosdfpfkusacimwkcsosbicdxkicaua    14606 non-null  bool
 59  channel_lmkebamcaaclubfxadlmueccxoimlema    14606 non-null  bool
 60  channel_usilxuppasemubllopkaafesmlibmsdf    14606 non-null  bool
 61  origin_up_kamkkxfxxuwbdslkwifmmcsiusiuosws  14606 non-null  bool
 62  origin_up_ldkssxwpmemidmecebumciepifcamkci  14606 non-null  bool
 63  origin_up_lxidpiddsbxsbosboudacockeimpuepw  14606 non-null  bool
dtypes: bool(8), float64(50), int64(1), object(5)
memory usage: 6.4+ MB
```

In [67]:

```
#dropping columns that may hamper random forest
df = df.drop(columns=['id'])
df = df.drop(columns=['date_activ', 'date_end', 'date_modif_prod', 'date_renewal'])
```

In [68]:
```
#Modifying columns with binary values
df['channel_MISSING'] = df['channel_MISSING'].replace([True, False], [1, 0])
df['channel_ewpakwlliwisiwduibdlfmalxowmwpci'] = df['channel_ewpakwlliwisiwduibdlfmalxow
df['channel_foosdfpfkusacimwkcsosbicdxkicaua'] = df['channel_foosdfpfkusacimwkcsosbicdxk
df['channel_lmkebamcaaclubfxadlmueccxoimlema'] = df['channel_lmkebamcaaclubfxadlmueccxoi
df['channel_usilxuppasemubllopkaafesmlibmsdf'] = df['channel_usilxuppasemubllopkaafesmli
df['origin_up_kamkkxfxxuwbdslkwifmmcsiusiuosws'] = df['origin_up_kamkkxfxxuwbdslkwifmmcs
df['origin_up_lxidpiddsbxsbosboudacockeimpuepw'] = df['origin_up_lxidpiddsbxsbosboudacoc
df['origin_up_ldkssxwpmemidmecebumciepifcamkci'] = df['origin_up_ldkssxwpmemidmecebumcie
```

In [69]:
```
#examining new dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 59 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   cons_12m                                    14606 non-null  float64
 1   cons_gas_12m                                14606 non-null  float64
 2   cons_last_month                             14606 non-null  float64
 3   forecast_cons_12m                           14606 non-null  float64
 4   forecast_cons_year                          14606 non-null  float64
 5   forecast_discount_energy                    14606 non-null  float64
 6   forecast_meter_rent_12m                     14606 non-null  float64
 7   forecast_price_energy_off_peak              14606 non-null  float64
 8   forecast_price_energy_peak                  14606 non-null  float64
 9   forecast_price_pow_off_peak                 14606 non-null  float64
 10  has_gas                                     14606 non-null  float64
 11  imp_cons                                    14606 non-null  float64
 12  margin_gross_pow_ele                        14606 non-null  float64
 13  margin_net_pow_ele                          14606 non-null  float64
 14  nb_prod_act                                 14606 non-null  float64
 15  net_margin                                  14606 non-null  float64
 16  num_years_antig                             14606 non-null  float64
 17  pow_max                                     14606 non-null  float64
 18  var_year_price_off_peak_var                 14606 non-null  float64
 19  var_year_price_peak_var                     14606 non-null  float64
 20  var_year_price_mid_peak_var                 14606 non-null  float64
 21  var_year_price_off_peak_fix                 14606 non-null  float64
 22  var_year_price_peak_fix                     14606 non-null  float64
 23  var_year_price_mid_peak_fix                 14606 non-null  float64
 24  var_year_price_off_peak                     14606 non-null  float64
 25  var_year_price_peak                         14606 non-null  float64
 26  var_year_price_mid_peak                     14606 non-null  float64
 27  var_6m_price_off_peak_var                   14606 non-null  float64
 28  var_6m_price_peak_var                       14606 non-null  float64
 29  var_6m_price_mid_peak_var                   14606 non-null  float64
 30  var_6m_price_off_peak_fix                   14606 non-null  float64
 31  var_6m_price_peak_fix                       14606 non-null  float64
 32  var_6m_price_mid_peak_fix                   14606 non-null  float64
 33  var_6m_price_off_peak                       14606 non-null  float64
 34  var_6m_price_peak                           14606 non-null  float64
 35  var_6m_price_mid_peak                       14606 non-null  float64
 36  churn                                       14606 non-null  float64
 37  offpeak_diff_dec_january_energy             14606 non-null  float64
 38  offpeak_diff_dec_january_power              14606 non-null  float64
 39  off_peak_peak_var_max_monthly_diff          14606 non-null  float64
 40  peak_mid_peak_var_max_monthly_diff          14606 non-null  float64
 41  off_peak_mid_peak_var_max_monthly_diff      14606 non-null  float64
 42  off_peak_peak_fix_max_monthly_diff          14606 non-null  float64
 43  peak_mid_peak_fix_max_monthly_diff          14606 non-null  float64
 44  off_peak_mid_peak_fix_max_monthly_diff      14606 non-null  float64
 45  customer_tenure_days                        14606 non-null  float64
```

