Problem Statement ## Credit Card Lead Prediction

Happy Customer Bank is a mid-sized private bank that deals in all kinds of banking products, like Savings accounts, Current accounts, investment products, credit products, among other offerings.

The bank also cross-sells products to its existing customers and to do so they use different kinds of communication like tele-calling, e-mails, recommendations on net banking, mobile banking, etc.

In this case, the Happy Customer Bank wants to cross sell its credit cards to its existing customers. The bank has identified a set of customers that are eligible for taking these credit cards.

Now, the bank is looking for your help in identifying customers that could show higher intent towards a recommended credit card, given:

```
- Customer details (gender, age, region etc.)
- Details of his/her relationship with the bank (Channel_Code,Vintage, 'Avg_Asset_Value etc.)
```

Data Dictionary

Train Data

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 0 : Customer is not interested 1 : Customer is interested

Test Data

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

Sample Submission

This file contains the exact submission format for the predictions. Please submit CSV file only.

Variable : Definition

ID **:** Unique Identifier for a row

Is_Lead (Target)**:** Probability of Customer showing interest (class 1)

Table of Content

- Step 1: Importing the Relevant Libraries
- Step 2: Data Inspection
- Step 3: Data Cleaning
- Step 4: Exploratory Data Analysis
- Step 5: Building Model
- Submission

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Data Inspetion

```
# loading dataset
# train dataset in Train df
# test dataset in Test df
train=pd.read_csv('train_s3TEQDk.csv')
test=pd.read_csv('test_mSzZ8RL.csv')
```

```
# Checking no. of rows and columns in Train data
train.shape
```

```
(245725, 11)
```

- We have 245725 rows and 11 column in Train Dataset

```
# Checking no. of rows and columns in Test data
test.shape
```

```
(105312, 10)
```

- We have 105313 rows and 10 column in test set

```
#Taking look at first few entries in Train Data
train.head()
```

	ID	Gender	Age	Region_Code	Occupation	Channel_Code	Vintage	Credit
0	NNVBBKZB	Female	73	RG268	Other	X3	43	
1	IDD62UNG	Female	30	RG277	Salaried	X1	32	
2	HD3DSEMC	Female	56	RG268	Self_Employed	X3	26	
3	BF3NC7KV	Male	34	RG270	Salaried	X1	19	
4	TEASRWXV	Female	30	RG282	Salaried	X1	33	

```
#Taking look at first few entries in Test Data.
test.head()
```

	ID	Gender	Age	Region_Code	Occupation	Channel_Code	Vintage	Credit_F
0	VBENBARO	Male	29	RG254	Other	X1	25	
1	CCMEWNKY	Male	43	RG268	Other	X2	49	
2	VK3KGA9M	Male	31	RG270	Salaried	X1	14	
3	TT8RPZVC	Male	29	RG272	Other	X1	33	
4	SHQZEYTZ	Female	29	RG270	Other	X1	19	

- By looking at Train and Test data, we have some categorical features.
- Also Is_Lead is our target feature which we need to predict for Test data.
- We are having some missing values too can bee seen in Test data.

+ Code + Text

Checking for Missing/Null values in both the dataset(Train & Test)

[ ] ↳ *4 cells hidden*

Checking the no. of categorical & numerical feature in Train & Test Dataset

[ ] ↳ *2 cells hidden*

Data Prepration/Data Cleaning

```
# Checking the total no of missing values in Train dataset
train.isnull().sum()
```

```
ID                         0
Gender                     0
Age                        0
Region_Code                0
Occupation                 0
Channel_Code               0
Vintage                    0
Credit_Product         29325
Avg_Account_Balance        0
Is_Active                  0
Is_Lead                    0
dtype: int64
```

- Total of 29325 values are null out of 245725 values in Train dataset

```
# Checking the total no of missing values in Test dataset
test.isnull().sum()
```

```
ID                         0
Gender                     0
Age                        0
Region_Code                0
Occupation                 0
Channel_Code               0
Vintage                    0
Credit_Product         12522
Avg_Account_Balance        0
Is_Active                  0
dtype: int64
```

- Total of 12522 values are null out of 105312 Values in test set

```
# Creating a new dataframe df similar to train dataset
df=train
```

```
# Checking the counts of different tpes of value in feature "Credit_Product"
df['Credit_Product'].value_counts()
```

```
No     144357
Yes     72043
Name: Credit_Product, dtype: int64
```

