In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.pyplot import figure
import pickle
```

In [8]:
```
data = pd.read_csv('Churn_Modelling.csv')
```

In [9]:
```
data
```

Out[9]:

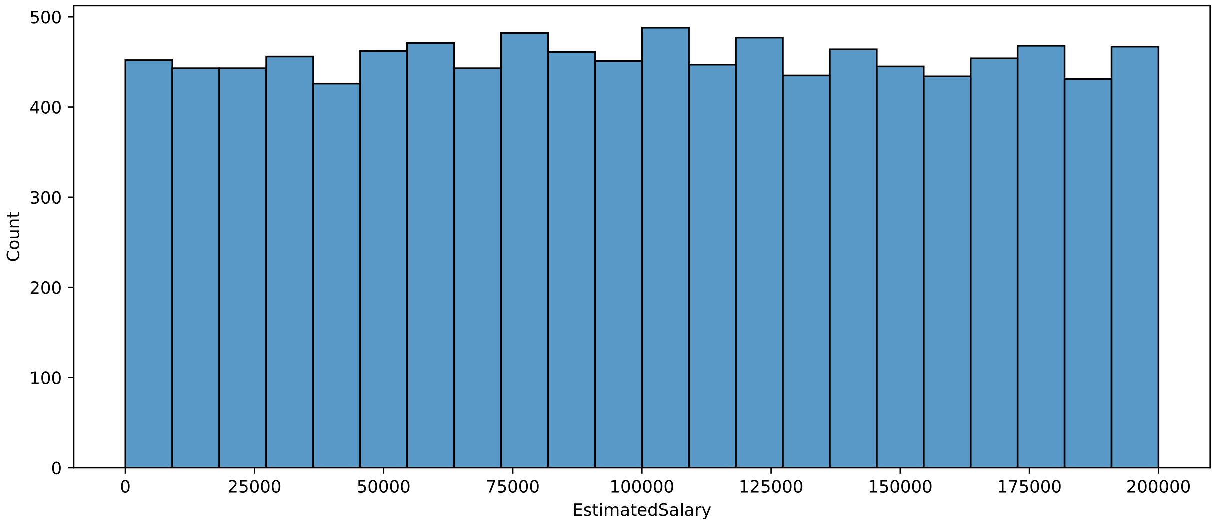
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	1	15634602	Hargrave	619	France	Female	42	2
1	2	15647311	Hill	608	Spain	Female	41	1
2	3	15619304	Onio	502	France	Female	42	8
3	4	15701354	Boni	699	France	Female	39	1
4	5	15737888	Mitchell	850	Spain	Female	43	2
...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5
9996	9997	15569892	Johnstone	516	France	Male	35	10
9997	9998	15584532	Liu	709	France	Female	36	7
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3
9999	10000	15628319	Walker	792	France	Female	28	4

10000 rows × 14 columns

In [10]:
```
tenure = []
for value in data['Tenure']:
    if((value >= 0) and (value < 3)):
        value = 'Low'
        tenure.append(value)
    elif((value >= 3) and (value <= 6)):
        value = 'Moderate'
        tenure.append(value)
    else:
        value = 'High'
        tenure.append(value)
```

In [11]:
```
figure(figsize = (12, 5))
sns.histplot(data, x = 'EstimatedSalary')
```

Out[11]:
```
<AxesSubplot:xlabel='EstimatedSalary', ylabel='Count'>
```



In [12]:
```
data['EstimatedSalary'].quantile([0.25, 0.75])
```

Out[12]:
```
0.25     51002.1100
0.75    149388.2475
Name: EstimatedSalary, dtype: float64
```

In [13]:
```
sal_class = []
for value in data['EstimatedSalary']:
    if(value <= 50000):
        value = 'Low'
        sal_class.append(value)
    elif((value > 50000) and (value <= 150000)):
        value = 'Moderate'
        sal_class.append(value)
    else:
        value = 'High'
        sal_class.append(value)
```

In [14]:
```
tenure = pd.Series(tenure)
```

In [15]:
```
sal_class = pd.Series(sal_class)
```

In [16]:
```
data = pd.concat([data, sal_class.rename('sal_class')], axis = 1)
```

In [17]:
```
data = pd.concat([data, tenure.rename('tenure_class')], axis = 1)
```

In [18]:
```
data['Est_sal_ratio'] = data['EstimatedSalary']/data['EstimatedSalary'].media
```

In [19]:
```
data
```

Out[19]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	1	15634602	Hargrave	619	France	Female	42	2
1	2	15647311	Hill	608	Spain	Female	41	1
2	3	15619304	Onio	502	France	Female	42	8
3	4	15701354	Boni	699	France	Female	39	1
4	5	15737888	Mitchell	850	Spain	Female	43	2
...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5
9996	9997	15569892	Johnstone	516	France	Male	35	10

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
9997	9998	15584532	Liu	709	France	Female	36	7
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3
9999	10000	15628319	Walker	792	France	Female	28	4

10000 rows × 17 columns

In [20]:
```
data['tenure_class'].head(10)
```

Out[20]:
```
0         Low
1         Low
2        High
3         Low
4         Low
5        High
6        High
7    Moderate
8    Moderate
9         Low
Name: tenure_class, dtype: object
```

In [21]:
```
mod = data.iloc[2]
data.iloc[2] = data.iloc[7]
data.iloc[7] = mod
```

In [22]:
```
print(data['tenure_class'].head())
print("-------------------------")
print(data['sal_class'].head())
```

```
0         Low
1         Low
2    Moderate
3         Low
4         Low
Name: tenure_class, dtype: object
-------------------------
0    Moderate
1    Moderate
2    Moderate
3    Moderate
4    Moderate
Name: sal_class, dtype: object
```

In [23]:
```
data.head()
```

Out[23]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balar
0	1	15634602	Hargrave	619	France	Female	42	2	0
1	2	15647311	Hill	608	Spain	Female	41	1	83807
2	8	15656148	Obinna	376	Germany	Female	29	4	115046
3	4	15701354	Boni	699	France	Female	39	1	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510

In [24]:
```
data.iloc[1], data.iloc[7] = data.iloc[7], data.iloc[1]
data.iloc[0], data.iloc[16] = data.iloc[16], data.iloc[0]
```

```
data.iloc[2], data.iloc[5] = data.iloc[5], data.iloc[2]
```

In [25]:

In [26]:
```
data.head()
```

Out[26]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	17	15737452	Romeo	653	Germany	Male	58	1	132602
1	3	15619304	Onio	502	France	Female	42	8	159660
2	6	15574012	Chu	645	Spain	Male	44	8	113755
3	4	15701354	Boni	699	France	Female	39	1	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510

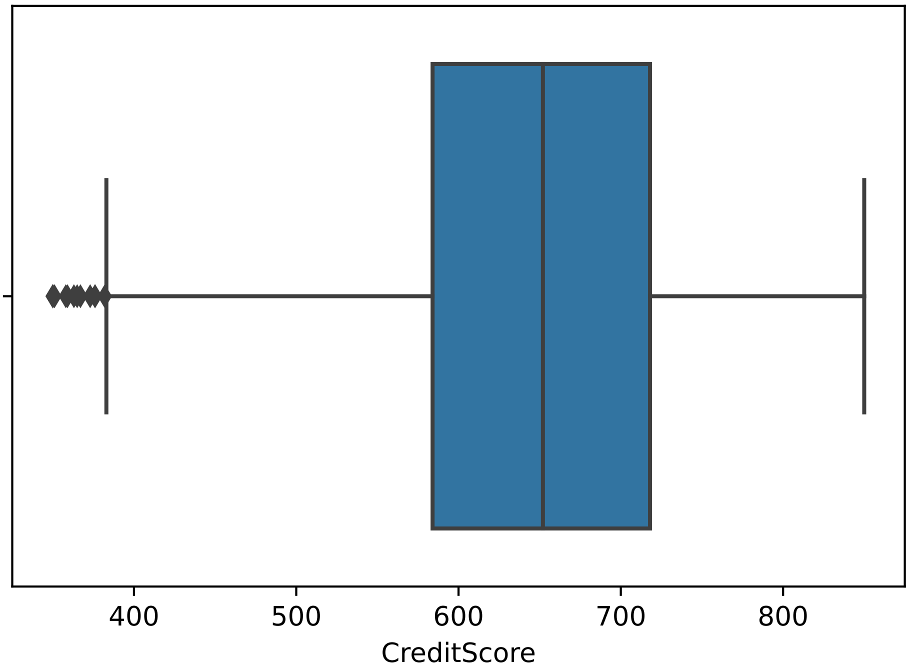
In [27]:
```
# Checking for null values:
for col in data.columns:
    print(f'{col}: {(data[col].loc[data[col].isnull() == True].shape[0])/data
```

