12.2 Requirement: Course Project: Milestone 5--Final Project Paper and Presentation

- Course Presentation:
  - Each team should prepare a presentation to describe the results of the analytics project and use of the concepts and methods taught throughout the course. Presentations should be 15-20 minutes in length. The presentation will be recorded and submitted through Blackboard. It is recommended to record audio over your slides.
- Course Final Paper:
  - The final paper should be 8-10 pages double spaced in length not including figures and tables. The executive summary should be no longer than one typewritten page, describing the conclusions of your data analysis to a non-technical audience. It should be intelligible to a person who does not know data mining or machine learning techniques. Suppose you are talking to your boss or to a friend who is not familiar with statistical terminology and data science methods. This can be seen as the executive summary/introduction of your report.
  - The technical report should follow. The technical report should include an intro/background of the problem, methods, results, discussion/conclusion and acknowledgments, references, in that order. Clearly state the problem you have chosen to investigate. List the resources you used to come up with the project and reference all sources you used to complete the project. This section is intended for a technical audience and must be written in a clear organized fashion.
- Margins should be 1-inch top, bottom, left, and right. Use any font that is suitable for a professional paper and use 12-point type. The final paper is due by Saturday at midnight.
- **If you are working independently, post this assignment in your Teams project milestones folder – you will use this folder the remainder of the course for other independent team members to be able to review your work.**

==

Abstract :

In consumer space, businesses to survive, scale, grow needs customers and must adapt to customer's requirements. A common understanding is it costs a lot more to acquire a new customer than retain existing ones. For some industries, like banking, insurance, telecom this cost is typical 2x- 5x multiples. Also, observed is the behavior that increase in customer retention by x% leads to 2x-5x % increase in profits. Likewise for sales initiatives, success rate with selling to an existing customer is 60-70% while the same is 5-20% for a new customer.

Customer retention though is getting friction with the recent actionable nomenclature "customer value" i.e., how much is the customer worth for the business. It becomes a planning and strategy initiative than piecemeal task with respect to customer retention given longer term value implications for the business. For academic scope, it makes a lot of sense to develop a prediction model that predicts customer loyalty, tenure and past-the-fence behavior or attempts to react to the signals from the prediction engine.

Intro/problem background :

Scope of this project is limited to banking industry. Banks need to have a better sense of their customers to retain the competitive advantage – either selectively or at scale. It's about prioritization of efforts in the right areas that makes progressive positive to the business.

- In this exercise, using bank's customer dataset, following are the objectives of the effort:
  - Identify what factors contribute to customer churn
  - Build a prediction model, which will perform the following:
    - Classify is a customer is going to churn or not
    - Based on model performance, chose model assigning probability to the churn, so that the concerned redressal department (ex - customer service, or product channel mgr, etc) can easily prioritize low hanging fruits w.r.t churn

Method :

- First, explore and clean the data i.e. through data wrangling and data preparation efforts, cleanse the dataset is the objective. While data wrangling, I would also layout outlier handling policy as I would like to extrapolate incorrect data to the "best-guess" value for each customer. The "best guess" could be based on similar customer attributes or follow a linear regression logic.
- Understanding relationship amongst the variables. For example,
  - identifying dependent and indepent variables amongst the dataset,
  - distribution of values in numeric columns

    - status of variables based on dependent variables
    - categorical values behavior based on dependent variables
- Feature Engineering : create actionable inference from dataset that we expect/ assume to explain customer retention/ tenacy behavior. For example, some of the features I'm planning to explore w.r.t. customer churn are:

    - customer grouping based on their creditworthiness
    - customer product utilization trends with respect to following dimensions
        - across years/ tenacy
        - income levels
        - region based comparison/ baselining
- Based on identified features, I would prepare a model

    - first training the model with appropriate data samples. The key here is to randomize the dataset, with an intention to make it representation of the population, while simultaneously aligning to feature.
    - apply logistic regression model on the trained model
    - model tuning using XGBoost, Random forest would be subsequent iterations based on desired accuracy outputs

"A Layman/ non-tech Explanation of Method" :

- ***Data Exploration***

    - The dataset consists of Customer spread across multiple dimensions like gender, geo location, income level, credit score, etc

    - Given such contexts like geo-spread, wherein there is a possibility of the dataset being skewed towards any particular dimesnion (say geography as an example) -- we perform data exploration exercise to understand how the dataset looks like. We're probably trying to get some answers like --

        - is this evenly distributed, normalized dataset
        - which are indepdent and dependent variables in the dataset?
        - distribution of values in the numeric columns *are there autliers that need special handling?*
        - how do dependent ~ independent ~ categorical variables behave w.r.t assessment of the impacted variable i.e. if customer churn if the event being observed, then which attribute (like income, age, geo, credit worthiness, etc) of the customer can be "considered" to have impacted the event?
        - how does the customer profiles vary across dimensions
        - who are the customers churning / or about to?
            - if the efforts were to be prioritized ~ sequencing order would be? -
                - prevent exisitng customers from leaving
                - identify patterns/ causal realationship with churn events?
                - predicting who is about to churn and addressing the causal aspect
            - what are some of the low-hanging fruits that can be addressed today to prevent x% customer churn?
    - As part of data exploration, I also wrangle dataset i.e.

        - cleanse data, format columns
        - outlier and missing values - handling policy
            - for outliers and missing values, it is equally acceptable to ignore, *best-guess* or force-fit values or a hybrid/ combnation of as many. either of these trade-off lies with model accuracy and having a subjective element with model accuracy
        - for this exercise, i have adopted "best-guess" method with missing values and inclusive approach with outliers. *There aren't as many outliers in the dataset, so my inclusivity criteria isn't tested as much.*

- ***Feature Engineering and Modeling***

    - Based on data exploration phase, I realize a need to derive customer credit worthiness and tenacy score. As a means to the end, I lean toward using these "features" *i.e. to derive actionable inference*

        - customer grouping based on their credit worthiness
        - customer product utilization trends across dimensions
            - income levels
            - tenure/ tenacy years
            - geo location spread

- In the scope of this academic assignement, objective of the model is to improve with accuracy of predicting if the customer will churn or not. Since cost of loss of customer is expensive , so I'm biased towards adopting this modeling approach
- Using modeling methods, like logistic regression, XGBoost to improve the accuracy of prediction model

Key Observations when ^ methods are executed:

-

It looks like I got lucky with this dataset, since there are no missing values. I can pass with my plans with outlier/ missing value handling for this assignmenti will drop customer specific attributes (customerID, and surname) -- these being personal identifiers will result in profiling of customers -- which is not any of my study scope

In [6]:

```
# Review the top rows of what is left of the data frame
df.head()
```