- Can be seen that "No" is most frequent
- Imputing the missing values in Train dataset with most frequent occuring value i.e "No"

```
#using SimpleImputer Function to imput the missing value
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
imputer.fit(df[['Credit_Product']])
df[['Credit_Product']] = imputer.transform(df[['Credit_Product']])
```

```
#Checking for missing values after imputation
df.isnull().sum()
```

```
ID                     0
Gender                 0
Age                    0
Region_Code            0
Occupation             0
Channel_Code           0
Vintage                0
Credit_Product         0
Avg_Account_Balance    0
Is_Active              0
Is_Lead                0
dtype: int64
```

- There is no null values now as we have imputed them with most frequent value.

```
# Checking the counts of different tpes of value in Target feature "Is_Lead"
df['Is_Lead'].value_counts()
```

```
0    187437
1     58288
Name: Is_Lead, dtype: int64
```

- It shows that our dataset is Imbalanced
- So we we train our model with the same, It will be biased towards 0 target
- Will use **OverSampling** Technique for this

```
# Creating a new dataframe with value of Target Feature("Is_Lead") equal to 1
onedf=df[df['Is_Lead']==1]
```

```
onedf['Is_Lead'].value_counts()
```

```
1    58288
Name: Is_Lead, dtype: int64
```

```
#Appending the newly created dataframe "onedf" to df
df=df.append(onedf).append(onedf)
```

- Appending newly created dataframe with entries having only 1 in "Is_Lead" feature to "df"
- This is done to avoid Biasing of prediction

```
# Checking the counts of different tpes of value in Target feature "Is_Lead"
df['Is_Lead'].value_counts()
```

```
0    187437
1    174864
Name: Is_Lead, dtype: int64
```

- now data is almost balanced

Imputin missing value in Test set too with same strategy used in Train set.

[ ] ↳ *4 cells hidden*

Exploratory Data Analysis

```
#Checking the column names in Dataframe
df.columns
```

```
Index(['ID', 'Gender', 'Age', 'Region_Code', 'Occupation', 'Channel_Code',
       'Vintage', 'Credit_Product', 'Avg_Account_Balance', 'Is_Active',
```

```
       'Is_Lead'],
      dtype='object')
```

```
# checking how many different types of values are there in Gender feature
df['Gender'].value_counts()
```

```
Male      205363
Female    156938
Name: Gender, dtype: int64
```

```
# checking how many different types of values are there in Occupation feature
df['Occupation'].value_counts()
```

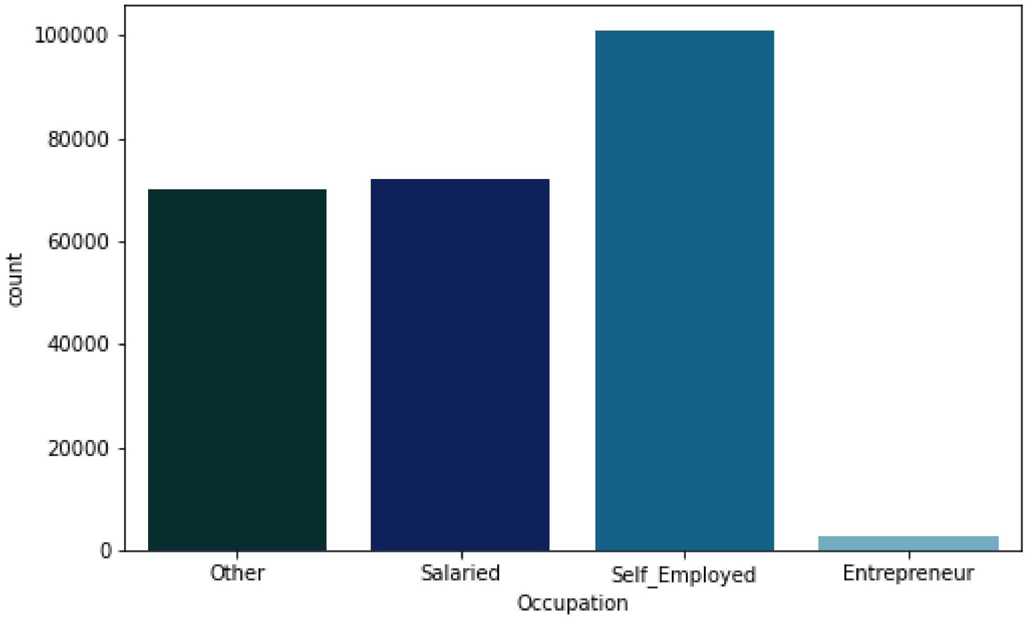
```
Self_Employed    156568
Other            104551
Salaried          94991
Entrepreneur       6191
Name: Occupation, dtype: int64
```

```
# checking how many different types of values are there in Channel_Code feature
df['Channel_Code'].value_counts()
```

```
X1    122682
X3    119150
X2    112140
X4      8329
Name: Channel_Code, dtype: int64
```

```
# plotting Occupation Feature to gain Insight
plt.figure(figsize=(8,5))
sns.countplot('Occupation',data=train,palette='ocean')
```