```
RowNumber: 0.0%
CustomerId: 0.0%
Surname: 0.0%
CreditScore: 0.0%
Geography: 0.0%
Gender: 0.0%
Age: 0.0%
Tenure: 0.0%
Balance: 0.0%
NumOfProducts: 0.0%
HasCrCard: 0.0%
IsActiveMember: 0.0%
EstimatedSalary: 0.0%
Exited: 0.0%
sal_class: 0.0%
tenure_class: 0.0%
Est_sal_ratio: 0.0%
```

In [28]:
```
data_copy = pd.read_csv('Churn_Modelling.csv')
```

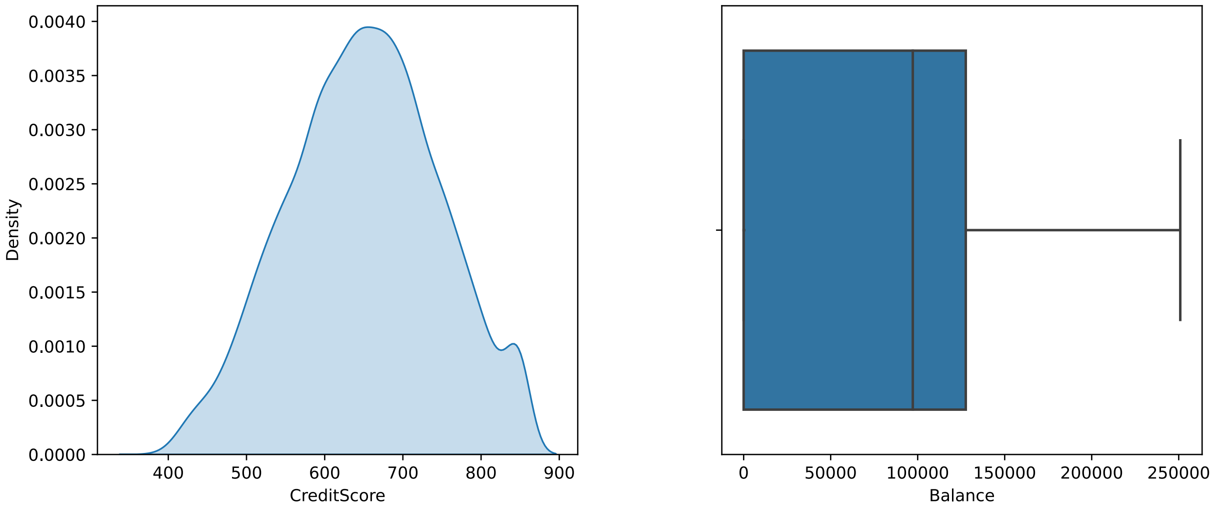
In [29]:
```
sns.boxplot(data = data_copy, x = 'CreditScore')
```

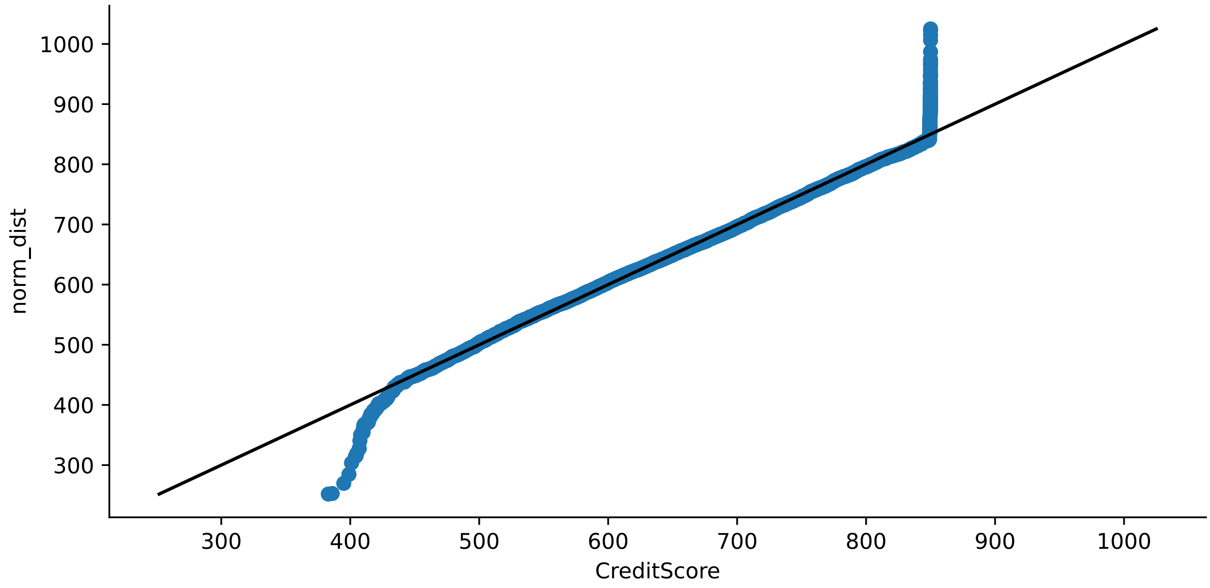
Out[29]: <AxesSubplot:xlabel='CreditScore'>



In [45]:
```
from seaborn_qqplot import pplot
from scipy.stats import norm
fig, axes_1 = plt.subplots(1,2, figsize = (12, 5))
plt.subplots_adjust(wspace=0.30, hspace=0.50)
sns.kdeplot(data = data_copy, x = 'CreditScore', fill = True, ax = axes_1[0])
sns.boxplot(data = data, x = 'Balance', ax = axes_1[1])
pplot(data = data_copy, x = 'CreditScore' , y = norm,kind = 'qq', height = 4,
```

Out[45]:
```
<seaborn.axisgrid.PairGrid at 0x7fc3ee485a30>
```





In [31]:
```
data_copy.head()
```