Out[6]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Few observations at this point: 1. Data looks like a point-in-time snapshot. So following are observations / unknowns: - what date is it extracted? Does this date have any relevance w.r.t seasonality / business cycle? - if we can have identical dataset at different points in time, than just a standalone dataset as in this case, it would have helped smoothen out seasonality aspects 2. There are customers who have exited but still have a balance in their account! What would this mean? Could they have exited from a product and not the bank? 3. We don't have product detail breakdown -- which might have indicated addtional information with respect to mapping of customer to bank's business spread For this exercise, we proceed to model without context even though typically having context and better understanding of the data extraction process would give better insight and possibly lead to better and contextual results of the modelling processMy primary intention is to assess what variables contribute to customer quitting i.e. "Exit" status - 20% of the customers have churned. - So the baseline model could be to predict that 20% of the customers will churn. - Given 20% is a small number, my focus would be on accuracy to as it is of interest to the bank to identify and keep this bunch as opposed to accurately predicting the customers that are retained. - Majority of data is from persons from France. Areas with fewer clients have lesser churn than with more customer population - female customers churn is greater than that of male customers - majority of the customer churn is with credit cards. Also majority of the customers have credit cards is anecdotal fact - inactive members have a greater churn. - overall proportion of inactive members is high suggesting that the bank may need a program implemented to turn this group to active customers as this could positively impact the customer churn.Observe that the salary has little effect on the chance of a customer churning. However, the ratio of the bank balance and the estimated salary indicates that customers with a higher balance salary ratio churn more which would be worrying to the bank as this impacts their source of loan capital.

Discussion :

As you see from the workings above, I have picked as many models so as to identify their best on scores alone. This approach obviously is kind of brute-force method, and can be finetuned further by focus on optimization/ tuning methods.

My main aim is to predict the customers that will possibly churn so they can be put in some sort of scheme to prevent churn hence the recall measures on the 1's is of more importance to me than the overall accuracy score of the model. Given that in the data we only had 20% of churn, a recall greater than this baseline will already be an improvement but we want to get as high as possible while trying to maintain a high precision so that the bank can train its resources effectively towards clients highlighted by the model without wasting too much resources on the false positives.

From the review of the fitted models above, the best model that gives a decent balance of the recall and precision is the random forest where according to the fit on the training set, with a precision score on 1's of 0.88, out of all customers that the model thinks will churn, 88% do actually churn and with the recall score of 0.53 on the 1's, the model is able to highlight 53% of all those who churned.

Conclusion :

The precision of the model on previousy unseen test data is slightly higher with regard to predicting 1's i.e. those customers that churn.

However, in as much as the model has a high accuracy, it still misses about half of those who end up churning. This could be imprved by providing retraining the model with more data over time while in the meantime working with the model to save the 41% that would have churned :-)

References :

- https://medium.com/@noah.fintech/creating-a-banking-customer-churn-model-1a2d0850f071
- https://www.tigeranalytics.com/blog/addressing-customer-churn-in-banking/
- https://www.qualtrics.com/blog/customer-churn-banking/

- https://www.kaggle.com/mathchi/churn-problem-for-bank-customer

P.S. Please do refer enclosed summary presentation document, as we get into code-working details.

Code Workings :

In [28]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [1]:
```
# For data wrangling
import numpy as np
import pandas as pd

# For visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
pd.options.display.max_rows = None
pd.options.display.max_columns = None
```

In [2]:
```
# Read the data frame
df = pd.read_csv('Churn_Modelling.csv', delimiter=',')
df.shape
```

Out[2]:
```
(10000, 14)
```

we dont know what columns are necessary and the manipulations needed before data exploration and prediction modeling

In [3]:
```
# Check columns list and missing values
df.isnull().sum()
```

Out[3]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

It looks like I got lucky with this dataset, since there are no missing values. I can pass with my plans with outlier/ missing value handling for this assignment

In [4]:
```
# Get unique count for each variable
df.nunique()
```

Out[4]:
```
RowNumber          10000
CustomerId         10000
Surname             2932
CreditScore          460
Geography              3
Gender                 2
Age                   70
Tenure                11
Balance             6382
NumOfProducts          4
HasCrCard              2
IsActiveMember         2
```

```
EstimatedSalary     9999
Exited                 2
dtype: int64
```

i will drop customer specific attributes (customerID, and surname) -- these being personal identifiers will result in profiling of customers -- which is not any of my study scope

In [5]:
```
# Drop the columns as explained above
df = df.drop(["RowNumber", "CustomerId", "Surname"], axis = 1)
```

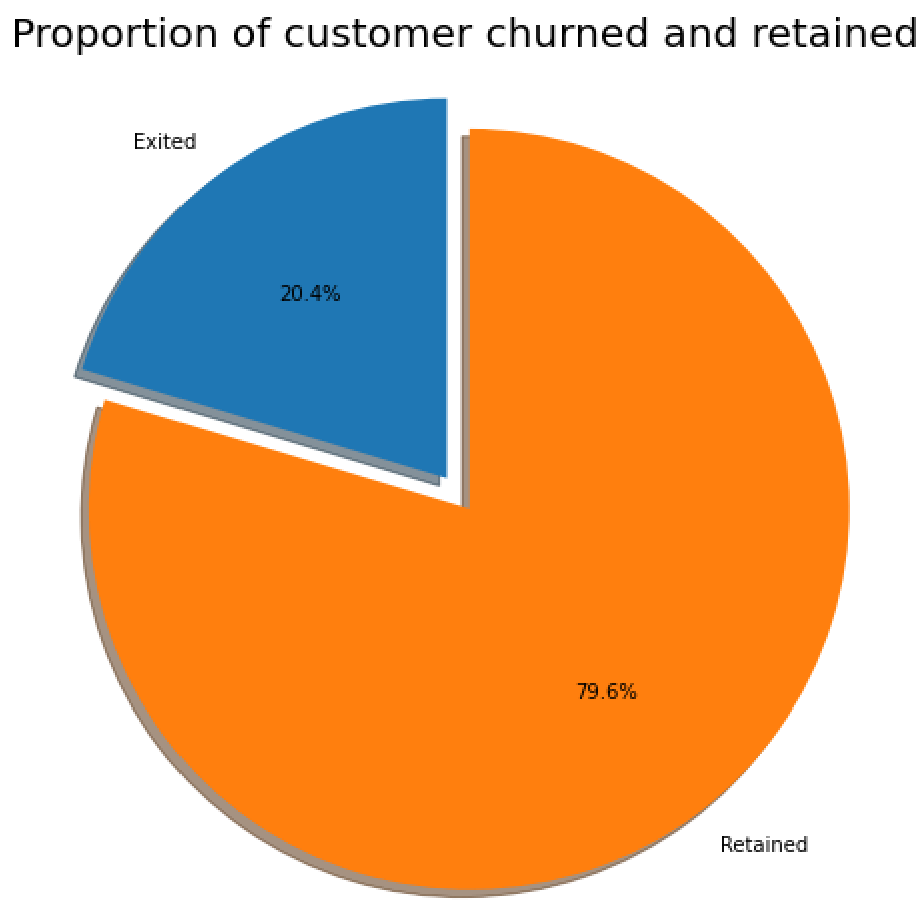
In [6]:
```
# Review the top rows of what is left of the data frame
df.head()
```