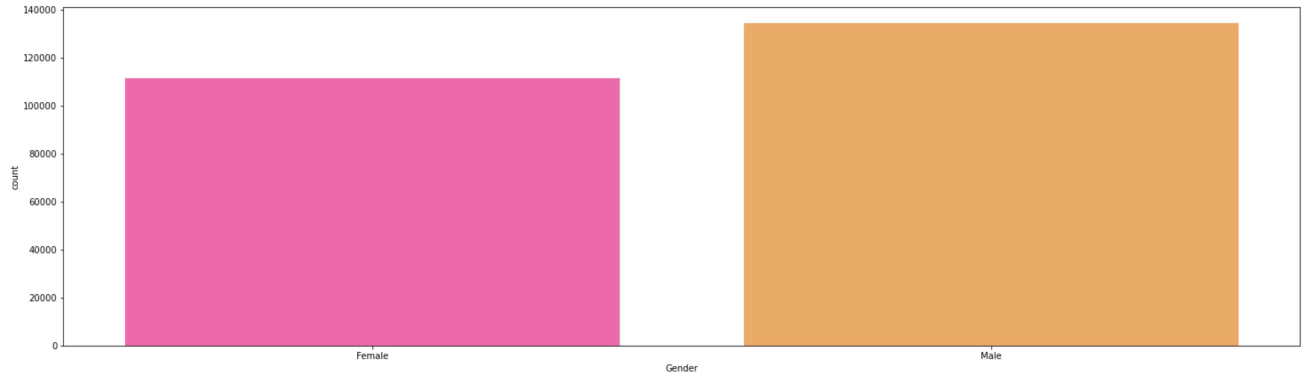
```
<AxesSubplot:xlabel='Occupation', ylabel='count'>
```



- Most of the customers self_Employed, While customers with occupation as Others & Salaried are almost equal, where as customers who Entreprenuer are very few as compared to rest

```
# plotting Gender Feature to gain Insight
plt.figure(figsize=(25,7))
sns.countplot('Gender',data=train,palette='spring')
```

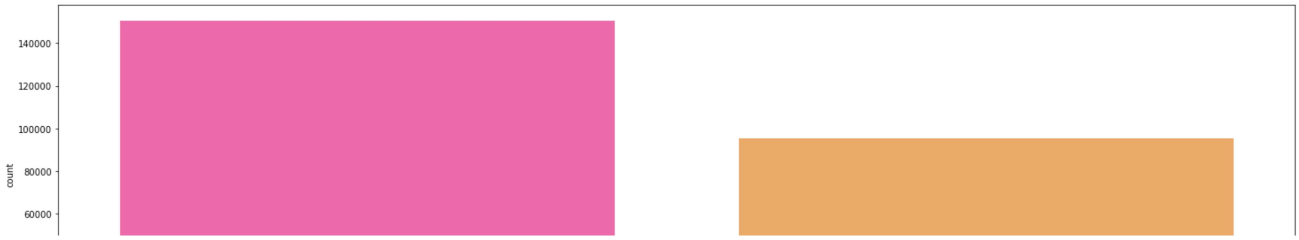
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



- Males customers are more as compared to Female

```
# plotting Is_Active Feature to gain Insight
plt.figure(figsize=(25,7))
sns.countplot('Is_Active',data=train,palette='spring')
```

