# Credit Card Lead Prediction

## Problem Statement:

Happy Customer Bank wants to cross-sell its credit cards to existing customers. For this they have identified a set of customers that are eligible to take these credit cards. The bank is looking for a help in identifying customers that could show higher intent towards a recommended credit card. The bank uses different kinds of communication like tele-calling, e-mails, recommendations on net banking, mobile banking, etc. But targeting specific customers who are most likely to take the credit cards will save the banks time, cost and effort on advertisement.

The dataset provided has following rows:

| Variable | Definition |
|---|---|
| ID | Unique Identifier for a row |
| Gender | Gender of the Customer |
| Age | Age of the Customer (in Years) |
| Region_Code | Code of the Region for the customers |
| Occupation | Occupation Type for the customer |
| Channel_Code | Acquisition Channel Code for the Customer (Encoded) |
| Vintage | Vintage for the Customer (In Months) |
| Credit_Product | If the Customer has any active credit product (Home loan, Personal loan, Credit Card etc.) |
| Avg_Account_Balance | Average Account Balance for the Customer in last 12 Months |
| Is_Active | If the Customer is Active in last 3 Months |
| Is_Lead(Target) | If the Customer is interested for the Credit Card<br>0 : Customer is not interested<br>1 : Customer is interested |

**Solution:** The problem here is a classification problem, where we are given predict the probability of the target variable. The more the probability, the more is the chance of the customer to take the credit card. The accuracy metrics to be used is the **roc_auc** which the is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the **ROC curve**. The higher the **AUC**, the better the performance of the model at distinguishing between the positive and negative classes.

Approach:

Tech Stack Used:

1. Python, Panda, Matplotlib, Seaborn
2. Scikit-Learn, XgBoost
3. Jupyter Notebook

Data Exploration: During data exploration, below are the key observations:

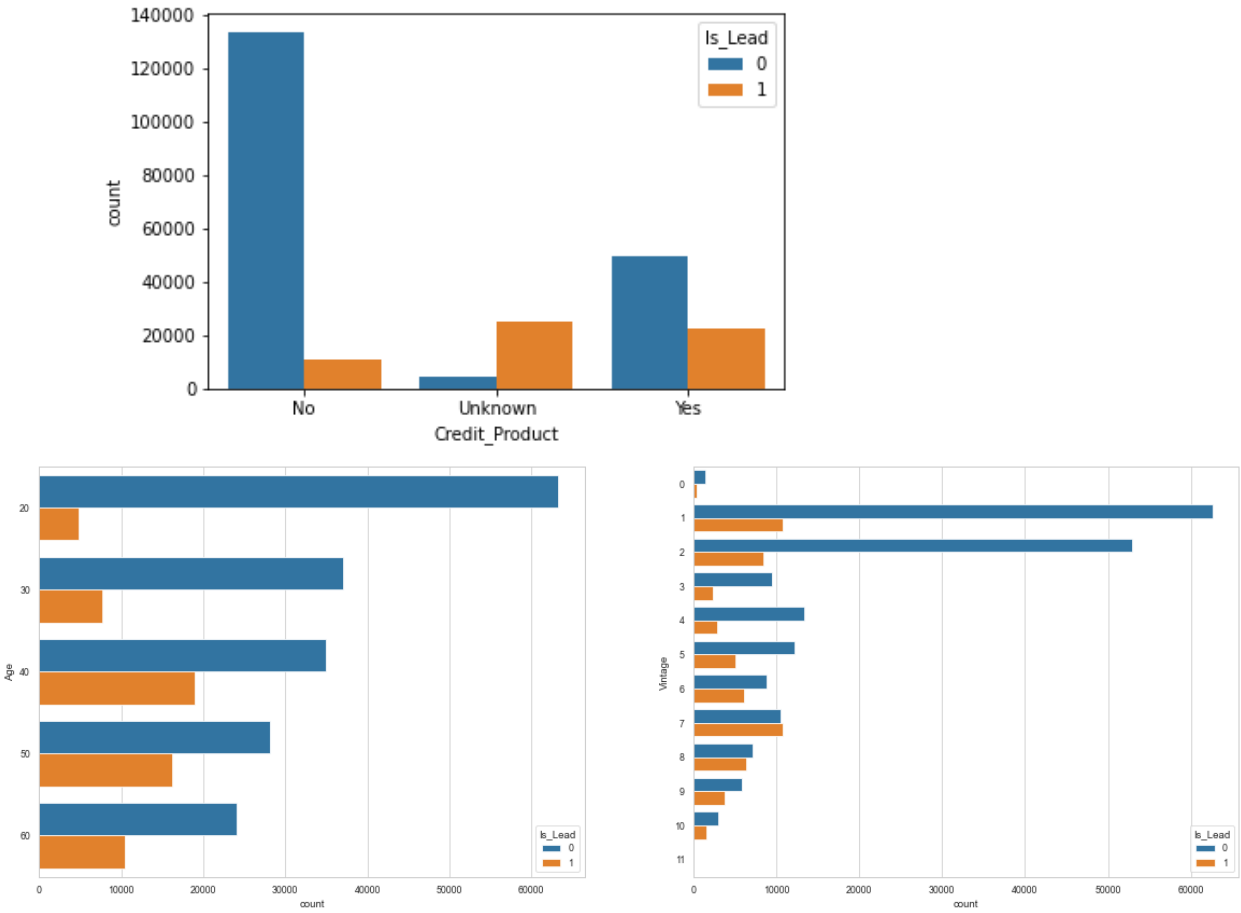
4. There are total of 245725 records and 11 columns.
5. I found out that the target variable is not equally distributes among the total records. Its 76:24 percentage ratio negative to positive class.
6. The ID column specifies is the unique IDs of each customer.
7. The categorical columns present are Gender, Region Code, Credit Product Channel Code, Occupation and Is Active.
8. The continuous columns are Age, Vintage and Avg Account Balance.
9. There were missing values in Credit Product column.
10. There were 35 different categories in the Region Code column