Out[6]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Few observations or questions at this point: 1. Data looks like a point-in-time snapshot. So following are observations / unknowns: - what date is it extracted? Does this date have any relevance w.r.t seasonality / business cycle? - if we can have identical dataset at different points in time, than just a standalone dataset as in this case, it would have helped smoothen out seasonality aspects 2. There are customers who have exited but still have a balance in their account! What would this mean? Could they have exited from a product and not the bank? 3. We don't have product detail breakdown -- which might have indicated addtional information with respect to mapping of customer to bank's business spread For this exercise, we proceed to model without context even though typically having context and better understanding of the data extraction process would give better insight and possibly lead to better and contextual results of the modelling processMy primary intention is to assess what variables contribute to customer quitting i.e. "Exit" status

In [7]:
```
labels = 'Exited', 'Retained'
sizes = [df.Exited[df['Exited']==1].count(), df.Exited[df['Exited']==0].count()]
explode = (0, 0.1)
fig1, ax1 = plt.subplots(figsize=(10, 8))
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.title("Proportion of customer churned and retained", size = 20)
plt.show()
```



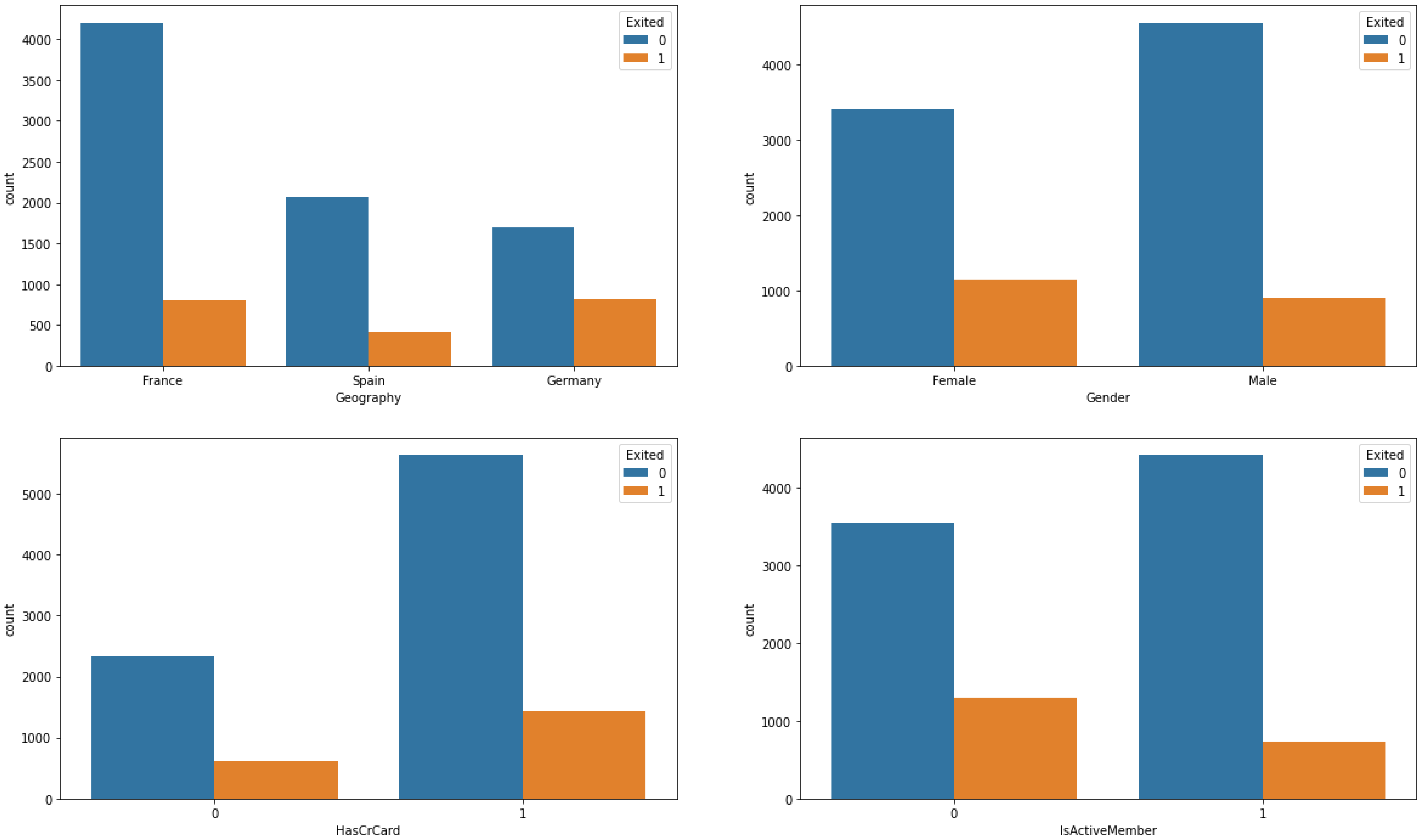
- 20% of the customers have churned. - So the baseline model could be to predict that 20% of the customers will churn. - Given 20% is a small number, my focus would be on accuracy to as it is of interest to the bank to identify and keep this bunch as opposed to accurately predicting the customers that are retained.

In [8]:

```
# We first review the 'Status' relation with categorical variables
fig, axarr = plt.subplots(2, 2, figsize=(20, 12))
sns.countplot(x='Geography', hue = 'Exited',data = df, ax=axarr[0][0])
sns.countplot(x='Gender', hue = 'Exited',data = df, ax=axarr[0][1])
sns.countplot(x='HasCrCard', hue = 'Exited',data = df, ax=axarr[1][0])
sns.countplot(x='IsActiveMember', hue = 'Exited',data = df, ax=axarr[1][1])
```

Out[8]:

```
<AxesSubplot:xlabel='IsActiveMember', ylabel='count'>
```



Following observations: - Majority of data is from persons from France. Areas with fewer clients have lesser churn than with more customer population - female customers churn is greater than that of male customers - majority of the customer churn is with credit cards. Also majority of the customers have credit cards is anecdotal fact - inactive members have a greater churn. - overall proportion of inactive members is high suggesting that the bank may need a program implemented to turn this group to active customers as this could positively impact the customer churn.