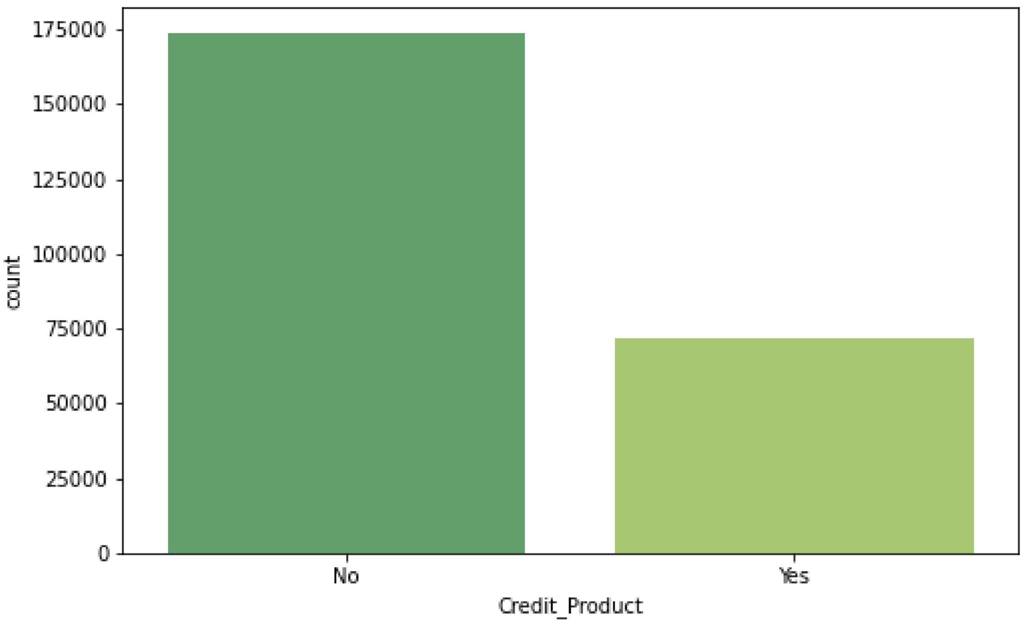
<AxesSubplot:xlabel='Is_Active', ylabel='count'>



- Max No of users are Inactive since last three months

```
# plotting Credit_Product Feature to gain Insight
plt.figure(figsize=(8,5))
sns.countplot('Credit_Product',data=train,palette='summer')
```

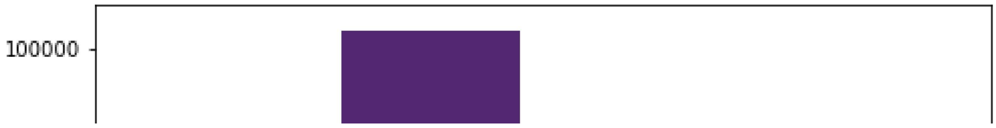
<AxesSubplot:xlabel='Credit_Product', ylabel='count'>



- Max no of customers in bank have no active credit product(i.e. Loans etc)

```
# plotting Gender Feature to gain Insight
plt.figure(figsize=(8,5))
sns.countplot('Channel_Code',data=train,palette='twilight')
```

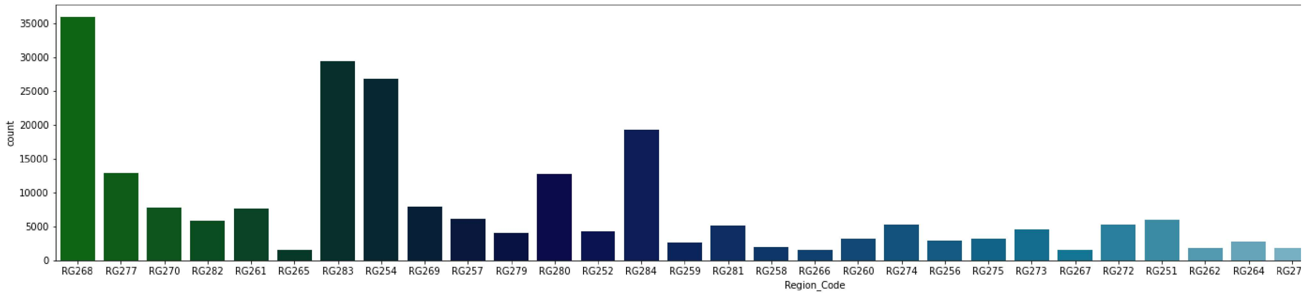
<AxesSubplot:xlabel='Channel_Code', ylabel='count'>



- channel code 1 count is more

```
# plotting Gender Feature to gain Insight
plt.figure(figsize=(29,5))
sns.countplot('Region_Code',data=train,palette='ocean')
```

<AxesSubplot:xlabel='Region_Code', ylabel='count'>



Building Model

```
#  Using Labelencoder to encode categorical features
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
var_mod = df.select_dtypes(include='object').columns
for i in var_mod:
    df[i] = le.fit_transform(df[i])

for i in var_mod:
    df1[i] = le.fit_transform(df1[i])
```

- Encoding the categorical features for model training purpose using label encode function from scikit learn

Separating target Feature & Independent Features

[ ] ↳ *1 cell hidden*

splitting the train Dataset in to train and validation

[ ] ↳ *2 cells hidden*

Training CatBoostClassifier

[ ] ↳ *10 cells hidden*