Out[31]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1	15634602	Hargrave	619	France	Female	42	2	(
1	2	15647311	Hill	608	Spain	Female	41	1	83807
2	3	15619304	Onio	502	France	Female	42	8	15966(
3	4	15701354	Boni	699	France	Female	39	1	(
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551(

In [32]:
```
data_copy = data_copy.drop(data_copy.columns[:3], axis = 1)
data_copy.head()
```

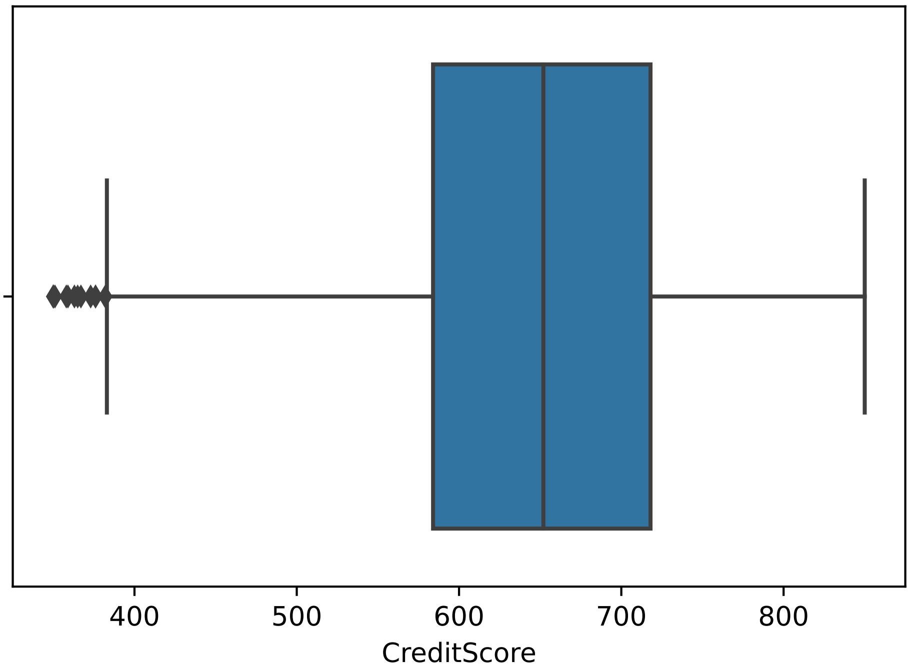
Out[32]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	Is/
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	

In [33]:
```
iqr = data_copy['CreditScore'].quantile(0.75) - data_copy['CreditScore'].quan
cred_outliers = data_copy['CreditScore'][data_copy['CreditScore'] < data_copy
cred_no_out = data_copy['CreditScore'][data_copy['CreditScore'] < 385]
```

In [34]:
```
sns.boxplot(data = data_copy, x = 'CreditScore')
```

Out[34]: <AxesSubplot:xlabel='CreditScore'>



In [35]:

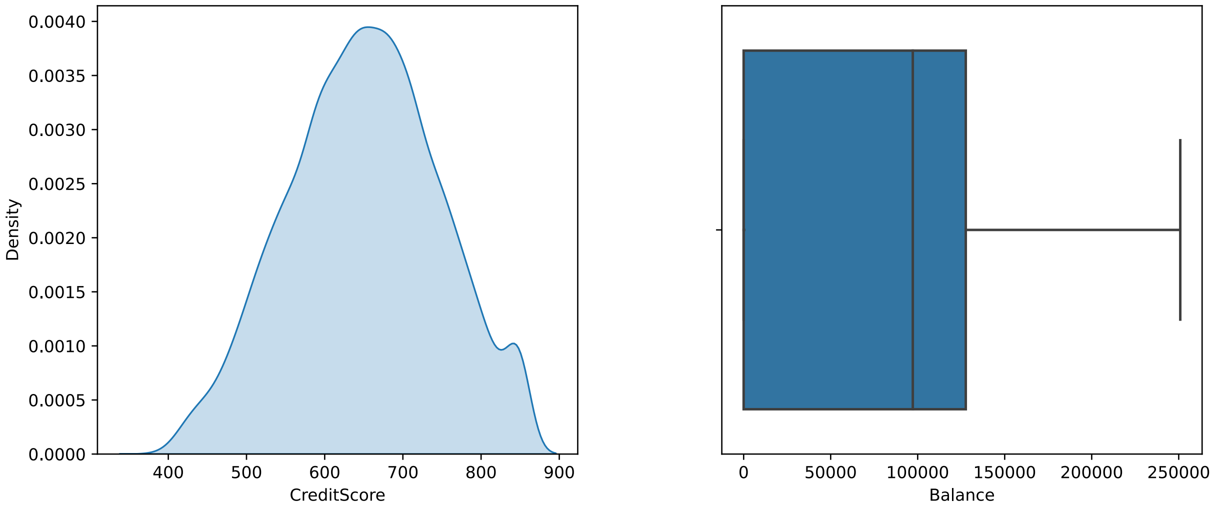
```
data_copy = data_copy.drop(cred_outliers.index, axis = 0)
```

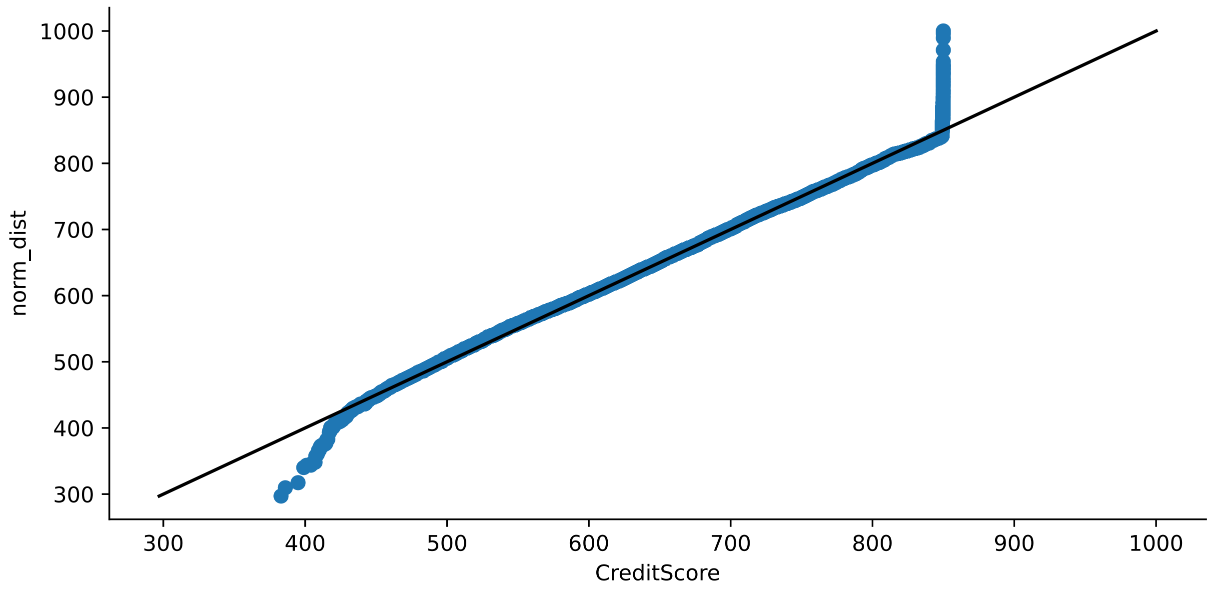
In [36]:

```
from seaborn_qqplot import pplot
from scipy.stats import norm
fig, axes_1 = plt.subplots(1,2, figsize = (12, 5))
plt.subplots_adjust(wspace=0.30, hspace=0.50)
sns.kdeplot(data = data_copy, x = 'CreditScore', fill = True, ax = axes_1[0])
sns.boxplot(data = data, x = 'Balance', ax = axes_1[1])
pplot(data = data_copy, x = 'CreditScore' , y = norm,kind = 'qq', height = 4,
```

Out[36]:

```
<seaborn.axisgrid.PairGrid at 0x7fc3ec7b53a0>
```

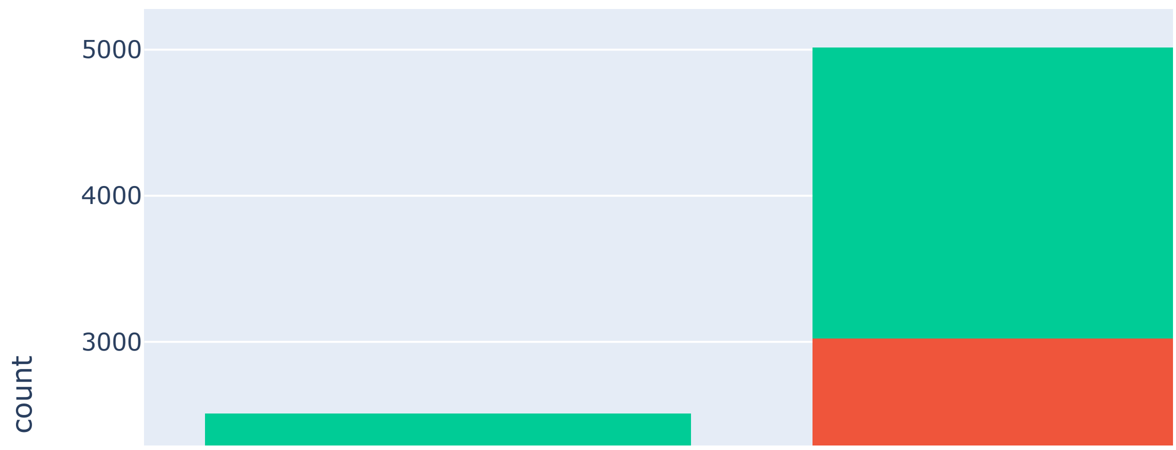




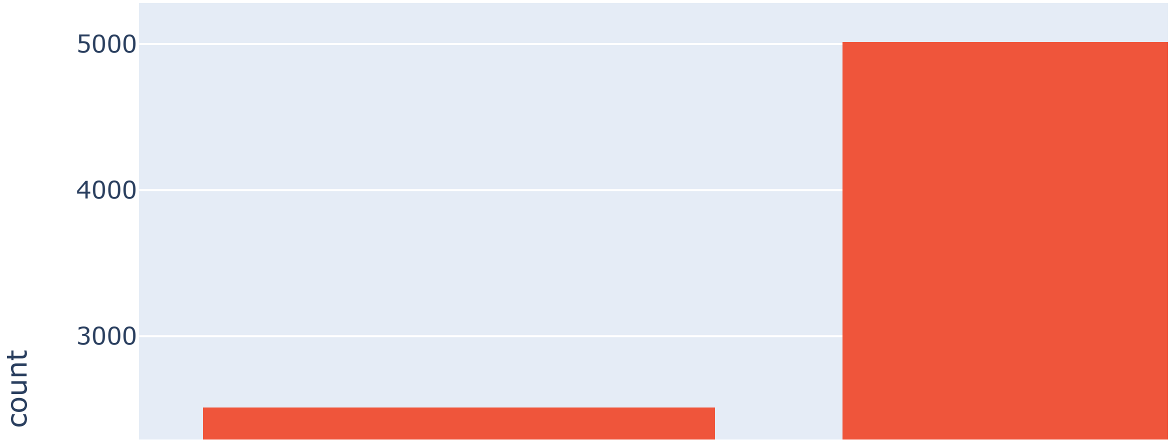
Analysis

For the analysis we first see that we have data from three different nations: Germany, Spain, France.
Due to the cultural differences, it makes sense that we make conclusions for each nation instead of making any conclusion on the whole dataset without taking into the account the geography

In [52]:
```
import plotly.express as px
import plotly.offline as pyo
pyo.init_notebook_mode()
px.histogram(data, x = data['Geography'], color = data['tenure_class'])
```

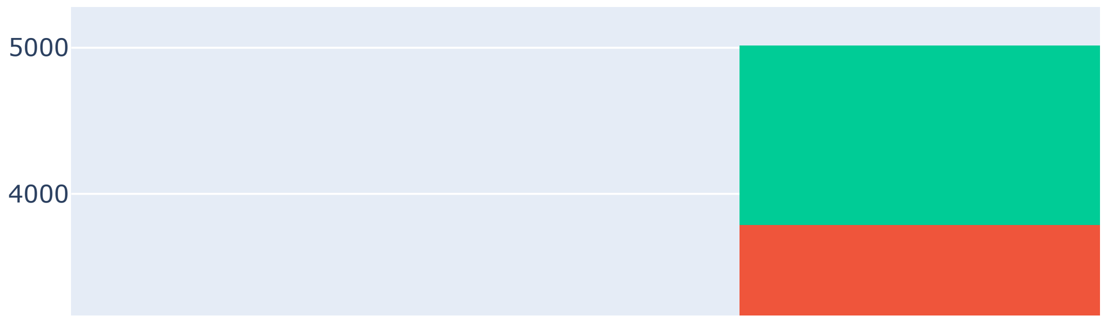


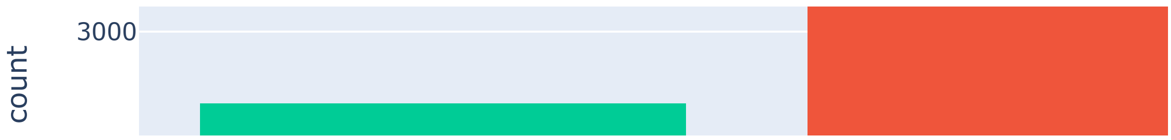
In [53]:
```
import plotly.express as px
px.histogram(data, x = data['Geography'], color = data['Exited'] )
```



1. Out of the total churned customers, 32% German and 16% French Customers have churned, and nearly 17% Spanish Customers have churned.
   So German customers don't seem to be satisfied with the services

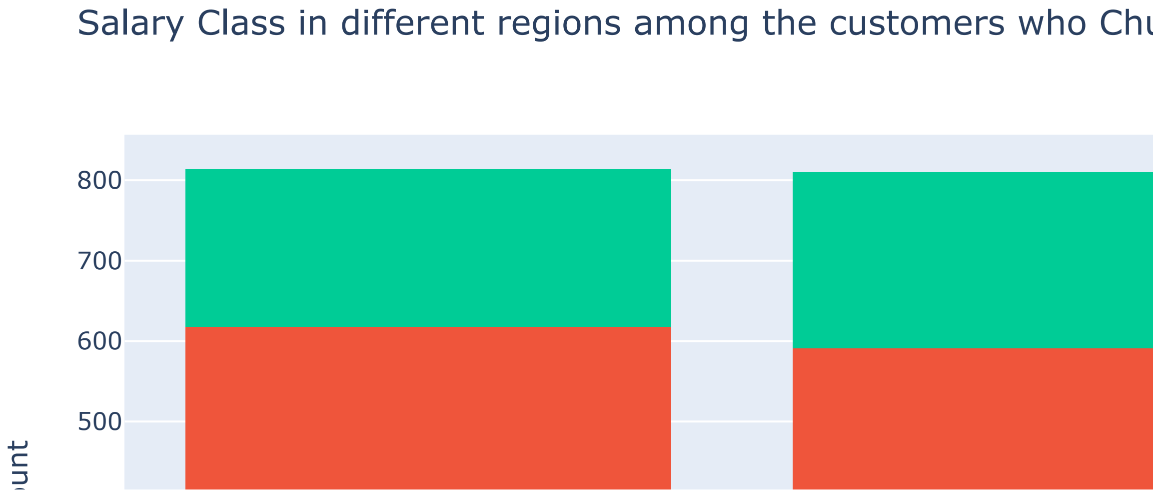
In [54]:
```
px.histogram(data, x = data['Geography'], color = data['sal_class'] )
```





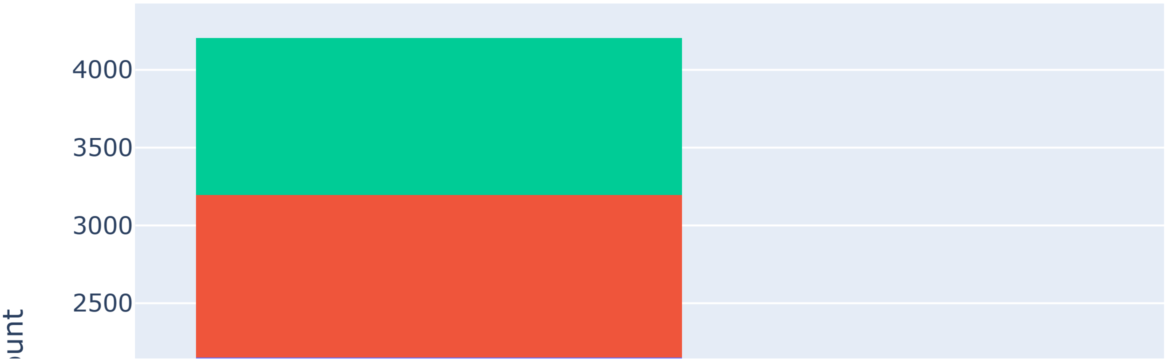
In [40]:
```
temp_1 = data[data['Exited'] == 1]
temp_2 = data[data['Exited'] == 0]
```

In [55]:
```
px.histogram(temp_1, x = temp_1['Geography'], color = temp_1['sal_class'], ti
```



In [56]:
```
px.histogram(temp_2, x = temp_2['Geography'], color = temp_2['sal_class'], ti
```

Salary Class in different regions among the customers who not



In [43]:
```
data_ger = data.loc[data['Geography'] == 'Germany']
data_fra = data.loc[data['Geography'] == 'France']
data_esp = data.loc[data['Geography'] == 'Spain']