Feature Engineering :

In [9]:

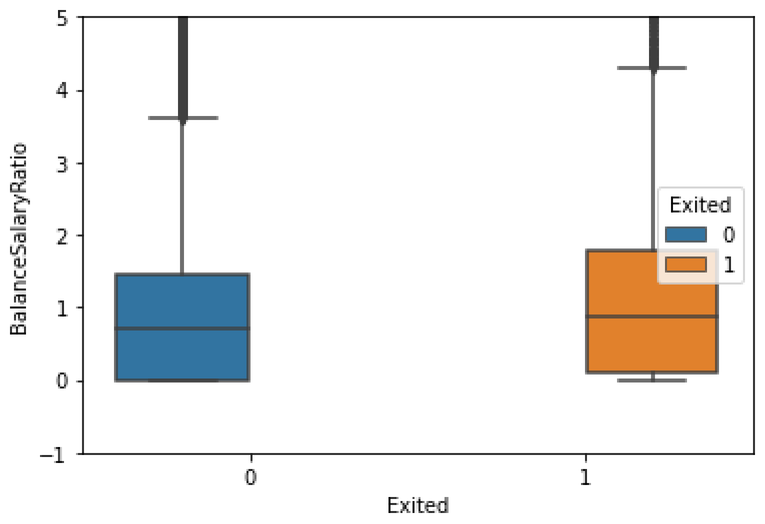
```
# Split Train, test data
df_train = df.sample(frac=0.8,random_state=200)
df_test = df.drop(df_train.index)
print(len(df_train))
print(len(df_test))
```

```
8000
2000
```

In [10]:

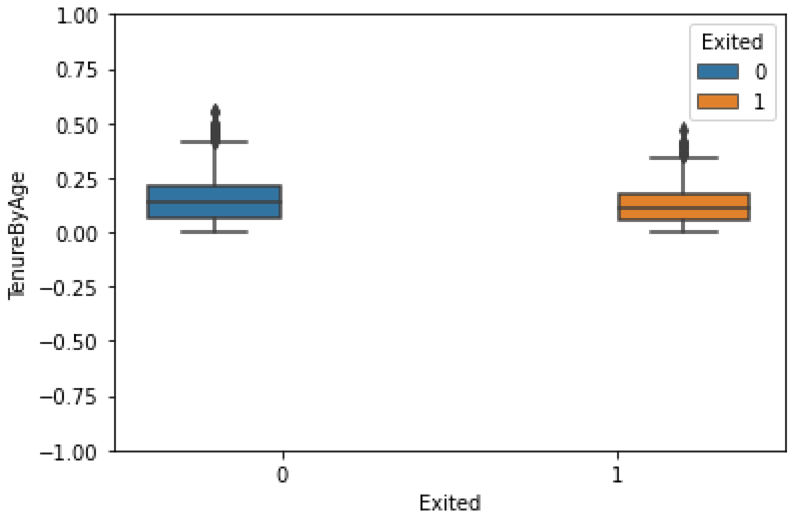
```
df_train['BalanceSalaryRatio'] = df_train.Balance/df_train.EstimatedSalary
sns.boxplot(y='BalanceSalaryRatio',x = 'Exited', hue = 'Exited',data = df_train)
plt.ylim(-1, 5)
```

Out[10]: (-1.0, 5.0)



Observe that the salary has little effect on the chance of a customer churning. However, the ratio of the bank balance and the estimated salary indicates that customers with a higher balance salary ratio churn more which would be worrying to the bank as this impacts their source of loan capital.

In [11]:
```
# Given that tenure is a 'function' of age, we introduce a variable aiming to standardize tenure over age:
df_train['TenureByAge'] = df_train.Tenure/(df_train.Age)
sns.boxplot(y='TenureByAge',x = 'Exited', hue = 'Exited',data = df_train)
plt.ylim(-1, 1)
plt.show()
```



In [12]:
```
'''Lastly we introduce a variable to capture credit score given age to take into account credit behaviour ~ adult life
:-)'''
df_train['CreditScoreGivenAge'] = df_train.CreditScore/(df_train.Age)
```

In [13]:
```
# Resulting Data Frame
df_train.head()
```

Out[13]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	BalanceSalaryRatio	TenureByAge	CreditScoreGivenAge
8159	461	Spain	Female	25	6	0.00	2	1	1	15306.29	0	0.000000	0.240000	18.440000
6332	619	France	Female	35	4	90413.12	1	1	1	20555.21	0	4.398550	0.114286	17.685714
8895	699	France	Female	40	8	122038.34	1	1	0	102085.35	0	1.195454	0.200000	17.475000
5351	558	Germany	Male	41	2	124227.14	1	1	1	111184.67	0	1.117305	0.048780	13.609756
4314	638	France	Male	34	5	133501.36	1	0	1	155643.04	0	0.857741	0.147059	18.764706

Data Preparation for Model fit

In [14]:
```
# Arrange columns by data type for easier manipulation
continuous_vars = ['CreditScore',  'Age', 'Tenure', 'Balance','NumOfProducts', 'EstimatedSalary', 'BalanceSalaryRatio',
                   'TenureByAge','CreditScoreGivenAge']
cat_vars = ['HasCrCard', 'IsActiveMember','Geography', 'Gender']
df_train = df_train[['Exited'] + continuous_vars + cat_vars]
df_train.head()
```

Out[14]:

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreGivenAge	HasCrCard	IsActiveMember	Geography	Gender
8159	0	461	25	6	0.00	2	15306.29	0.000000	0.240000	18.440000	1	1	Spain	Female
6332	0	619	35	4	90413.12	1	20555.21	4.398550	0.114286	17.685714	1	1	France	Female
8895	0	699	40	8	122038.34	1	102085.35	1.195454	0.200000	17.475000	1	0	France	Female
5351	0	558	41	2	124227.14	1	111184.67	1.117305	0.048780	13.609756	1	1	Germany	Male
4314	0	638	34	5	133501.36	1	155643.04	0.857741	0.147059	18.764706	0	1	France	Male

In [15]:
```
'''For the one hot variables, we change 0 to -1 so that the models can capture a negative relation
where the attribute in inapplicable instead of 0'''
df_train.loc[df_train.HasCrCard == 0, 'HasCrCard'] = -1
df_train.loc[df_train.IsActiveMember == 0, 'IsActiveMember'] = -1
df_train.head()
```

Out[15]:

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreGivenAge	HasCrCard	IsActiveMember	Geography	Gender
8159	0	461	25	6	0.00	2	15306.29	0.000000	0.240000	18.440000	1	1	Spain	Female
6332	0	619	35	4	90413.12	1	20555.21	4.398550	0.114286	17.685714	1	1	France	Female
8895	0	699	40	8	122038.34	1	102085.35	1.195454	0.200000	17.475000	1	-1	France	Female
5351	0	558	41	2	124227.14	1	111184.67	1.117305	0.048780	13.609756	1	1	Germany	Male
4314	0	638	34	5	133501.36	1	155643.04	0.857741	0.147059	18.764706	-1	1	France	Male

In [16]:
```
#encode the categorical variables
lst = ['Geography', 'Gender']
remove = list()
for i in lst:
    if (df_train[i].dtype == np.str or df_train[i].dtype == np.object):
        for j in df_train[i].unique():
            df_train[i+'_'+j] = np.where(df_train[i] == j,1,-1)
        remove.append(i)
df_train = df_train.drop(remove, axis=1)
df_train.head()
```

Out[16]:

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreGivenAge	HasCrCard	IsActiveMember	Geography_Spain	Geography_France	Geography_Germa
8159	0	461	25	6	0.00	2	15306.29	0.000000	0.240000	18.440000	1	1	1	-1	
6332	0	619	35	4	90413.12	1	20555.21	4.398550	0.114286	17.685714	1	1	-1	1	
8895	0	699	40	8	122038.34	1	102085.35	1.195454	0.200000	17.475000	1	-1	-1	1	
5351	0	558	41	2	124227.14	1	111184.67	1.117305	0.048780	13.609756	1	1	-1	-1	
4314	0	638	34	5	133501.36	1	155643.04	0.857741	0.147059	18.764706	-1	1	-1	1	

In [17]:
```
# minMax scaling the continuous variables
minVec = df_train[continuous_vars].min().copy()
maxVec = df_train[continuous_vars].max().copy()
df_train[continuous_vars] = (df_train[continuous_vars]-minVec)/(maxVec-minVec)
df_train.head()
```

Out[17]:

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreGivenAge	HasCrCard	IsActiveMember	Geography_Spain	Geography_France	Geography_G
8159	0	0.222	0.094595	0.6	0.000000	0.333333	0.076118	0.000000	0.432000	0.323157	1	1	1	-1	
6332	0	0.538	0.229730	0.4	0.360358	0.000000	0.102376	0.003317	0.205714	0.305211	1	1	-1	1	
8895	0	0.698	0.297297	0.8	0.486406	0.000000	0.510225	0.000901	0.360000	0.300198	1	-1	-1	1	
5351	0	0.416	0.310811	0.2	0.495130	0.000000	0.555744	0.000843	0.087805	0.208238	1	1	-1	-1	
4314	0	0.576	0.216216	0.5	0.532094	0.000000	0.778145	0.000647	0.264706	0.330882	-1	1	-1	1	

In [18]:

```
# data prep pipeline for test data
def DfPrepPipeline(df_predict,df_train_Cols,minVec,maxVec):
    # Add new features
    df_predict['BalanceSalaryRatio'] = df_predict.Balance/df_predict.EstimatedSalary
    df_predict['TenureByAge'] = df_predict.Tenure/(df_predict.Age - 18)
    df_predict['CreditScoreGivenAge'] = df_predict.CreditScore/(df_predict.Age - 18)
    # Reorder the columns
    continuous_vars = ['CreditScore','Age','Tenure','Balance','NumOfProducts','EstimatedSalary','BalanceSalaryRatio',
                       'TenureByAge','CreditScoreGivenAge']
    cat_vars = ['HasCrCard','IsActiveMember',"Geography", "Gender"]
    df_predict = df_predict[['Exited'] + continuous_vars + cat_vars]
    # Change the 0 in categorical variables to -1
    df_predict.loc[df_predict.HasCrCard == 0, 'HasCrCard'] = -1
    df_predict.loc[df_predict.IsActiveMember == 0, 'IsActiveMember'] = -1
    # One hot encode the categorical variables
    lst = ["Geography", "Gender"]
    remove = list()
    for i in lst:
        for j in df_predict[i].unique():
            df_predict[i+'_'+j] = np.where(df_predict[i] == j,1,-1)
        remove.append(i)
    df_predict = df_predict.drop(remove, axis=1)
    # Ensure that all one hot encoded variables that appear in the train data appear in the subsequent data
    L = list(set(df_train_Cols) - set(df_predict.columns))
    for l in L:
        df_predict[str(l)] = -1
    # MinMax scaling coontinuous variables based on min and max from the train data
    df_predict[continuous_vars] = (df_predict[continuous_vars]-minVec)/(maxVec-minVec)
    # Ensure that The variables are ordered in the same way as was ordered in the train set
    df_predict = df_predict[df_train_Cols]
    return df_predict
```

Model fit and Selection

- Logistic regression in the primal space and with different kernels - SVM in the primal and with different Kernels - Ensemble models

In [19]:

```
# Support functions
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from scipy.stats import uniform

# Fit models
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Scoring functions
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

In [20]:
```
# Function to give best model score and parameters
def best_model(model):
    print(model.best_score_)
    print(model.best_params_)
    print(model.best_estimator_)
def get_auc_scores(y_actual, method,method2):
    auc_score = roc_auc_score(y_actual, method);
    fpr_df, tpr_df, _ = roc_curve(y_actual, method2);
    return (auc_score, fpr_df, tpr_df)
```

In [21]:
```
# Fit primal logistic regression
param_grid = {'C': [0.1,0.5,1,10,50,100], 'max_iter': [250], 'fit_intercept':[True],'intercept_scaling':[1],
              'penalty':['l2'], 'tol':[0.00001,0.0001,0.000001]}
log_primal_Grid = GridSearchCV(LogisticRegression(solver='lbfgs'),param_grid, cv=10, refit=True, verbose=0)
log_primal_Grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(log_primal_Grid)
```

```
0.8149999999999998
{'C': 100, 'fit_intercept': True, 'intercept_scaling': 1, 'max_iter': 250, 'penalty': 'l2', 'tol': 1e-05}
LogisticRegression(C=100, max_iter=250, tol=1e-05)
```

In [22]:
```
# Fit logistic regression with degree 2 polynomial kernel
param_grid = {'C': [0.1,10,50], 'max_iter': [300,500], 'fit_intercept':[True],'intercept_scaling':[1],'penalty':['l2'],
              'tol':[0.0001,0.000001]}
poly2 = PolynomialFeatures(degree=2)
df_train_pol2 = poly2.fit_transform(df_train.loc[:, df_train.columns != 'Exited'])
log_pol2_Grid = GridSearchCV(LogisticRegression(solver = 'liblinear'),param_grid, cv=5, refit=True, verbose=0)
log_pol2_Grid.fit(df_train_pol2,df_train.Exited)
best_model(log_pol2_Grid)
```

```
0.8553750000000001
{'C': 50, 'fit_intercept': True, 'intercept_scaling': 1, 'max_iter': 300, 'penalty': 'l2', 'tol': 0.0001}
LogisticRegression(C=50, max_iter=300, solver='liblinear')
```

In [23]:
```
# Fit SVM with RBF Kernel
param_grid = {'C': [0.5,100,150], 'gamma': [0.1,0.01,0.001],'probability':[True],'kernel': ['rbf']}
SVM_grid = GridSearchCV(SVC(), param_grid, cv=3, refit=True, verbose=0)
SVM_grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(SVM_grid)
```

```
0.8518747609662071
{'C': 100, 'gamma': 0.1, 'kernel': 'rbf', 'probability': True}
SVC(C=100, gamma=0.1, probability=True)
```

In [24]:
```
# Fit SVM with pol kernel
param_grid = {'C': [0.5,1,10,50,100], 'gamma': [0.1,0.01,0.001],'probability':[True],'kernel': ['poly'],'degree':[2,3] }
SVM_grid = GridSearchCV(SVC(), param_grid, cv=3, refit=True, verbose=0)
SVM_grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(SVM_grid)
```

```
0.8544999485716948
{'C': 100, 'degree': 2, 'gamma': 0.1, 'kernel': 'poly', 'probability': True}
SVC(C=100, degree=2, gamma=0.1, kernel='poly', probability=True)
```

In [25]:
```
# Fit random forest classifier
param_grid = {'max_depth': [3, 5, 6, 7, 8], 'max_features': [2,4,6,7,8,9],'n_estimators':[50,100],'min_samples_split': [3, 5, 6, 7]}
RanFor_grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, refit=True, verbose=0)
RanFor_grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(RanFor_grid)
```

```
0.8639999999999999
{'max_depth': 8, 'max_features': 9, 'min_samples_split': 5, 'n_estimators': 100}
RandomForestClassifier(max_depth=8, max_features=9, min_samples_split=5)
```

Fit best model

In [37]:
```
# Fit primal logistic regression
log_primal = LogisticRegression(C=100, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, max_iter=250, multi_class='auto',n_jobs=None,
                                penalty='l2', random_state=None, solver='lbfgs',tol=1e-05, verbose=0, warm_start=False)
log_primal.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
```

Out[37]:
```
LogisticRegression(C=100, max_iter=250, tol=1e-05)
```

In [38]:
```
# Fit logistic regression with pol 2 kernel
poly2 = PolynomialFeatures(degree=2)
df_train_pol2 = poly2.fit_transform(df_train.loc[:, df_train.columns != 'Exited'])
log_pol2 = LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, max_iter=300, multi_class='auto', n_jobs=None,
                              penalty='l2', random_state=None, solver='liblinear',tol=0.0001, verbose=0, warm_start=False)
log_pol2.fit(df_train_pol2,df_train.Exited)
```

Out[38]:
```
LogisticRegression(C=10, max_iter=300, solver='liblinear')
```

In [39]:
```
# Fit SVM with RBF Kernel
SVM_RBF = SVC(C=100, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf', max_iter=-1, probability=True,
              random_state=None, shrinking=True,tol=0.001, verbose=False)
SVM_RBF.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
```

Out[39]:
```
SVC(C=100, gamma=0.1, probability=True)
```

In [40]:
```
# Fit SVM with Pol Kernel
SVM_POL = SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,  decision_function_shape='ovr', degree=2, gamma=0.1, kernel='poly',  max_iter=-1,
              probability=True, random_state=None, shrinking=True, tol=0.001, verbose=False)
SVM_POL.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
```

Out[40]:
```
SVC(C=100, degree=2, gamma=0.1, kernel='poly', probability=True)
```

In [41]:
```
# Fit Random Forest classifier
RF = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',max_depth=8, max_features=6, max_leaf_nodes=None,min_impurity_decrease=0.0,
                            min_impurity_split=None,min_samples_leaf=1, min_samples_split=3,min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=None,
                            oob_score=False, random_state=None, verbose=0,warm_start=False)
RF.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
```

Out[41]:
```
RandomForestClassifier(max_depth=8, max_features=6, min_samples_split=3,
                       n_estimators=50)
```

In [42]:
```
# Fit Extreme Gradient Boost Classifier
XGB = XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,colsample_bytree=1, gamma=0.01, learning_rate=0.1, max_delta_step=0,max_depth=7,
                    min_child_weight=5, missing=None, n_estimators=20,n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,reg_alpha=0,
                    reg_lambda=1, scale_pos_weight=1, seed=None, silent=True, subsample=1)
XGB.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
```

```
[10:53:54] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:573:
Parameters: { "silent" } 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 this
  verification. Please open an issue if you find above cases.


[10:53:54] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from
'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[42]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0.01, gpu_id=-1,
```

```
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=7,
              min_child_weight=5, missing=None, monotone_constraints='()',
              n_estimators=20, n_jobs=1, nthread=1, num_parallel_tree=1,
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
              seed=0, silent=True, subsample=1, tree_method='exact',
              validate_parameters=1, verbosity=None)
```

Review best model fit accuracy : Keen interest is on the performance in predicting 1's (Customers who churn)

In [43]:

```python
print(classification_report(df_train.Exited, log_primal.predict(df_train.loc[:, df_train.columns != 'Exited'])))
```

```
              precision    recall  f1-score   support

           0       0.83      0.97      0.89      6353
           1       0.64      0.24      0.35      1647

    accuracy                           0.82      8000
   macro avg       0.73      0.60      0.62      8000
weighted avg       0.79      0.82      0.78      8000
```

In [44]:

```python
print(classification_report(df_train.Exited,  log_pol2.predict(df_train_pol2)))
```

```
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      6353
           1       0.77      0.46      0.57      1647

    accuracy                           0.86      8000
   macro avg       0.82      0.71      0.75      8000
weighted avg       0.85      0.86      0.85      8000
```

In [45]:

```python
print(classification_report(df_train.Exited,  SVM_RBF.predict(df_train.loc[:, df_train.columns != 'Exited'])))
```

```
              precision    recall  f1-score   support

           0       0.86      0.98      0.92      6353
           1       0.85      0.40      0.54      1647

    accuracy                           0.86      8000
   macro avg       0.86      0.69      0.73      8000
weighted avg       0.86      0.86      0.84      8000
```

In [46]:

```python
print(classification_report(df_train.Exited,  SVM_POL.predict(df_train.loc[:, df_train.columns != 'Exited'])))
```

```
              precision    recall  f1-score   support

           0       0.86      0.98      0.92      6353
           1       0.84      0.38      0.52      1647

    accuracy                           0.86      8000
   macro avg       0.85      0.68      0.72      8000
weighted avg       0.85      0.86      0.83      8000
```

In [47]:

```python
print(classification_report(df_train.Exited,  RF.predict(df_train.loc[:, df_train.columns != 'Exited'])))
```

```
              precision    recall  f1-score   support

           0       0.89      0.98      0.93      6353
           1       0.88      0.52      0.65      1647
```

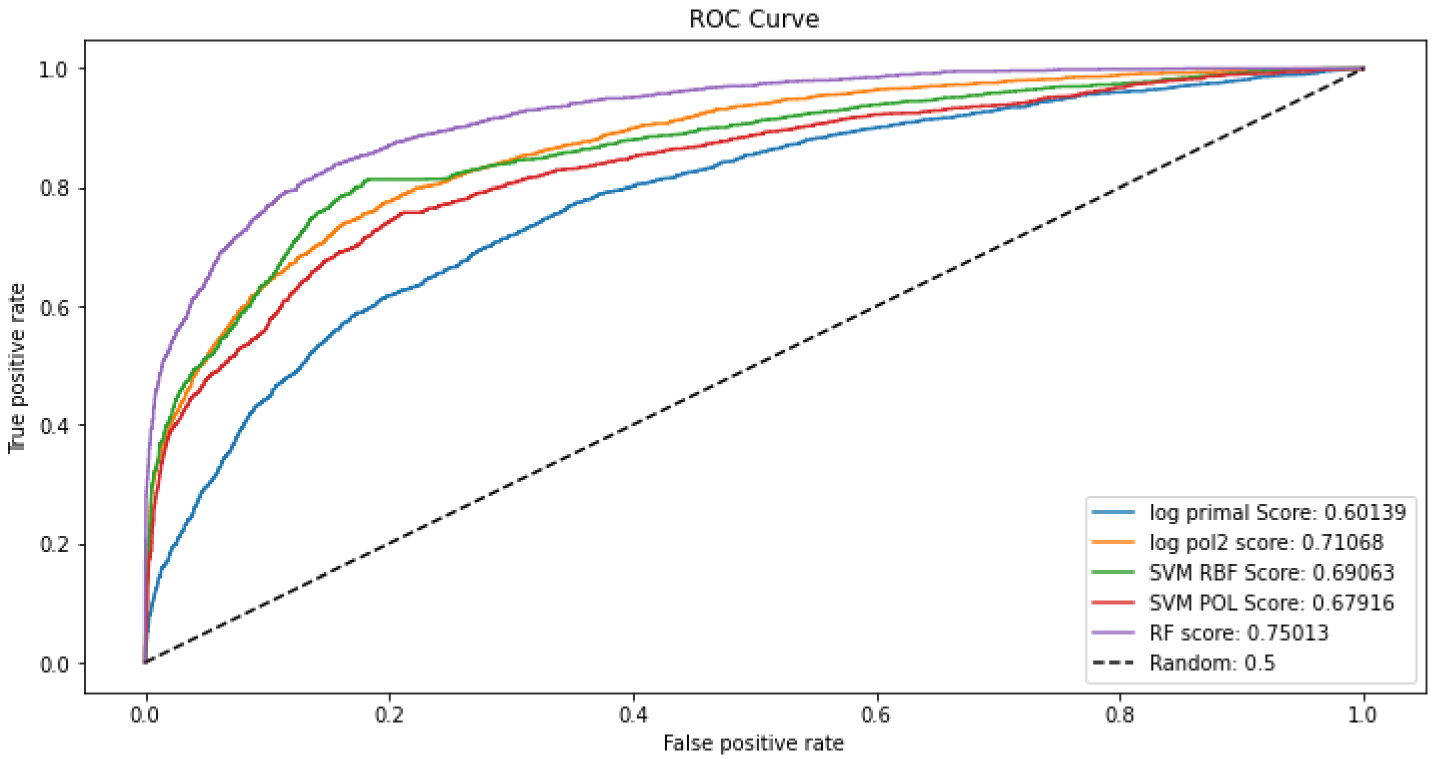
```
    accuracy                           0.89      8000
   macro avg       0.88      0.75      0.79      8000
weighted avg       0.89      0.89      0.87      8000
```

In [50]:

```
y = df_train.Exited
X = df_train.loc[:, df_train.columns != 'Exited']
X_pol2 = df_train_pol2
auc_log_primal, fpr_log_primal, tpr_log_primal = get_auc_scores(y, log_primal.predict(X),log_primal.predict_proba(X)[:,1])
auc_log_pol2, fpr_log_pol2, tpr_log_pol2 = get_auc_scores(y, log_pol2.predict(X_pol2),log_pol2.predict_proba(X_pol2)[:,1])
auc_SVM_RBF, fpr_SVM_RBF, tpr_SVM_RBF = get_auc_scores(y, SVM_RBF.predict(X),SVM_RBF.predict_proba(X)[:,1])
auc_SVM_POL, fpr_SVM_POL, tpr_SVM_POL = get_auc_scores(y, SVM_POL.predict(X),SVM_POL.predict_proba(X)[:,1])
auc_RF, fpr_RF, tpr_RF = get_auc_scores(y, RF.predict(X),RF.predict_proba(X)[:,1])
#auc_XGB, fpr_XGB, tpr_XGB = get_auc_scores(y, XGB.predict(X),XGB.predict_proba(X)[:,1])
```

In [51]:

```
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_log_primal, tpr_log_primal, label = 'log primal Score: ' + str(round(auc_log_primal, 5)))
plt.plot(fpr_log_pol2, tpr_log_pol2, label = 'log pol2 score: ' + str(round(auc_log_pol2, 5)))
plt.plot(fpr_SVM_RBF, tpr_SVM_RBF, label = 'SVM RBF Score: ' + str(round(auc_SVM_RBF, 5)))
plt.plot(fpr_SVM_POL, tpr_SVM_POL, label = 'SVM POL Score: ' + str(round(auc_SVM_POL, 5)))
plt.plot(fpr_RF, tpr_RF, label = 'RF score: ' + str(round(auc_RF, 5)))
#plt.plot(fpr_XGB, tpr_XGB, label = 'XGB score: ' + str(round(auc_XGB, 5)))
plt.plot([0,1], [0,1], 'k--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
#plt.savefig('roc_results_ratios.png')
plt.show()
```



Discussion :

As you see from the workings above, I have picked as many models so as to identify their best on scores alone. This approach obviously is kind of brute-force method, and can be finetuned further by focus on optimization/ tuning methods.

My main aim is to predict the customers that will possibly churn so they can be put in some sort of scheme to prevent churn hence the recall measures on the 1's is of more importance to me than the overall accuracy score of the model. Given that in the data we only had 20% of churn, a recall greater than this baseline will already be an improvement but we want to get as high as possible while trying to maintain a high precision so that the bank can train its resources effectively towards clients highlighted by the model without wasting too much resources on the false positives.

From the review of the fitted models above, the best model that gives a decent balance of the recall and precision is the random forest where according to the fit on the training set, with a precision score on 1's of 0.88, out of all customers that the model thinks will churn, 88% do actually churn and with the recall score of 0.53 on the 1's, the model is able to highlight 53% of all those who churned.

Test Model Accuracy on Test-Data

In [52]:
```
# Make the data transformation for test data
df_test = DfPrepPipeline(df_test,df_train.columns,minVec,maxVec)
df_test = df_test.mask(np.isinf(df_test))
df_test = df_test.dropna()
df_test.shape
```

Out[52]:
```
(1996, 17)
```

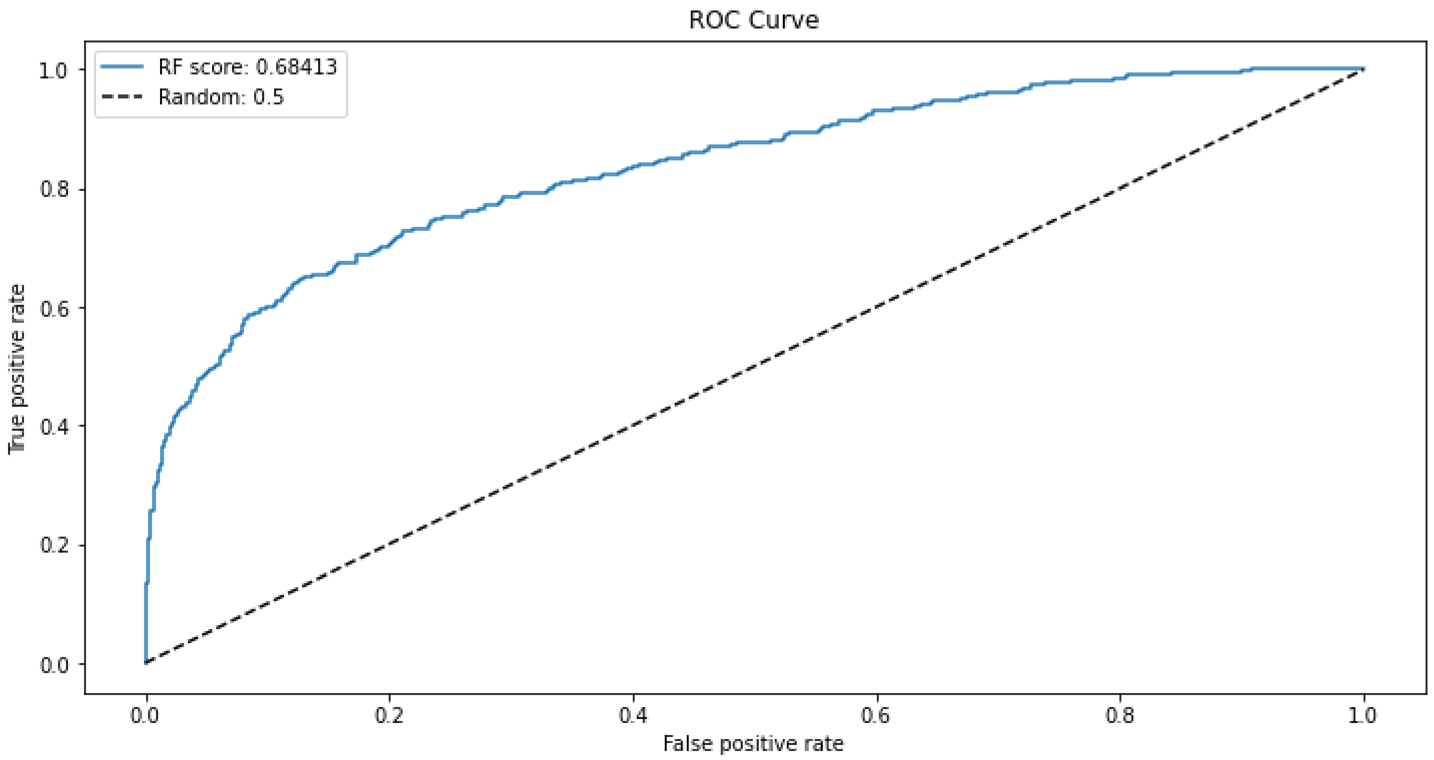
In [53]:
```
print(classification_report(df_test.Exited,  RF.predict(df_test.loc[:, df_test.columns != 'Exited'])))
```

```
              precision    recall  f1-score   support

           0       0.87      0.98      0.92      1607
           1       0.83      0.39      0.53       389

    accuracy                           0.86      1996
   macro avg       0.85      0.68      0.72      1996
weighted avg       0.86      0.86      0.84      1996
```

In [54]:
```
auc_RF_test, fpr_RF_test, tpr_RF_test = get_auc_scores(df_test.Exited, RF.predict(df_test.loc[:, df_test.columns != 'Exited']),
                                                       RF.predict_proba(df_test.loc[:, df_test.columns != 'Exited'])[:,1])
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_RF_test, tpr_RF_test, label = 'RF score: ' + str(round(auc_RF_test, 5)))
plt.plot([0,1], [0,1], 'k--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
#plt.savefig('roc_results_ratios.png')
plt.show()
```



Conclusion :

The precision of the model on previousy unseen test data is slightly higher with regard to predicting 1's i.e. those customers that churn.

However, in as much as the model has a high accuracy, it still misses about half of those who end up churning. This could be imprved by providing retraining the model with more data over time while in the meantime working with the model to save the 41% that would have churned :-)

References :

- https://medium.com/@noah.fintech/creating-a-banking-customer-churn-model-1a2d0850f071
- https://www.tigeranalytics.com/blog/addressing-customer-churn-in-banking/
- https://www.qualtrics.com/blog/customer-churn-banking/
- https://www.kaggle.com/mathchi/churn-problem-for-bank-customer


```
In [ ]:
```