```
 46  time_to_renewal_days                          14606 non-null  int64
 47  avg_monthly_consumption                       14606 non-null  float64
 48  forecast_error                                14606 non-null  float64
 49  consumption_change                            14606 non-null  float64
 50  product_diversity                             14606 non-null  float64
 51  channel_MISSING                               14606 non-null  int64
 52  channel_ewpakwlliwisiwduibdlfmalxowmwpci      14606 non-null  int64
 53  channel_foosdfpfkusacimwkcsosbicdxkicaua      14606 non-null  int64
 54  channel_lmkebamcaaclubfxadlmueccxoimlema      14606 non-null  int64
 55  channel_usilxuppasemubllopkaafesmlibmsdf      14606 non-null  int64
 56  origin_up_kamkkxfxxuwbdslkwifmmcsiusiuosws    14606 non-null  int64
 57  origin_up_ldkssxwpmemidmecebumciepifcamkci    14606 non-null  int64
 58  origin_up_lxidpiddsbxsbosboudacockeimpuepw    14606 non-null  int64
dtypes: float64(50), int64(9)
memory usage: 6.6 MB
```

In [70]:
```
# converting churn to integer format
df.churn = df.churn.astype(int)
df.churn
```

Out[70]:
```
0        1
1        0
2        0
3        0
4        0
        ..
14601    0
14602    1
14603    1
14604    0
14605    0
Name: churn, Length: 14606, dtype: int32
```

In [71]:
```
#Splitting the dataset into X and y
X=df.drop(['churn'],axis=1)
y=df.churn
```

In [72]:
```
columns_to_scale = [
    'cons_12m', 'cons_gas_12m', 'cons_last_month', 'forecast_cons_12m',
    'forecast_cons_year', 'forecast_discount_energy', 'forecast_meter_rent_12m',
    'forecast_price_energy_off_peak', 'forecast_price_energy_peak',
    'forecast_price_pow_off_peak', 'has_gas', 'imp_cons', 'margin_gross_pow_ele',
    'margin_net_pow_ele', 'nb_prod_act', 'net_margin', 'num_years_antig',
    'pow_max', 'var_year_price_off_peak_var', 'var_year_price_peak_var',
    'var_year_price_mid_peak_var', 'var_year_price_off_peak_fix',
    'var_year_price_peak_fix', 'var_year_price_mid_peak_fix',
    'var_year_price_off_peak', 'var_year_price_peak', 'var_year_price_mid_peak',
    'var_6m_price_off_peak_var', 'var_6m_price_peak_var',
    'var_6m_price_mid_peak_var', 'var_6m_price_off_peak_fix',
    'var_6m_price_peak_fix', 'var_6m_price_mid_peak_fix', 'var_6m_price_off_peak',
    'var_6m_price_peak', 'var_6m_price_mid_peak',
    'offpeak_diff_dec_january_energy', 'offpeak_diff_dec_january_power',
    'off_peak_peak_var_max_monthly_diff', 'peak_mid_peak_var_max_monthly_diff',
    'off_peak_mid_peak_var_max_monthly_diff',
    'off_peak_peak_fix_max_monthly_diff',
    'peak_mid_peak_fix_max_monthly_diff',
    'off_peak_mid_peak_fix_max_monthly_diff', 'customer_tenure_days',
    'time_to_renewal_days', 'avg_monthly_consumption',
    'forecast_error', 'consumption_change', 'product_diversity'
]
```

In [73]:
```
#initializing the scaler
scaler = StandardScaler()
```

In [74]:
```
# Scaling the selected columns
```

```
X[columns_to_scale] = scaler.fit_transform(X[columns_to_scale])
```

In [75]:
```
#splitting dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=4
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(10954, 58)
(10954,)
(3652, 58)
(3652,)
```

In [76]:
```
# Initializing the RF model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [77]:
```
# Training the model on the training data
rf_model.fit(X_train, y_train)
```

Out[77]:
```
RandomForestClassifier(random_state=42)
```

In [78]:
```
# Predicting the churn on the test data
y_pred = rf_model.predict(X_test)
```

In [79]:
```
# Evaluating the model's performance
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
[[3283    3]
 [ 349   17]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      3286
           1       0.85      0.05      0.09       366

    accuracy                           0.90      3652
   macro avg       0.88      0.52      0.52      3652
weighted avg       0.90      0.90      0.86      3652


Accuracy Score:
0.9036144578313253
```