for clas in sal_class.unique():
    print(f"In Germany {clas}: {data_ger.loc[(data_ger['sal_class'] == clas)
    print(f"In France {clas}: {data_fra.loc[(data_fra['sal_class'] == clas) &
    print(f"In Spain {clas}: {data_esp.loc[(data_esp['sal_class'] == clas) &
    print('-'*50)
```

```
In Germany Moderate: 31.687898089171973
In France Moderate: 16.140350877192983
In Spain Moderate: 16.443745082612114
--------------------------------------------------
In Germany Low: 35.714285714285715
In France Low: 14.48445171849427
In Spain Low: 14.959349593495935
--------------------------------------------------
In Germany High: 30.76923076923077
In France High: 17.84841075794621
In Spain High: 18.95093062605753
--------------------------------------------------
```

1. In France and Spain, the customer churn distributed by Salary Class is less than 20% of their respective class, but in Germany it is greater than 30%, high proportion of churning among the high Salary class customers show prices of company products may not be the only reason for the Churn

In [44]:
```
data_ger_cred = data.drop(cred_outliers.index, axis = 0)
data_ger_cred = data_ger_cred.loc[data_ger_cred['Geography'] == 'Germany']

data_fra_cred = data.drop(cred_outliers.index, axis = 0)
data_fra_cred = data_fra_cred.loc[data_fra_cred['Geography'] == 'France']
```

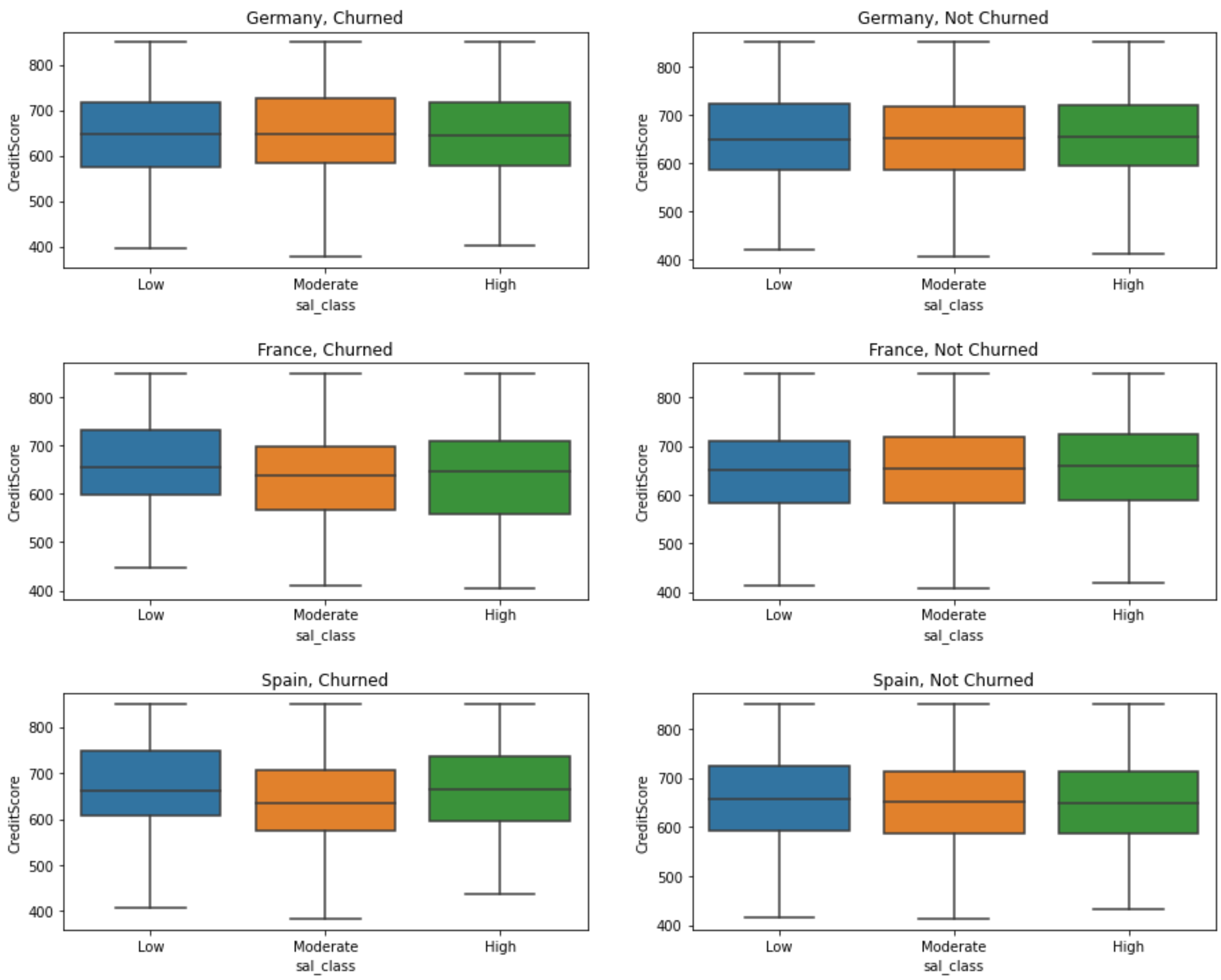
```
data_esp_cred = data.drop(cred_outliers.index, axis = 0)
data_esp_cred = data_esp_cred.loc[data_esp_cred['Geography'] == 'Spain']
```

In [40]:

```
fig, axes = plt.subplots(3,2, figsize = (15,12))
sns.boxplot(data = data_ger_cred.loc[data_ger_cred['Exited'] == 1], x = 'sal_
sns.boxplot(data = data_ger_cred.loc[data_ger_cred['Exited'] == 0], x = 'sal_
axes[0][0].title.set_text('Germany, Churned')
axes[0][1].title.set_text('Germany, Not Churned')

sns.boxplot(data = data_fra_cred.loc[data_fra_cred['Exited'] == 1], x = 'sal_
sns.boxplot(data = data_fra_cred.loc[data_fra_cred['Exited'] == 0], x = 'sal_
axes[1][0].title.set_text('France, Churned')
axes[1][1].title.set_text('France, Not Churned')

sns.boxplot(data = data_esp_cred.loc[data_esp_cred['Exited'] == 1], x = 'sal_
sns.boxplot(data = data_esp_cred.loc[data_esp_cred['Exited'] == 0], x = 'sal_
axes[2][0].title.set_text('Spain, Churned')
axes[2][1].title.set_text('Spain, Not Churned')
plt.subplots_adjust(wspace=0.20, hspace=0.40)
```



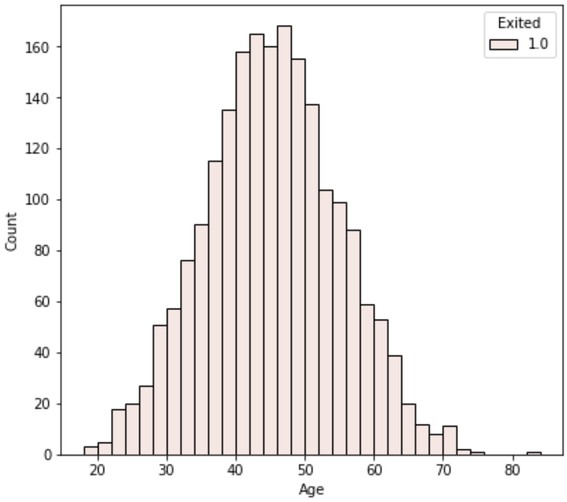
1. Across all the nations, most of the customers have similar proportion of customers with good and bad for all the salary class

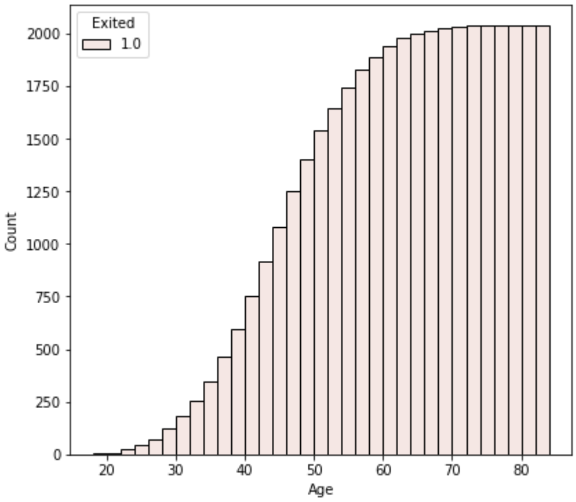
In [41]:

```
exit_0 = data['Exited'][data['Exited'] == 0]
exit_1 = data['Exited'][data['Exited'] == 1]

fig, axes = plt.subplots(1,2, figsize = (15, 6))
sns.histplot(data = data, x = 'Age', hue = exit_1, ax = axes[0])
sns.histplot(data, x = 'Age', hue = exit_1, ax = axes[1], cumulative = True)
plt.subplots_adjust(wspace=0.30, hspace=0.30)
```

```
# Exited vs Estimated Salary
# CreditScore vs Balance
```



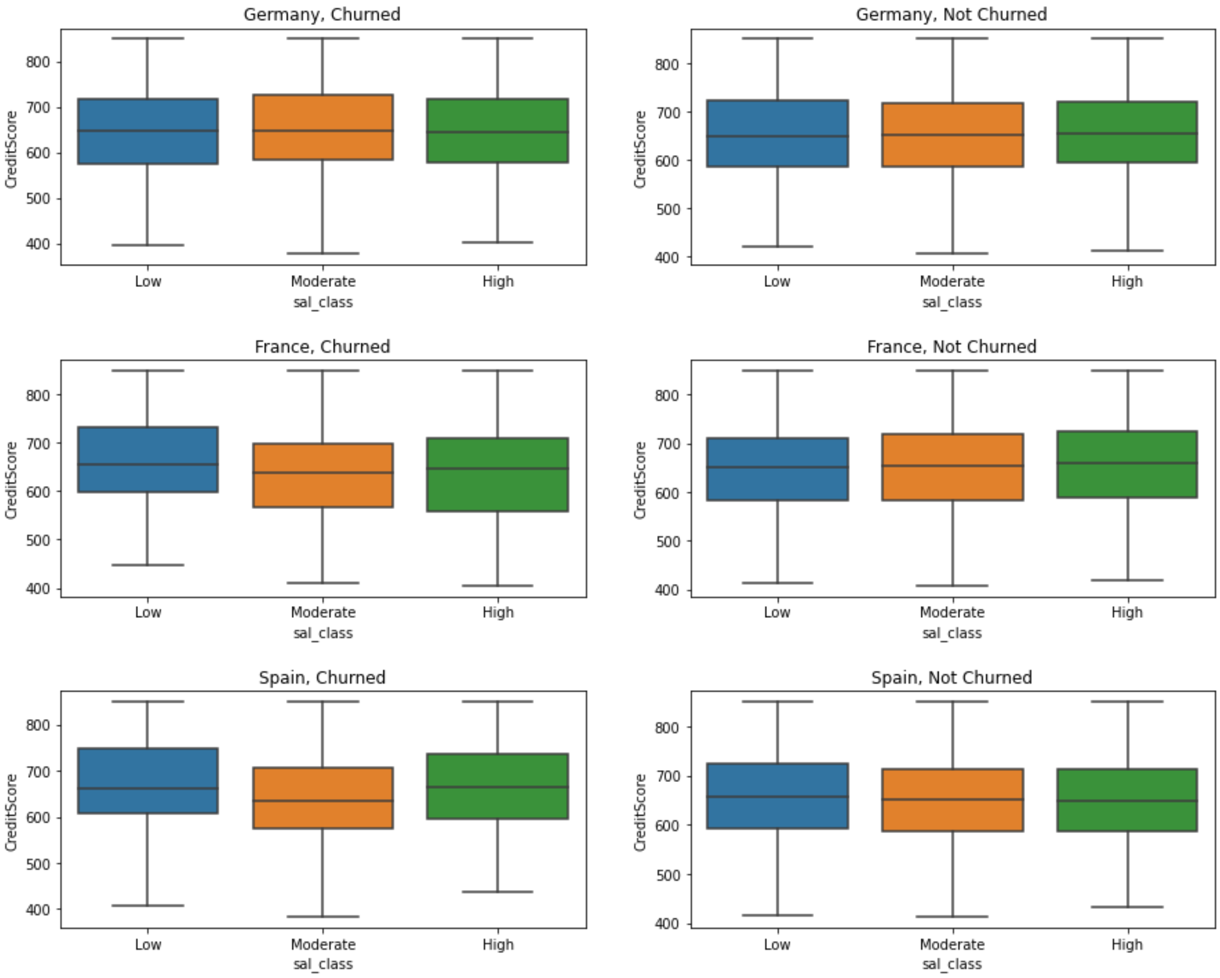


1. We see that customers from the age group 40 - 50 constitute 43% of the total number of churns.
   So we can create provide some services directed at these age groups so they don't leave.

In [42]:
```
fig, axes = plt.subplots(3,2, figsize = (15,12))
sns.boxplot(data = data_ger_cred.loc[data_ger_cred['Exited'] == 1], x = 'sal_
sns.boxplot(data = data_ger_cred.loc[data_ger_cred['Exited'] == 0], x = 'sal_
axes[0][0].title.set_text('Germany, Churned')
axes[0][1].title.set_text('Germany, Not Churned')

sns.boxplot(data = data_fra_cred.loc[data_fra_cred['Exited'] == 1], x = 'sal_
sns.boxplot(data = data_fra_cred.loc[data_fra_cred['Exited'] == 0], x = 'sal_
axes[1][0].title.set_text('France, Churned')
axes[1][1].title.set_text('France, Not Churned')

sns.boxplot(data = data_esp_cred.loc[data_esp_cred['Exited'] == 1], x = 'sal_
sns.boxplot(data = data_esp_cred.loc[data_esp_cred['Exited'] == 0], x = 'sal_
axes[2][0].title.set_text('Spain, Churned')
axes[2][1].title.set_text('Spain, Not Churned')
plt.subplots_adjust(wspace=0.20, hspace=0.40)
```



1. In all the countries we see that out of those customers that churned, most of them belong to the age group 40 - 50 years, so the services of the bank might not be attractive to those age group, on the other hand customers who stayed mostly belong to the age group 20 - 40

In [43]:
```
data.columns
```

Out[43]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited', 'sal_class',
       'tenure_class', 'Est_sal_ratio'],
      dtype='object')
```