Feature Engineering:

1. I converted the binary columns Gender and Is Active to numeric (0-1) using Label Encoder.
2. Next, I imputed the missing rows in the credit product column with a new category 'Unknown'. Plotting it after imputing showed that this feature could be very important one as the Unknown category has maximum Lead conversion ratio.

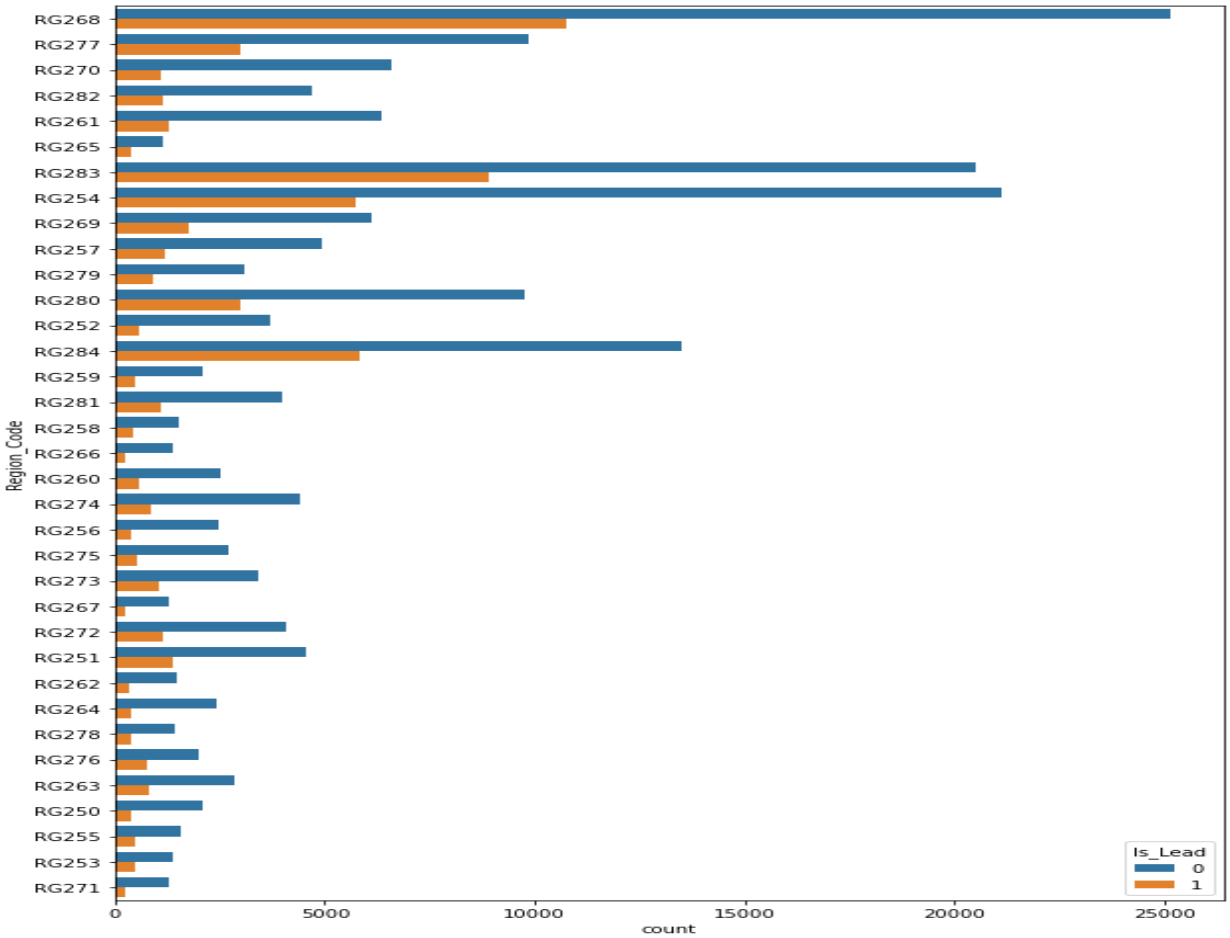
3. Then, I one hot encoded other categorical variables using pandas is get_dummies function. This created ~42 extra features.

4. Coming to numerical features, I converted Vintage and Age in to categorical again using the binning approach.




5. Converted the Avg Account Balance to log10 scale.

6. Region Code was having most categories.




**Modelling:** I started by splitting the train dataset in to 80:20 train test split.

Then created a baseline RandomForest Model.

For this model, I had used Target Encoding for the Region Code column.

To handle the data imbalance, I used ADASYN from imblearn library to oversample the minority class. This increased the model tuning time, but improved the accuracy.

But the Model didn't perform well with this model and gave below accuracy:

```
Train AUC Score: 0.7502576879756989 Test AUC Score: 0.7357228839728609
```

I did some feature engineering experiments and converted the Region Code into One hot encoded column. Upon tuning the model using GridSearchCV, I could get up to below accuracy

```
Test AUC: 0.8678747535664254
```

After this, I switched to XGBoost Model.

**eXtreme Gradient Boosting (XGBoost)** is a scalable and improved version of the gradient boosting algorithm *(terminology alert)* designed for efficacy, computational speed and model performance. It is an open-source library and a part of the Distributed Machine Learning Community.