In [81]:
```
# Defining the parameter grid for Random Forest

param_grid = {
    'n_estimators': [200, 300, 500],
    'max_depth': [10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [2, 4, 6],
    'max_features': ['sqrt', 'log2'],
    'bootstrap': [True],
```

```
    'criterion': ['gini', 'entropy'],
    'class_weight': [
        'balanced',
        'balanced_subsample',
        {0: 1, 1: 2},
        {0: 1, 1: 3},
        {0: 1, 1: 4},
    ],
    }
```

In [82]:
```
# Initializing the Random Forest model
rf = RandomForestClassifier(random_state=42)
```

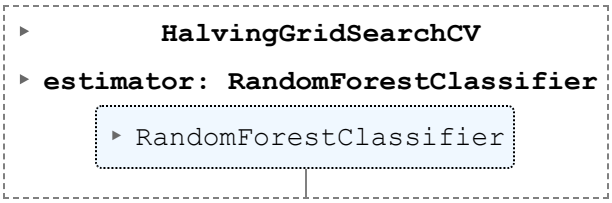
In [83]:
```
# Initializing Halving Grid Search
halving_search = HalvingGridSearchCV(
    rf,
    param_grid,
    factor=2,
    max_resources='auto',
    random_state=42,
    n_jobs=-1,
    cv=3,
    verbose=1
)
```

In [84]:
```
# Fitting the model using Halving Grid Search
halving_search.fit(X_train, y_train)
```

```
n_iterations: 10
n_required_iterations: 11
n_possible_iterations: 10
min_resources_: 12
max_resources_: 10954
aggressive_elimination: False
factor: 2
----------
iter: 0
n_candidates: 1620
n_resources: 12
Fitting 3 folds for each of 1620 candidates, totalling 4860 fits
----------
iter: 1
n_candidates: 810
n_resources: 24
Fitting 3 folds for each of 810 candidates, totalling 2430 fits
----------
iter: 2
n_candidates: 405
n_resources: 48
Fitting 3 folds for each of 405 candidates, totalling 1215 fits
----------
iter: 3
n_candidates: 203
n_resources: 96
Fitting 3 folds for each of 203 candidates, totalling 609 fits
----------
iter: 4
n_candidates: 102
n_resources: 192
Fitting 3 folds for each of 102 candidates, totalling 306 fits
----------
iter: 5
n_candidates: 51
n_resources: 384
Fitting 3 folds for each of 51 candidates, totalling 153 fits
```

```
----------
iter: 6
n_candidates: 26
n_resources: 768
Fitting 3 folds for each of 26 candidates, totalling 78 fits
----------
iter: 7
n_candidates: 13
n_resources: 1536
Fitting 3 folds for each of 13 candidates, totalling 39 fits
----------
iter: 8
n_candidates: 7
n_resources: 3072
Fitting 3 folds for each of 7 candidates, totalling 21 fits
----------
iter: 9
n_candidates: 4
n_resources: 6144
Fitting 3 folds for each of 4 candidates, totalling 12 fits
```

Out[84]:



In [86]:
```python
# Printing the best parameters and best score

best_rf_model = halving_search.best_estimator_
print(f"Best parameters: {halving_search.best_params_}")
print(f"Best score: {halving_search.best_score_}")
```

```
Best parameters: {'bootstrap': True, 'class_weight': 'balanced', 'criterion': 'entropy',
'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5,
'n_estimators': 500}
Best score: 0.9058939466801009
```

In [87]:
```python
# Initializing the best possible Random Forest model
bestrf = RandomForestClassifier(
    bootstrap=True,
    class_weight='balanced',
    criterion='entropy',
    max_depth=20,
    max_features='log2',
    min_samples_leaf=2,
    min_samples_split=5,
    n_estimators=500,
    random_state=42
)
```

In [88]:
```python
#  Fitting the model
bestrf.fit(X_train, y_train)
```

Out[88]:

RandomForestClassifier

```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=20, max_features='log2', min_samples_leaf=2,
                       min_samples_split=5, n_estimators=500, random_state=42)
```

In [91]:
```python
# Predictions on the test set
y_pred_new = bestrf.predict(X_test)
```

Out[91]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [90]:
```
# Evaluating the model's performance
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_new))

print("\nClassification Report:")
print(classification_report(y_test, y_pred_new))

print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred_new))
```

```
Confusion Matrix:
[[3274   12]
 [ 344   22]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      3286
           1       0.65      0.06      0.11       366

    accuracy                           0.90      3652
   macro avg       0.78      0.53      0.53      3652
weighted avg       0.88      0.90      0.86      3652


Accuracy Score:
0.9025191675794085
```