In [44]:
```
temp = data.loc[(data['Age'] > 39) & (data['Age'] < 51)]
(temp.Exited[temp.Exited == 1].count()/data.Exited[data.Exited == 1].shape[0]
```

Out[44]:
```
43.053510063819346
```

In [44]:
```

```

Data Preparation for Models

In [45]:
```
mod_data =  data_copy.drop(['Exited'], axis = 1)
target = data_copy['Exited']
```

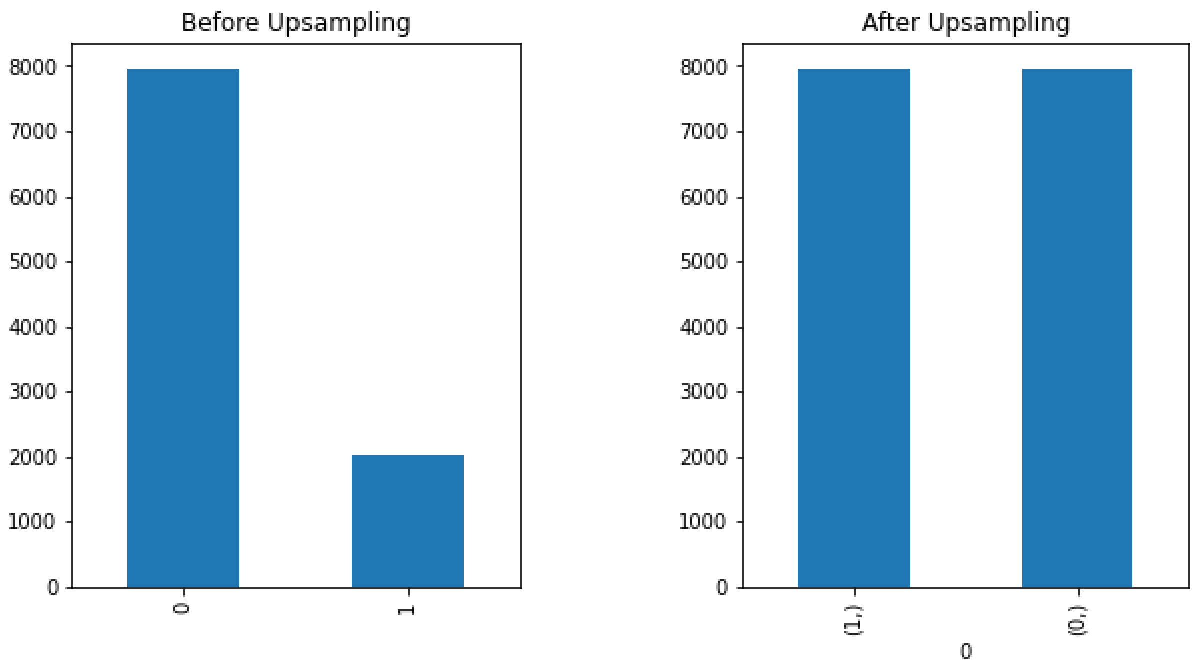
In [46]:
```
pd.Series(data_copy.columns)
```

Out[46]:
```
0          CreditScore
1            Geography
```

```
2                Gender
3                   Age
4                Tenure
5               Balance
6         NumOfProducts
7             HasCrCard
8        IsActiveMember
9       EstimatedSalary
10               Exited
dtype: object
```

In [ ]:
```python
cat_indices = [1,2,7,8]
from imblearn.over_sampling import SMOTENC
X = mod_data
y = target
smote_nc = SMOTENC(categorical_features = cat_indices, random_state = 0)
X_resampled, y_resampled = smote_nc.fit_resample(X, y)
```

In [48]:
```python
fig_up, ax_up = plt.subplots(1, 2, figsize = (10,5))
target_count = data_copy['Exited'].value_counts()
target_count_2 = pd.DataFrame(y_resampled).value_counts()
target_count.plot(kind = 'bar', title = 'Before Upsampling', ax = ax_up[0])
target_count_2.plot(kind = 'bar',title = 'After Upsampling', ax = ax_up[1])
plt.subplots_adjust(wspace=0.50, hspace=0.30)
```



In [49]:
```python
X_resampled = pd.DataFrame(X_resampled, columns = mod_data.columns)
y_resampled = pd.DataFrame(y_resampled)
y_resampled.columns = ['Exited']
```

In [50]:
```python
y_resampled
```

Out[50]:

	Exited
0	1
1	0
2	1
3	0
4	0
...	...

	Exited
15921	1
15922	1
15923	1
15924	1
15925	1

15926 rows × 1 columns

In [51]:
```
#OHE Encoding
data_ohe =  pd.get_dummies(X_resampled, columns = ['Gender', 'Geography', 'Is
data_ohe = pd.concat([data_ohe, y_resampled], axis = 1)
```

In [52]:
```
data_ohe.columns[:6]
```

Out[52]:
```
Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
       'EstimatedSalary'],
      dtype='object')
```

In [53]:
```
data_ohe_copy = data_ohe.copy()
```

In [54]:
```
colnames = data_ohe.columns[:6]
features = data_ohe_copy[colnames]
```

In [55]:
```
# Scaling the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features.values)
```

In [56]:
```
data_ohe_copy[colnames] = scaled_features
```

In [57]:
```
data_ohe_copy.head()
```

Out[57]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	Gender_Fe
0	-0.330293	0.080527	-1.099468	-1.339328	-0.833911	0.009501	
1	-0.451617	-0.018455	-1.467972	0.033428	-0.833911	0.203151	
2	-1.620737	0.080527	1.111552	1.275884	2.443461	0.227180	
3	0.552062	-0.216420	-1.467972	-1.339328	0.804775	-0.120634	
4	2.217507	0.179509	-1.099468	0.716514	-0.833911	-0.375679	

In [58]:
```
dep_var = data_ohe_copy.iloc[:, 0:15]
churn = data_ohe_copy['Exited']
```

In [59]:
```
# Creating Train and Test Set
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(dep_var, churn, random_stat
```

Logistic Regression

In [60]:
```
import statsmodels.api as sm
```

```
from sklearn.linear_model import LogisticRegression
log_reg = sm.Logit(y_train, X_train).fit()
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: Futur
eWarning:

pandas.util.testing is deprecated. Use the functions in the public API at pand
as.testing instead.

Optimization terminated successfully.
         Current function value: 0.527328
         Iterations 8
```