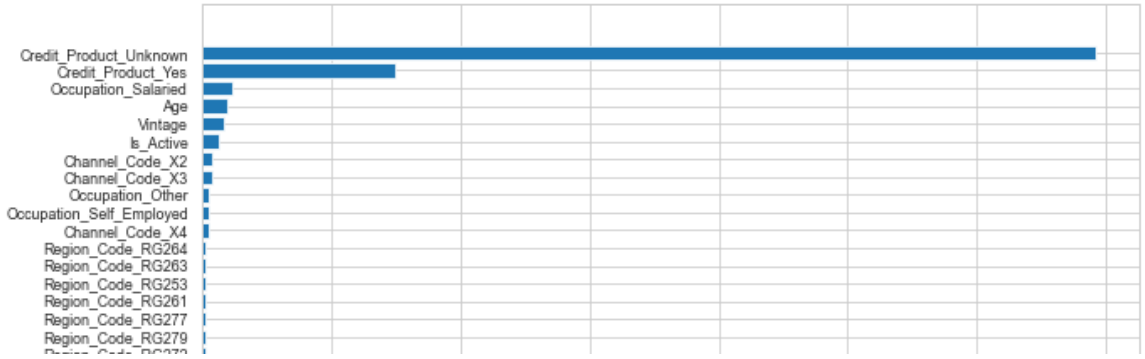
I started with some base parameters

```
param_dist = {'learning_rate' :0.1,
              'n_estimators':200,
              'max_depth':15,
              'min_child_weight':2,
              'objective': 'binary:logistic',
              'scale_pos_weight': scale_post_wt,
              'seed':42}
```

The role of scale_pos_weight is to handle the class imbalance. XGboost does it for us. We must set this parameter which is calculated by the below formula.

*(len(y_train)-sum(y_train))/sum(y_train)*

After further manual tuning and RandomizedSearch tuning, I got the below accuracy and the best model parameters



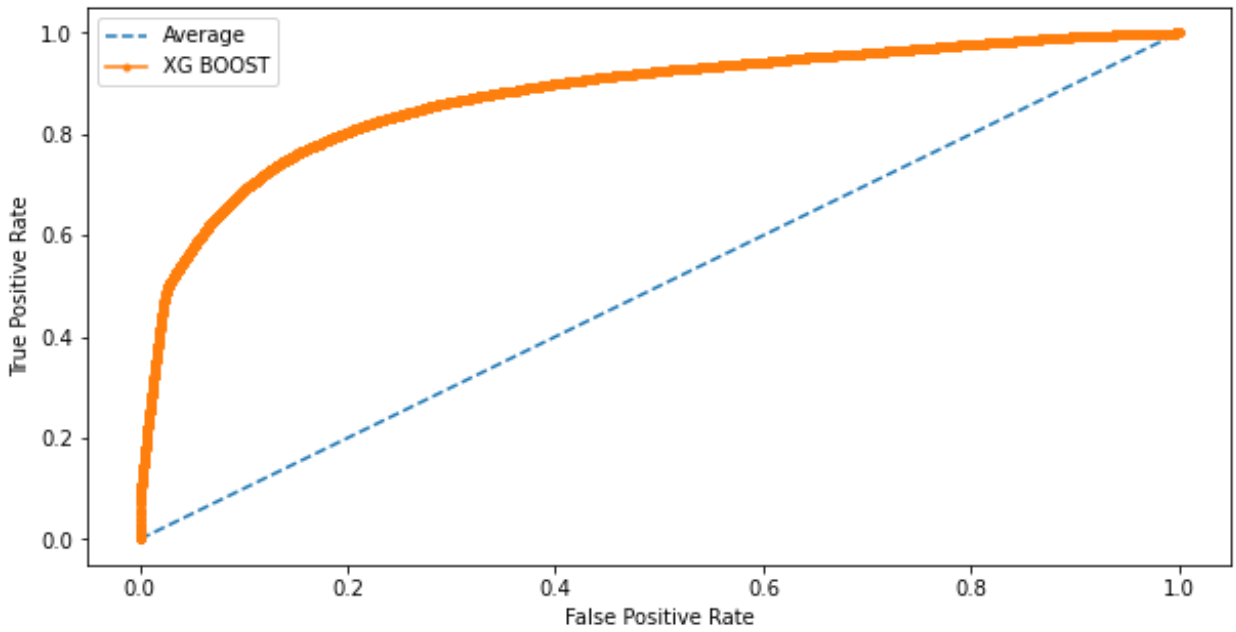
These are the top 10 parameters:

1. Credit_Product_Unknown
2. Credit_Product_Yes
3. Occupation_Salaried
4. Age
5. Vintage
6. Is_Active
7. Channel_Code_X3
8. Occupation_other
9. Occupation_Self_employed
10. Channel_Code_X4

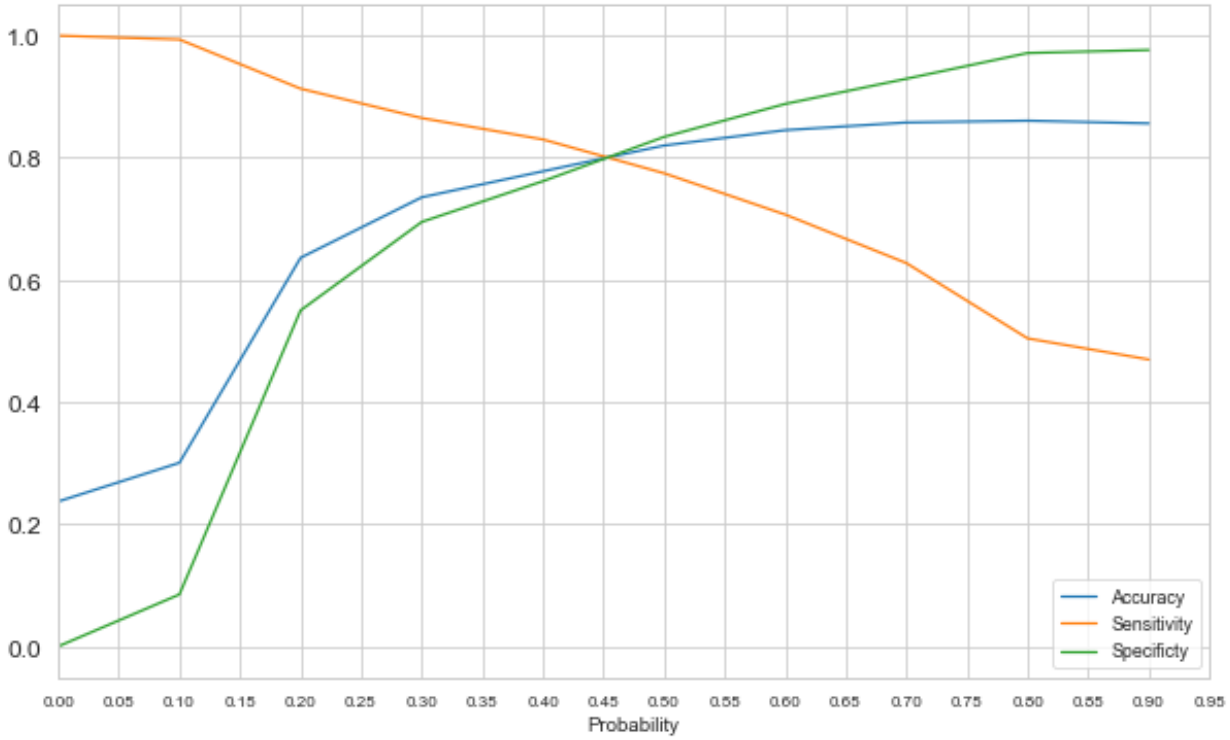
As, we have seen in the data exploration phase, the Credit Product Unknown category was the best feature to predict the Lead. Here, Unknown would mean that the customer has not taken any kind of credit from the bank before.

After that, we can see the Credit Product Yes. Which means the customer has an active credit product.

The other key parameters like salaried, age and if the customer is active describes the model very well.




The ROC -AUC score is 0.847 which is quite descent score.




The above is the Accuracy, Sensitivity Specificity graph which tells that the optimum value of the threshold to decide if the customer is a probable Lead or not.

The customers having the probability score more than this threshold can be considered to be contacted by the bank as they are most likely to take the credit cards.

## Next Steps

The XGboost model has given a decent accuracy score. But we can always try new models to see if the score increases. Also, we can ask the bank to get more accurate datasets, if possible. From my side if time would have permitted, I would try stacking classifier models and see it helps in getting better score.