In [61]:
```
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 Exited   No. Observations:                11944
Model:                          Logit   Df Residuals:                    11932
Method:                           MLE   Df Model:                           11
Date:                Tue, 10 Aug 2021   Pseudo R-squ.:                  0.2392
Time:                        15:08:22   Log-Likelihood:                -6298.4
converged:                       True   LL-Null:                       -8278.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
=======
                       coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
CreditScore         -0.0733      0.022     -3.343      0.001      -0.116
-0.030
Age                  0.9421      0.025     37.608      0.000       0.893
0.991
Tenure              -0.0374      0.022     -1.709      0.087      -0.080
0.005
Balance              0.0909      0.024      3.716      0.000       0.043
0.139
NumOfProducts       -0.0592      0.022     -2.692      0.007      -0.102
-0.016
EstimatedSalary      0.0274      0.022      1.248      0.212      -0.016
0.070
Gender_Female        0.2404        nan        nan        nan        nan
nan
Gender_Male         -0.4118        nan        nan        nan        nan
nan
Geography_France    -0.1939        nan        nan        nan        nan
nan
Geography_Germany    0.7128        nan        nan        nan        nan
nan
Geography_Spain     -0.6902        nan        nan        nan        nan
nan
IsActiveMember_0     0.6018        nan        nan        nan        nan
nan
IsActiveMember_1    -0.7731        nan        nan        nan        nan
nan
HasCrCard_0         -0.3005        nan        nan        nan        nan
nan
HasCrCard_1          0.1292        nan        nan        nan        nan
nan
==============================================================================
=======
/usr/local/lib/python3.7/dist-packages/statsmodels/base/model.py:1286: Runtime
Warning:

invalid value encountered in sqrt
```

In [62]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
pred_logistic = log_reg.predict(X_val)
prediction_logistic = list(map(round, pred_logistic))

cm = confusion_matrix(y_val, prediction_logistic)
print("Confusion Matrix: \n", cm)
print('Accuracy on test_set: ', accuracy_score(y_val, prediction_logistic)*10
```

```
Confusion Matrix:
 [[1485  482]
 [ 459 1556]]
Accuracy on test_set:  76.36865896534405
```

In [63]:

```
from sklearn.model_selection import cross_val_score
X_cv = dep_var
y_cv = churn
model_cv = LogisticRegression()
cross_val = cross_val_score(model_cv, X_cv, y_cv, scoring='accuracy')
print(cross_val)
print(cross_val.mean()*100)
```

```
[0.70087884 0.7299843  0.77237049 0.76923077 0.76514914]
74.75227077648385
```

Random Forest

In [64]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline

model_rf = RandomForestClassifier(random_state = 1)
model_rf.fit(X_train, y_train)
preds_rf = model_rf.predict(X_val)
cm_rf = confusion_matrix(y_val, preds_rf)
print("Confusion Matrix: \n", cm_rf)
print("Accuracy on validation set: ", accuracy_score(preds_rf, y_val))
```

```
Confusion Matrix:
 [[1775  192]
 [ 278 1737]]
Accuracy on validation set:  0.8819688598694123
```

In [65]:

```
from sklearn.model_selection import cross_val_score
X_cv_rf = dep_var
y_cv_rf = churn
model_cv_rf = RandomForestClassifier()
cross_val = cross_val_score(model_cv_rf, X_cv_rf, y_cv_rf, scoring='accuracy'
print(cross_val)
print(cross_val.mean()*100)
```

```
[0.75737602 0.88414443 0.91302983 0.92747253 0.91742543]
87.98896467177337
```

Conclusions:

1. Out of the total churned customers, 32% German and 16% French Customers have churned, and nearly 17% Spanish Customers have churned.
   So German customers don't seem to be satisfied with the services

2. In France and Spain, the customer churn distributed by Salary Class is less than 20% of their respective class, but in Germany it is greater than 30%, high proportion of churning among the high Salary class customers show prices of company products may not be the only reason for the Churn

3. We see that customers from the age group 40 - 50 constitute 43% of the total number of churns. So we can create provide some services directed at these age groups so they don't leave.

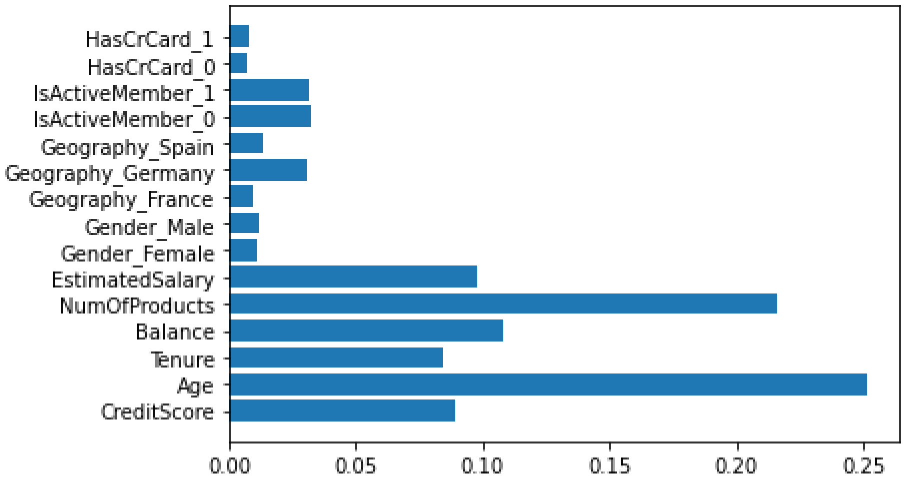
4. In all the countries we see that out of those customers that churned, most of them belong to the age group 40 - 50 years, so the services of the bank might not be attractive to those age group, on the other hand customers who stayed mostly belong to the age group 20 - 40

5. Among those who churned we see that proportion of customers with low credit score is relatively high than the other classes across different salary classes. But among those who didn't also we can see good proportion of customers with lower credit score

Important features

In [66]:
```
plt.barh(X_train.columns, model_rf.feature_importances_)
```

Out[66]:
```
<BarContainer object of 15 artists>
```



In [67]:
```
from sklearn.model_selection import cross_val_score
X_cv_rf = dep_var[['Age', 'NumOfProducts', 'EstimatedSalary', 'CreditScore',
y_cv_rf = churn
model_cv_rf = RandomForestClassifier(random_state=1)
cross_val = cross_val_score(model_cv_rf, X_cv_rf, y_cv_rf, scoring='accuracy'
print(cross_val)
print(cross_val.mean()*100)
```

```
[0.70182046 0.88320251 0.92590267 0.92684458 0.93751962]
87.50579704574861
```

In [68]:
```
X_cv_rf = dep_var[['Age', 'NumOfProducts', 'EstimatedSalary', 'CreditScore',
y_cv_rf = churn
estimates = [100, 500, 1000, 2000]

for n in estimates:
    model_cv_rf = RandomForestClassifier(random_state = 1, n_estimators = n)
    cross_val = cross_val_score(model_cv_rf, X_cv_rf, y_cv_rf, scoring='accur
    print(cross_val)
    print(cross_val.mean()*100)
    print('--------------------\n')
```

```
[0.70182046 0.88320251 0.92590267 0.92684458 0.93751962]
87.50579704574861
--------------------
```

```
[0.69993723 0.88288854 0.92527473 0.92935636 0.93877551]
87.52464717597888
--------------------

[0.69962335 0.88288854 0.9255887  0.93061224 0.9400314 ]
87.57488462573208
--------------------

[0.69930948 0.8844584  0.9255887  0.93092622 0.9400314 ]
87.60628377093269
--------------------
```

In [69]:

```
model_pi = RandomForestClassifier(random_state=1, n_estimators = 1000)
model_pi = model_pi.fit(X_train[['Age', 'NumOfProducts', 'EstimatedSalary', '
```

In [70]:

```
preds_mpi = model_pi.predict(X_val[['Age', 'NumOfProducts', 'EstimatedSalary'
print("Accuracy on validation set: ", accuracy_score(preds_mpi, y_val))
```

```
Accuracy on validation set:  0.876192867905575
```

In [71]:

```
# Storing the model in a pickle file
pickle.dump(model_pi, open('rfmodel.pkl', 'wb'))
model = pickle.load(open('rfmodel.pkl', 'rb'))
```

In [71]:

In [71]: