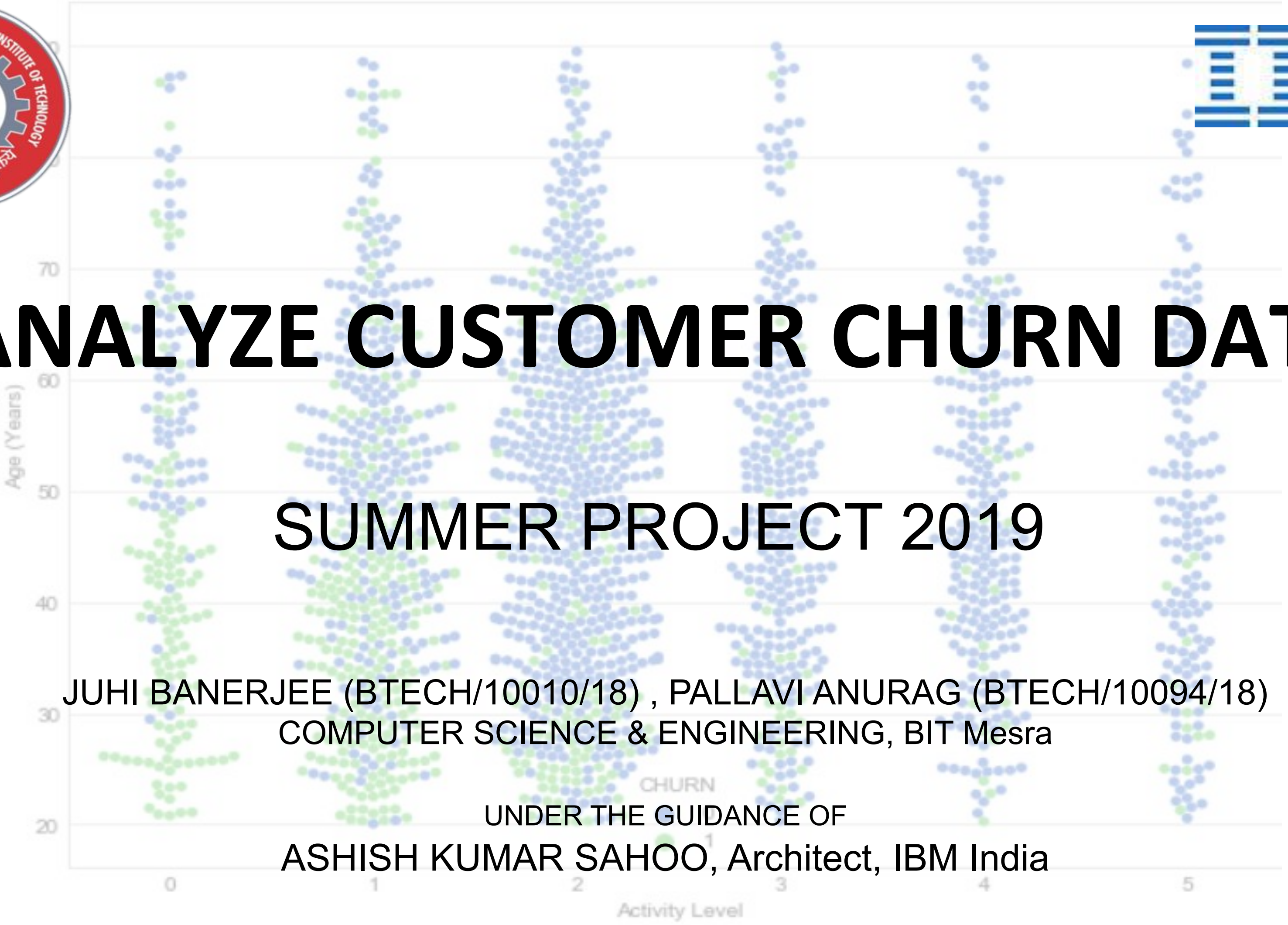


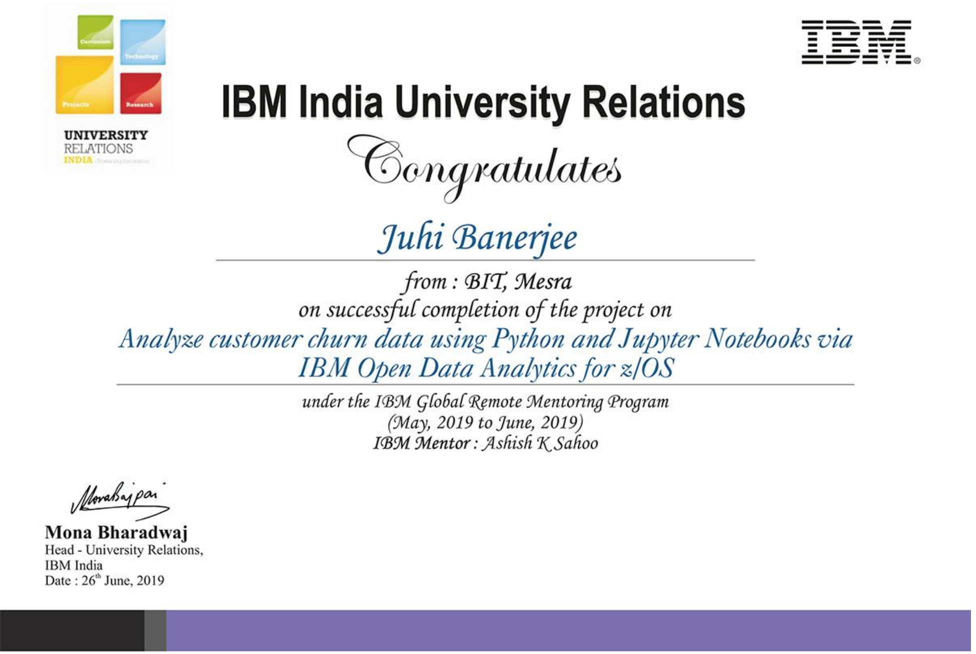
# ANALYZE CUSTOMER CHURN DATA

## SUMMER PROJECT 2019

JUHI BANERJEE (BTECH/10010/18) , PALLAVI ANURAG (BTECH/10094/18)
COMPUTER SCIENCE & ENGINEERING, BIT Mesra

UNDER THE GUIDANCE OF
ASHISH KUMAR SAHOO, Architect, IBM India

# Certificate of Completion



UNIVERSITY RELATIONS INDIA

IBM India University Relations

Congratulates

*Juhi Banerjee*

*from : BIT, Mesra*

*on successful completion of the project on*

*Analyze customer churn data using Python and Jupyter Notebooks via IBM Open Data Analytics for z/OS*

*under the IBM Global Remote Mentoring Program*

*(May, 2019 to June, 2019)*

*IBM Mentor : Ashish K Sahoo*

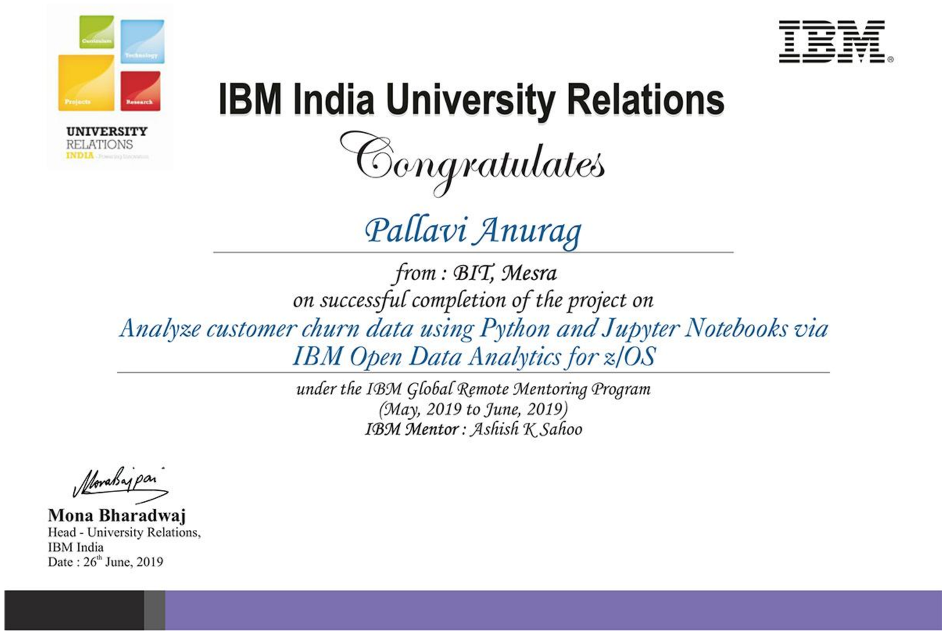
[illegible]

**Mona Bharadwaj**

Head - University Relations,

IBM India

Date : 26th June, 2019



UNIVERSITY RELATIONS INDIA

IBM India University Relations

Congratulates

*Pallavi Anurag*

*from : BIT, Mesra*

*on successful completion of the project on*

*Analyze customer churn data using Python and Jupyter Notebooks via IBM Open Data Analytics for z/OS*

*under the IBM Global Remote Mentoring Program*

*(May, 2019 to June, 2019)*

*IBM Mentor : Ashish K Sahoo*

[illegible]

**Mona Bharadwaj**

Head - University Relations,

IBM India

Date : 26th June, 2019

## Acknowledgement

We are grateful to Mr. Ashish Kumar Sahoo, Mainframe Architect IBM India, who is also our mentor and guide for the project. In spite of his hectic schedule, he reviewed our work, explained and guided us throughout the project.

## Table of Contents

## Problem Statement

Imagine you are a bank with client retention issues — your customers are leaving the bank (churning). Using Jupyter Notebooks with IBM Open Data Analytics for z/OS (IzODA) to look at credit card transactional data, with the various Python libraries and the optimized data layer provided by IzODA, you can create robust data visualizations that allow you to look for key features as to why a customer may want to leave your bank.

Input: Client information table with 6,001 records
Credit card transaction table with about 1.5 million records.

## Approach

- Learning Python, Matplotlib & Seaborn
    - HackerRank and IBM badges
- Understand the code in IBM Code Patterns
- Solution development with input data as CSV file in local host
    - Data preprocessing
    - Initial analysis through data visualization
    - Basics of supervised learning and classification problems
    - Prediction of customer churn using multiple models
    - Business loss analysis and Age loss analysis
- Solution development with input data from Mainframe (DB2 and VSAM) reusing above code in mainframe environment (zTrial)

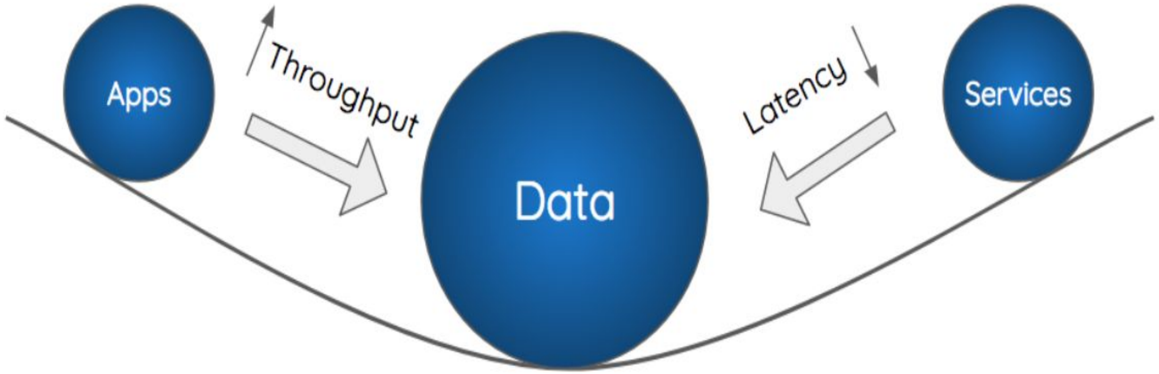
# Advantages of Analytics on z/OS

**Data Gravity** - https://medium.com/clarisights/data-gravity-why-should-marketers-care-273b4d8f7a4b

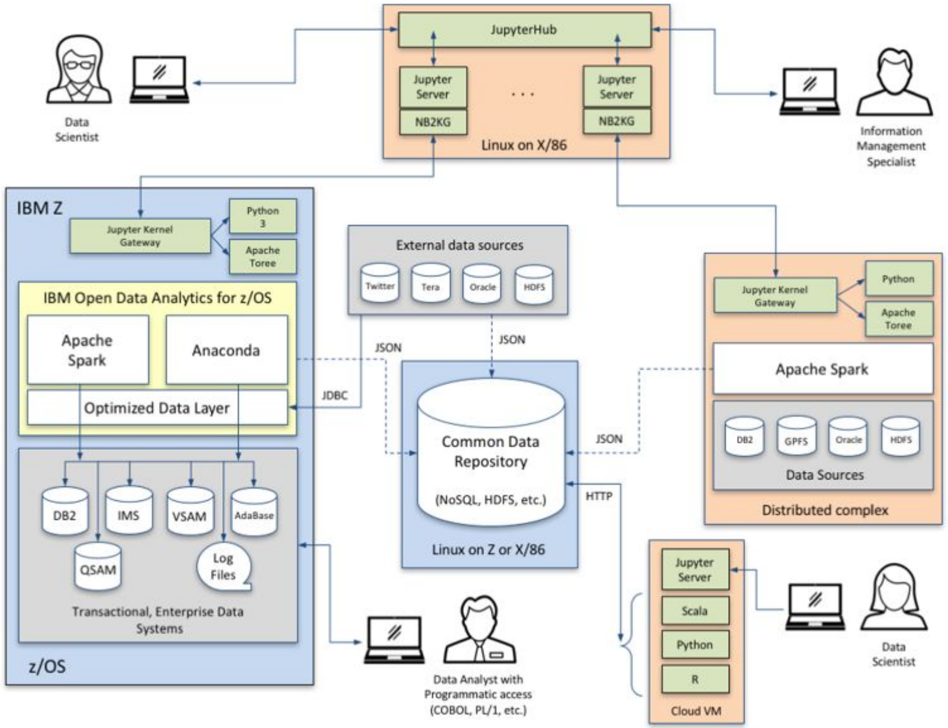
- There is no movement of data. So cost of data is reduced.
- Since there is no data movement, there is less latency.

**Data Security**

- Because the data remains in the Mainframe, it remains secured.



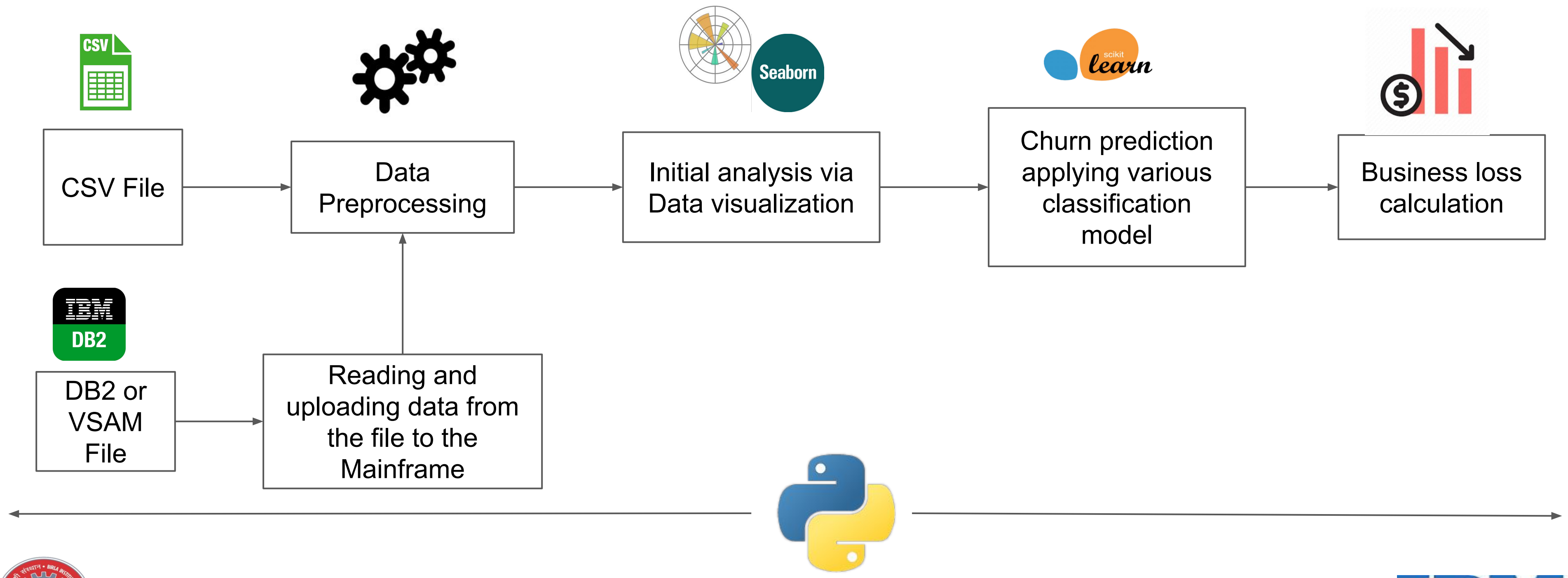
# What is IzODA?



Various types of data stores and databases are used in Mainframe. IzODA is a data virtualization layer that enables us to use various non-relational data stores by simple sql queries. Through IzODA we can access VSAM, IMS, DB2, etc and also create a join between different types of data sources.

# Code Components

## Architecture


CSV
CSV File
Data Preprocessing
Seaborn
Initial analysis via Data visualization
scikit learn
Churn prediction applying various classification model
Business loss calculation
IBM DB2
DB2 or VSAM File
Reading and uploading data from the file to the Mainframe

## Detail look at the Code blocks

**Data Preprocessing**

1. Importing client and transaction datasets.
2. Merging the transaction datasets.
3. Adding new features to client dataset upon initial analysis of transaction dataset.

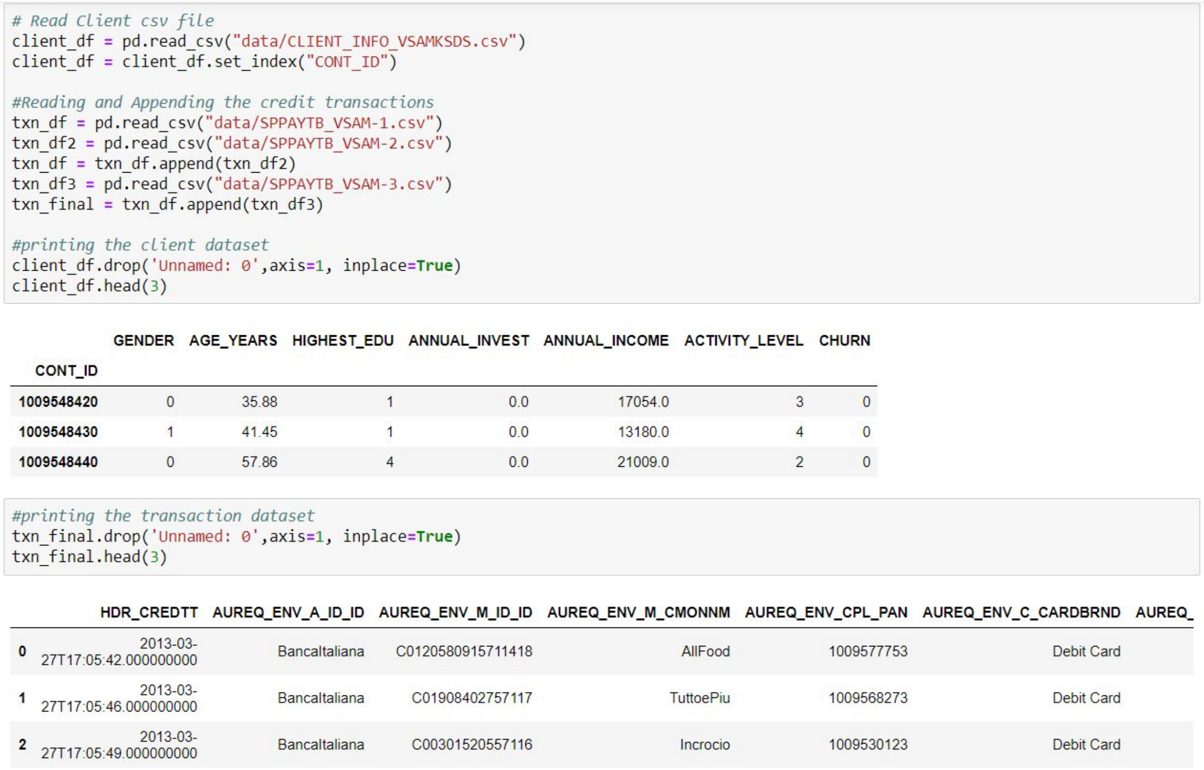
**Data Visualization**

1. Plotting techniques using matplotlib and seaborn

**Churn Prediction**

1. Applying 5 different classification models to predict customer churn but using the one given by IBM Code Patterns to calculate the overall loss.
2. Calculate overall business loss based on the predicted churn.

# Data Preprocessing - I



```
# Read Client csv file
client_df = pd.read_csv("data/CLIENT_INFO_VSAMKSDS.csv")
client_df = client_df.set_index("CONT_ID")

#Reading and Appending the credit transactions
txn_df = pd.read_csv("data/SPPAYTB_VSAM-1.csv")
txn_df2 = pd.read_csv("data/SPPAYTB_VSAM-2.csv")
txn_df = txn_df.append(txn_df2)
txn_df3 = pd.read_csv("data/SPPAYTB_VSAM-3.csv")
txn_final = txn_df.append(txn_df3)

#printing the client dataset
client_df.drop('Unnamed: 0',axis=1, inplace=True)
client_df.head(3)
```

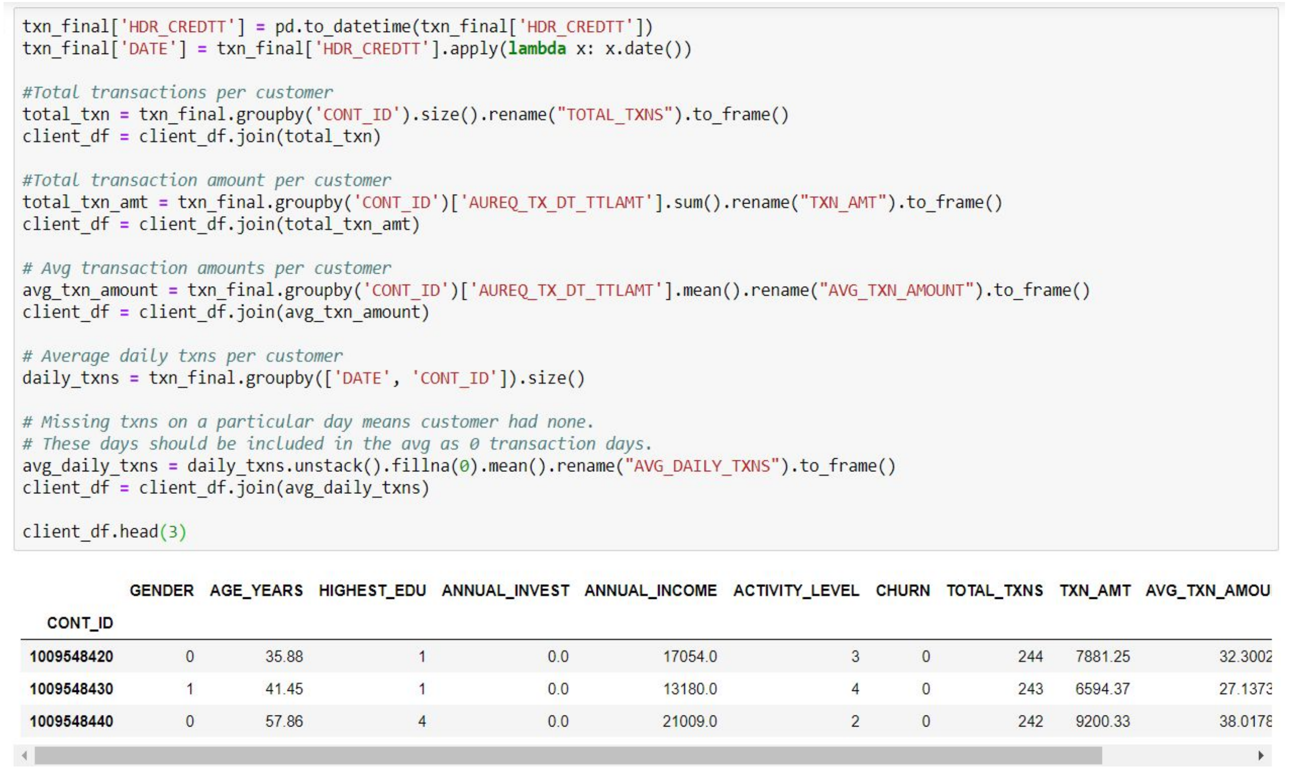
| | GENDER | AGE_YEARS | HIGHEST_EDU | ANNUAL_INVEST | ANNUAL_INCOME | ACTIVITY_LEVEL | CHURN |
|---|---|---|---|---|---|---|---|
| **CONT_ID** | | | | | | | |
| **1009548420** | 0 | 35.88 | 1 | 0.0 | 17054.0 | 3 | 0 |
| **1009548430** | 1 | 41.45 | 1 | 0.0 | 13180.0 | 4 | 0 |
| **1009548440** | 0 | 57.86 | 4 | 0.0 | 21009.0 | 2 | 0 |

```
#printing the transaction dataset
txn_final.drop('Unnamed: 0',axis=1, inplace=True)
txn_final.head(3)
```

| | HDR_CREDTT | AUREQ_ENV_A_ID_ID | AUREQ_ENV_M_ID_ID | AUREQ_ENV_M_CMONNM | AUREQ_ENV_CPL_PAN | AUREQ_ENV_C_CARDBRND | AUREQ_ |
|---|---|---|---|---|---|---|---|
| **0** | 2013-03-27T17:05:42.000000000 | BancaItaliana | C0120580915711418 | AllFood | 1009577753 | Debit Card | |
| **1** | 2013-03-27T17:05:46.000000000 | BancaItaliana | C01908402757117 | TuttoePiu | 1009568273 | Debit Card | |
| **2** | 2013-03-27T17:05:49.000000000 | BancaItaliana | C00301520557116 | Incrocio | 1009530123 | Debit Card | |

- Reading the csv file.
- Merging the transaction datasets.
- Dropping the 'Unnamed: 0' column and printing the first three rows of the client and transaction dataset.
- The client dataset has 6001 rows and 7 columns and the transaction dataset has 1497299 rows and 14 columns

# Data Preprocessing - II



```python
txn_final['HDR_CREDTT'] = pd.to_datetime(txn_final['HDR_CREDTT'])
txn_final['DATE'] = txn_final['HDR_CREDTT'].apply(lambda x: x.date())

#Total transactions per customer
total_txn = txn_final.groupby('CONT_ID').size().rename("TOTAL_TXNS").to_frame()
client_df = client_df.join(total_txn)

#Total transaction amount per customer
total_txn_amt = txn_final.groupby('CONT_ID')['AUREQ_TX_DT_TTLAMT'].sum().rename("TXN_AMT").to_frame()
client_df = client_df.join(total_txn_amt)

# Avg transaction amounts per customer
avg_txn_amount = txn_final.groupby('CONT_ID')['AUREQ_TX_DT_TTLAMT'].mean().rename("AVG_TXN_AMOUNT").to_frame()
client_df = client_df.join(avg_txn_amount)

# Average daily txns per customer
daily_txns = txn_final.groupby(['DATE', 'CONT_ID']).size()

# Missing txns on a particular day means customer had none.
# These days should be included in the avg as 0 transaction days.
avg_daily_txns = daily_txns.unstack().fillna(0).mean().rename("AVG_DAILY_TXNS").to_frame()
client_df = client_df.join(avg_daily_txns)

client_df.head(3)
```

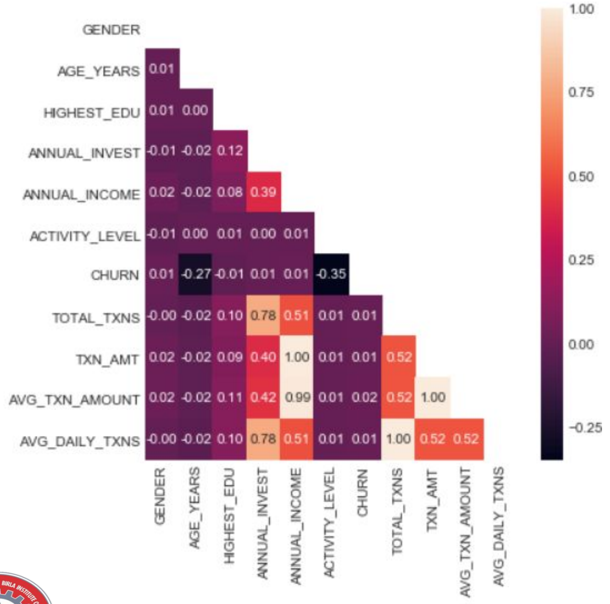
```
            GENDER  AGE_YEARS  HIGHEST_EDU  ANNUAL_INVEST  ANNUAL_INCOME  ACTIVITY_LEVEL  CHURN  TOTAL_TXNS  TXN_AMT  AVG_TXN_AMOU
CONT_ID
1009548420       0      35.88            1            0.0        17054.0               3      0         244  7881.25       32.3002
1009548430       1      41.45            1            0.0        13180.0               4      0         243  6594.37       27.1373
1009548440       0      57.86            4            0.0        21009.0               2      0         242  9200.33       38.0178
```

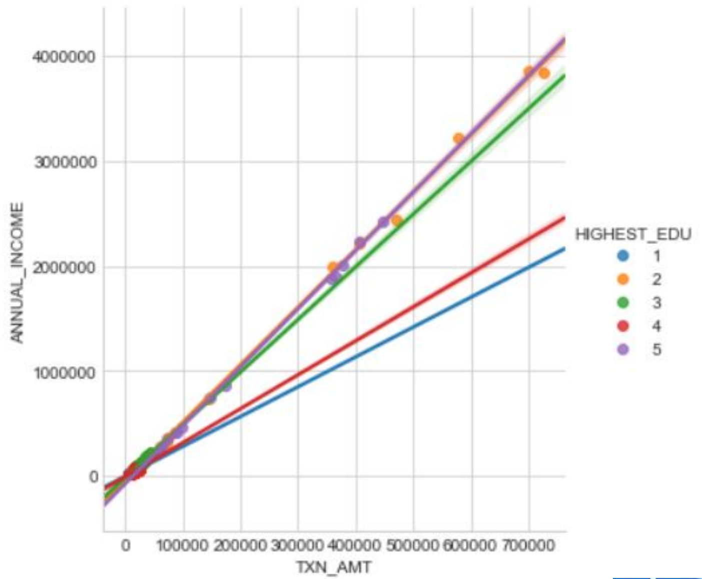
- Creating a new column which stores the date on which a customer transacted.
- Creating four new rows which stores the transaction details such as total transactions, total amount transacted, average transaction and average daily transaction per customer.

# Data Visualization - I

Finding the correlation between the datas and plotting it as a colour-coded matrix with the help of **heatmap**. Showing only the lower triangle of the matrix as it is symmetric.

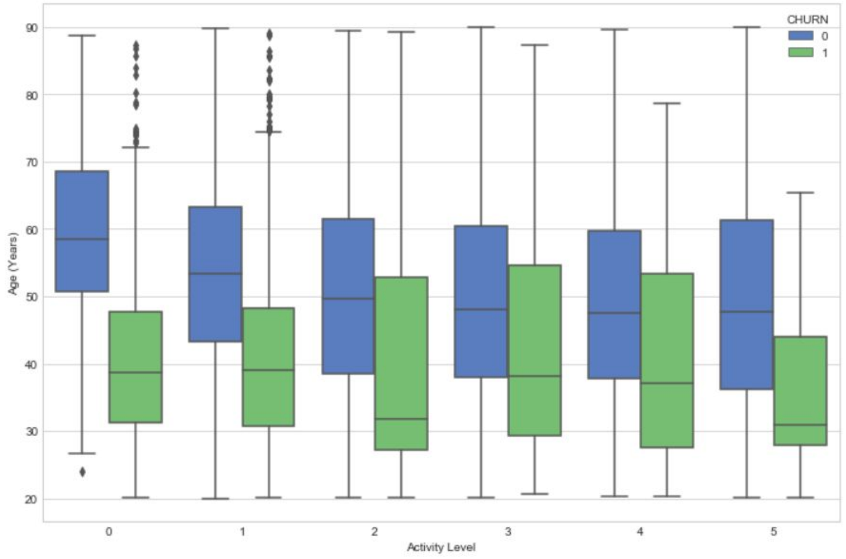
Relation between annual income and transaction amount - We see, from the **swarm plot**, more the annual income, more is the transaction amount of a customer with highest education level (level - 5).



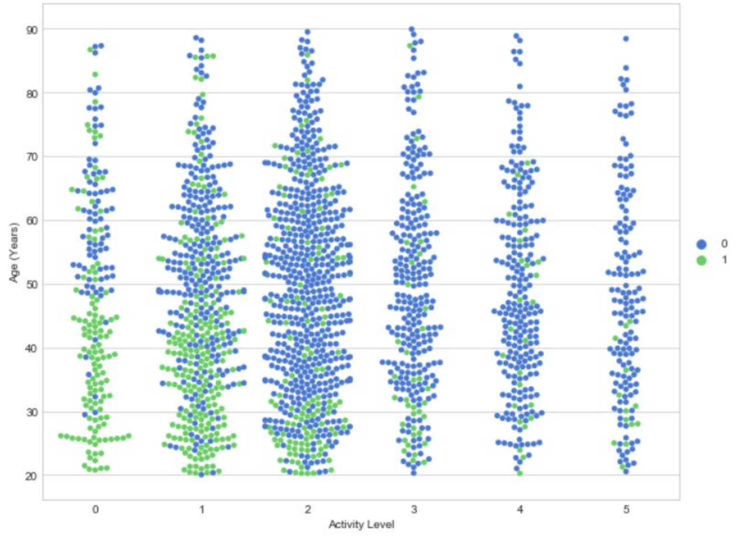


# Data Visualization - II

From the **box plot** and the **swarm plot**, we see that the mean age (in years) during which a customer might churn is between 30 - 40 and during which the customer will probably not churn is between 45 - 60.



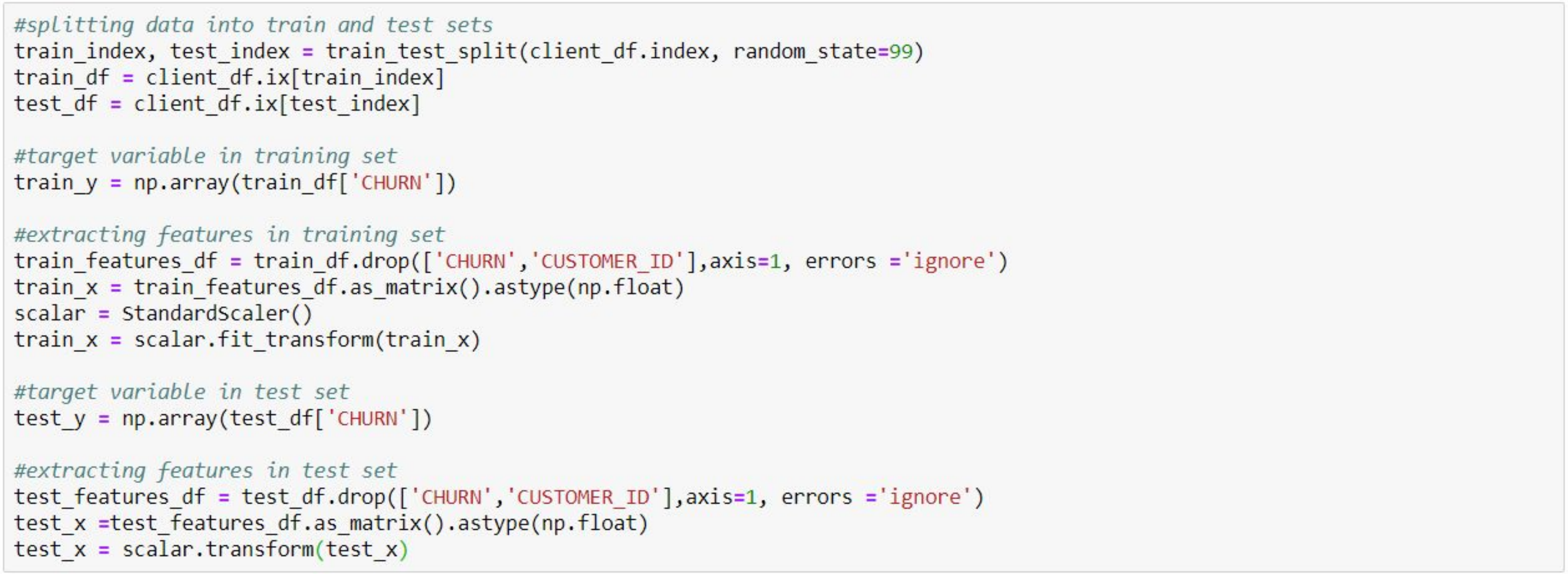
Box Plot



Swarm Plot

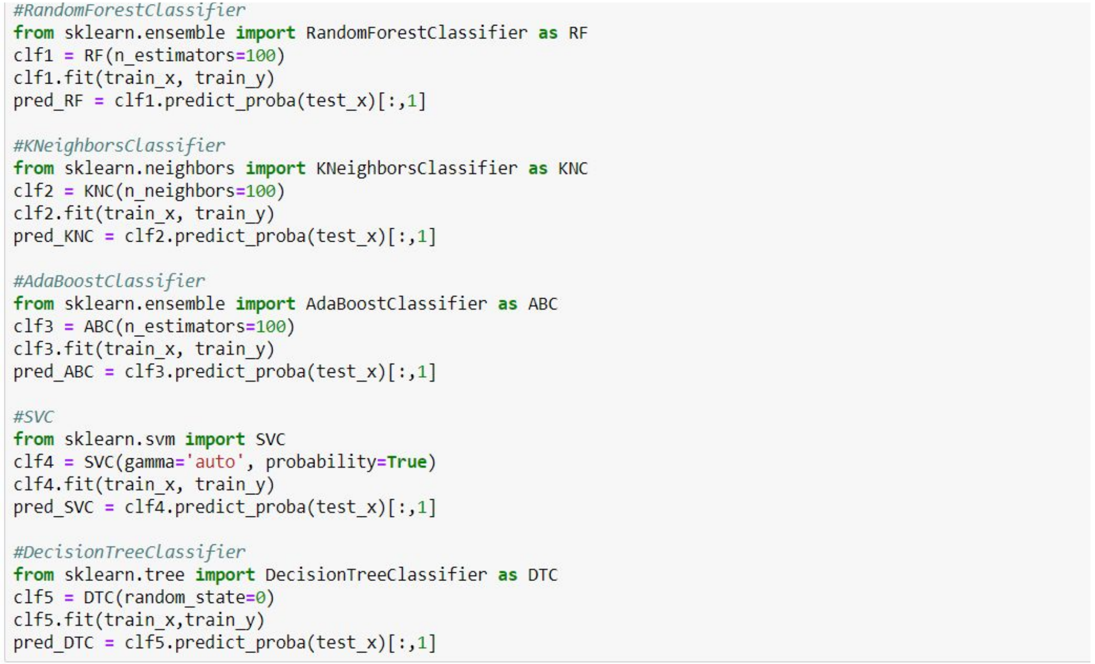
# Churn Prediction - I

Splitting the data into train and test sets to avoid **overfitting**.



```
#splitting data into train and test sets
train_index, test_index = train_test_split(client_df.index, random_state=99)
train_df = client_df.ix[train_index]
test_df = client_df.ix[test_index]

#target variable in training set
train_y = np.array(train_df['CHURN'])

#extracting features in training set
train_features_df = train_df.drop(['CHURN','CUSTOMER_ID'],axis=1, errors ='ignore')
train_x = train_features_df.as_matrix().astype(np.float)
scalar = StandardScaler()
train_x = scalar.fit_transform(train_x)

#target variable in test set
test_y = np.array(test_df['CHURN'])

#extracting features in test set
test_features_df = test_df.drop(['CHURN','CUSTOMER_ID'],axis=1, errors ='ignore')
test_x =test_features_df.as_matrix().astype(np.float)
test_x = scalar.transform(test_x)
```

# Churn Prediction - II



```
#RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier as RF
clf1 = RF(n_estimators=100)
clf1.fit(train_x, train_y)
pred_RF = clf1.predict_proba(test_x)[:,1]

#KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier as KNC
clf2 = KNC(n_neighbors=100)
clf2.fit(train_x, train_y)
pred_KNC = clf2.predict_proba(test_x)[:,1]

#AdaBoostClassifier
from sklearn.ensemble import AdaBoostClassifier as ABC
clf3 = ABC(n_estimators=100)
clf3.fit(train_x, train_y)
pred_ABC = clf3.predict_proba(test_x)[:,1]

#SVC
from sklearn.svm import SVC
clf4 = SVC(gamma='auto', probability=True)
clf4.fit(train_x, train_y)
pred_SVC = clf4.predict_proba(test_x)[:,1]

#DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier as DTC
clf5 = DTC(random_state=0)
clf5.fit(train_x,train_y)
pred_DTC = clf5.predict_proba(test_x)[:,1]
```

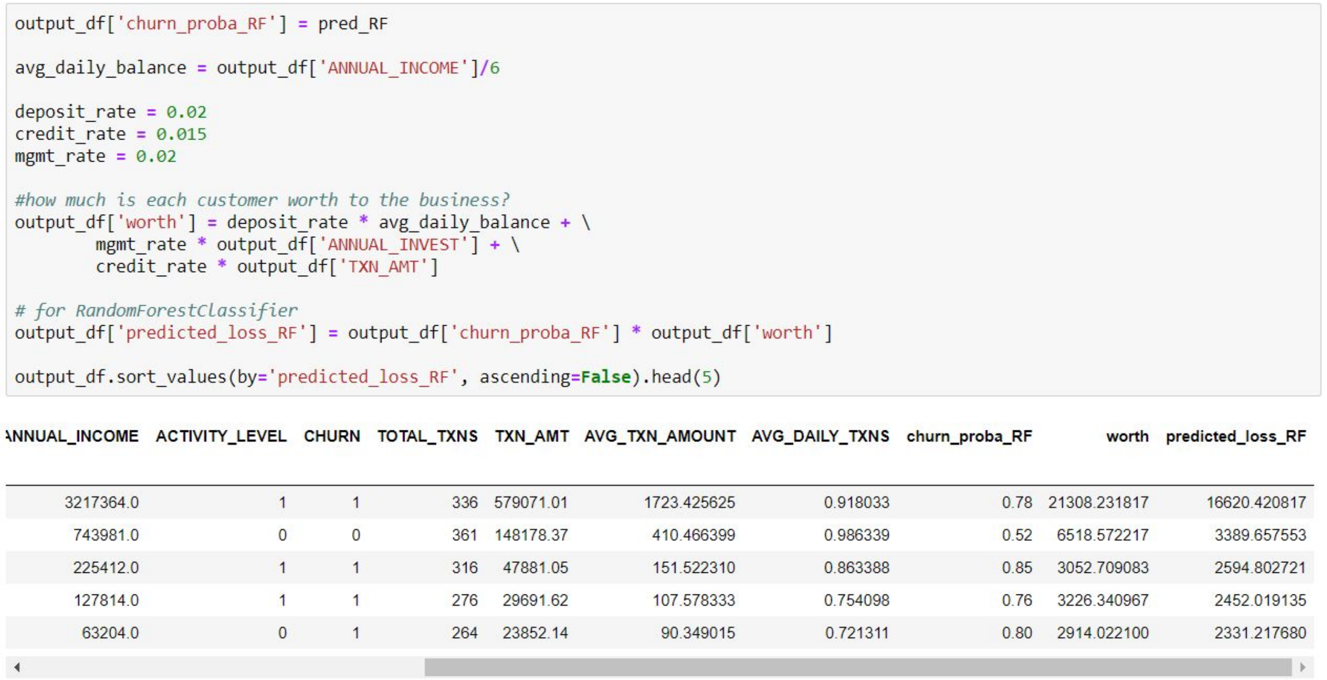
Using 5 different classification models.

- Random Forest Classifier*
- K-Neighbors Classifier
- AdaBoost Classifier
- SVC
- Decision Tree Classifier

*Random Forest Classifier is used in IBM Code Patterns. The other four models were done by us.

# Business Loss Analysis

Finding out the worth of each customer to the business and predicting the loss per customer using Random Forest Classifier.

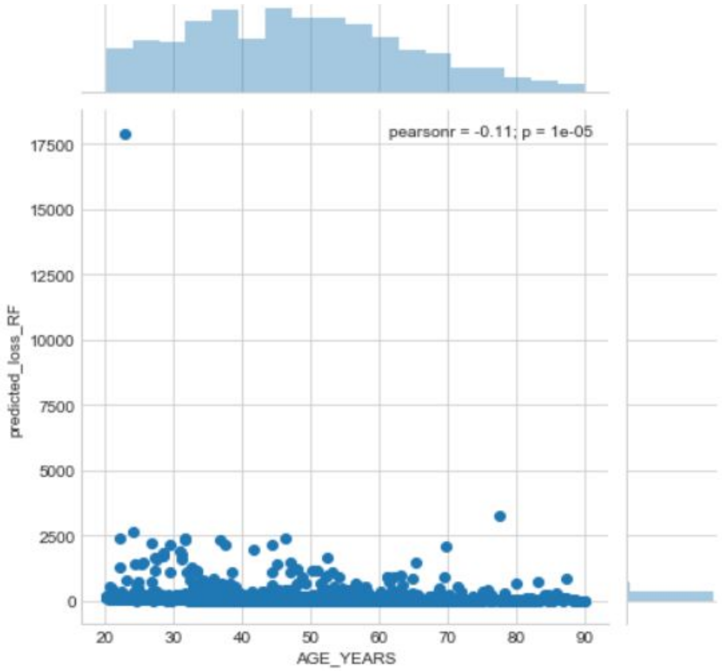


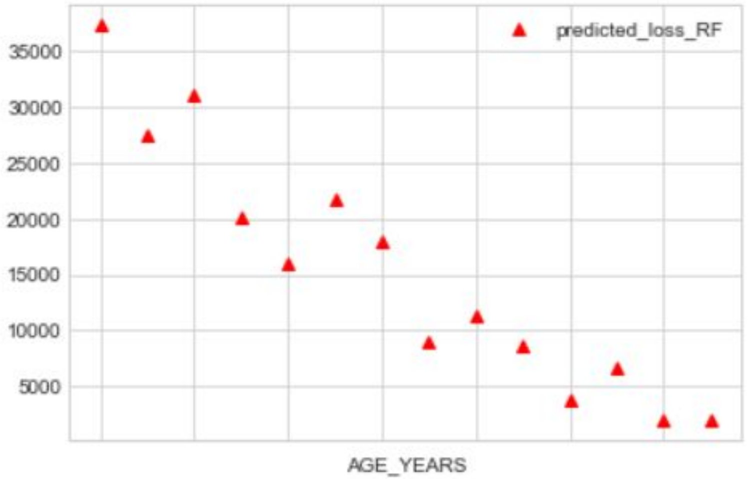
```
output_df['churn_proba_RF'] = pred_RF

avg_daily_balance = output_df['ANNUAL_INCOME']/6

deposit_rate = 0.02
credit_rate = 0.015
mgmt_rate = 0.02

#how much is each customer worth to the business?
output_df['worth'] = deposit_rate * avg_daily_balance + \
        mgmt_rate * output_df['ANNUAL_INVEST'] + \
        credit_rate * output_df['TXN_AMT']

# for RandomForestClassifier
output_df['predicted_loss_RF'] = output_df['churn_proba_RF'] * output_df['worth']

output_df.sort_values(by='predicted_loss_RF', ascending=False).head(5)
```

| NNUAL_INCOME | ACTIVITY_LEVEL | CHURN | TOTAL_TXNS | TXN_AMT | AVG_TXN_AMOUNT | AVG_DAILY_TXNS | churn_proba_RF | worth | predicted_loss_RF |
|---|---|---|---|---|---|---|---|---|---|
| 3217364.0 | 1 | 1 | 336 | 579071.01 | 1723.425625 | 0.918033 | 0.78 | 21308.231817 | 16620.420817 |
| 743981.0 | 0 | 0 | 361 | 148178.37 | 410.466399 | 0.986339 | 0.52 | 6518.572217 | 3389.657553 |
| 225412.0 | 1 | 1 | 316 | 47881.05 | 151.522310 | 0.863388 | 0.85 | 3052.709083 | 2594.802721 |
| 127814.0 | 1 | 1 | 276 | 29691.62 | 107.578333 | 0.754098 | 0.76 | 3226.340967 | 2452.019135 |
| 63204.0 | 0 | 1 | 264 | 23852.14 | 90.349015 | 0.721311 | 0.80 | 2914.022100 | 2331.217680 |

# Age Loss Analysis

From the graph, we see that for all the ages (in years), the predicted loss is almost between 0 - 2500.





# Difference Between Running the Analytics on Open System Data and the Data Residing in Mainframe

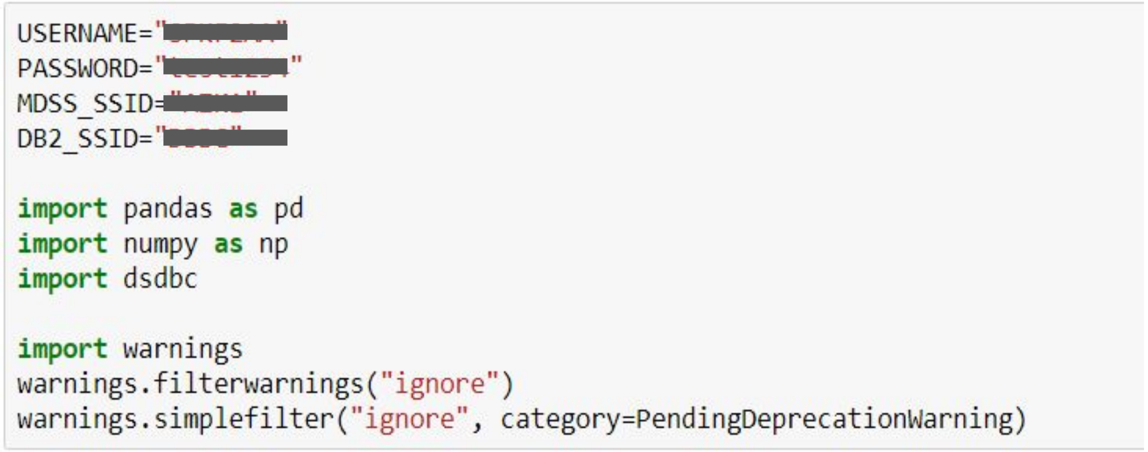
- In case of the CSV, we had two files which were accessed directly. We had to find the relationship between them and append them together.
- For Mainframe analytics (data in DB2 and VSAM), we require the Mainframe IP Address and the Port Number,SSID, username, password, MDSID.
- The data of DB2 and VSAM (through MDS) is accessed through SQL queries.
- Once we get the data, there is no difference between Mainframe and non-Mainframe analytics. The same Python code can be used.

# How to use zTrial?

- Go to the website - https://www.ibm.com/it-infrastructure/z/resources/trial
- Select Analytics and machine learning
- Select IBM Open Data Analytics
- Click on ‘Register for trial’
- To activate, change password of Analytics Instance
- Click on Start to activate and then upload the datas from https://github.com/ibmz-community-cloud/zAnalytics
- After uploading datas, click on Jupyter and start your code

Important Tip - Remember to stop the server when not in use.

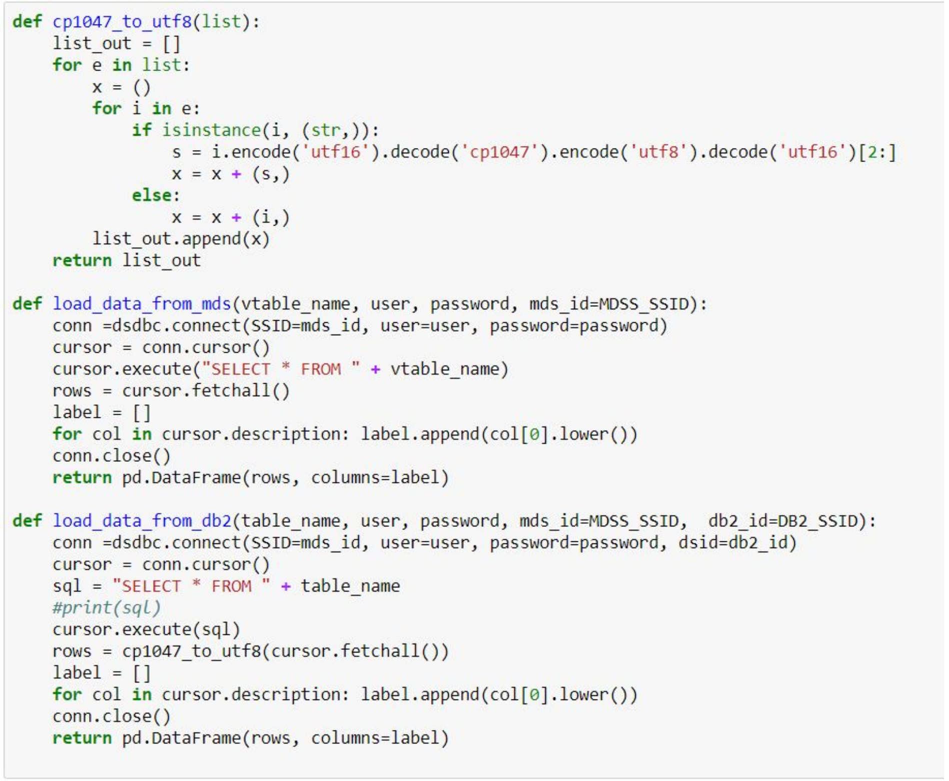
# Analytics in Mainframe zOS - I



```
USERNAME="[redacted]
PASSWORD="[redacted]"
MDSS_SSID=[redacted]
DB2_SSID="[redacted]

import pandas as pd
import numpy as np
import dsdbc

import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter("ignore", category=PendingDeprecationWarning)
```

- Set the DB2 and VSAM SSID and also set the Username and Password to access DB2 and MDS.
- Importing dsdbc to access files.

# Analytics in Mainframe zOS - II



```python
def cp1047_to_utf8(list):
    list_out = []
    for e in list:
        x = ()
        for i in e:
            if isinstance(i, (str,)):
                s = i.encode('utf16').decode('cp1047').encode('utf8').decode('utf16')[2:]
                x = x + (s,)
            else:
                x = x + (i,)
        list_out.append(x)
    return list_out

def load_data_from_mds(vtable_name, user, password, mds_id=MDSS_SSID):
    conn =dsdbc.connect(SSID=mds_id, user=user, password=password)
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM " + vtable_name)
    rows = cursor.fetchall()
    label = []
    for col in cursor.description: label.append(col[0].lower())
    conn.close()
    return pd.DataFrame(rows, columns=label)

def load_data_from_db2(table_name, user, password, mds_id=MDSS_SSID,  db2_id=DB2_SSID):
    conn =dsdbc.connect(SSID=mds_id, user=user, password=password, dsid=db2_id)
    cursor = conn.cursor()
    sql = "SELECT * FROM " + table_name
    #print(sql)
    cursor.execute(sql)
    rows = cp1047_to_utf8(cursor.fetchall())
    label = []
    for col in cursor.description: label.append(col[0].lower())
    conn.close()
    return pd.DataFrame(rows, columns=label)
```

- Converting data from EBCDIC to ASCII.
- Function to access datas of DB2 and MDS through SQL and accessing the VSAM file through MDS.

# Analytics in Mainframe zOS - III

The transaction datas are stored using the function load_data_from_db2

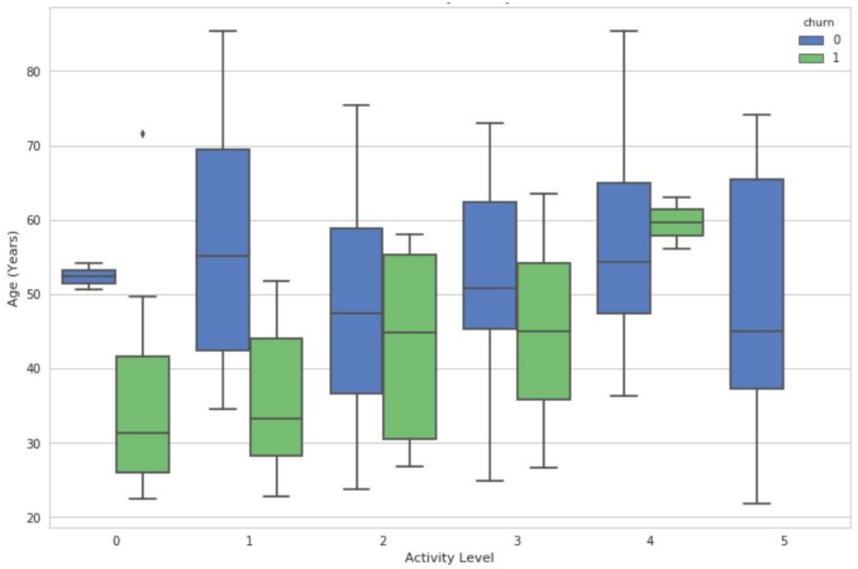
```
txn_df = load_data_from_db2(table_name='SPARKDB.SPPAYTB1', user=USERNAME, password=PASSWORD)
```

The client datas are stored using the function load_data_from_mds

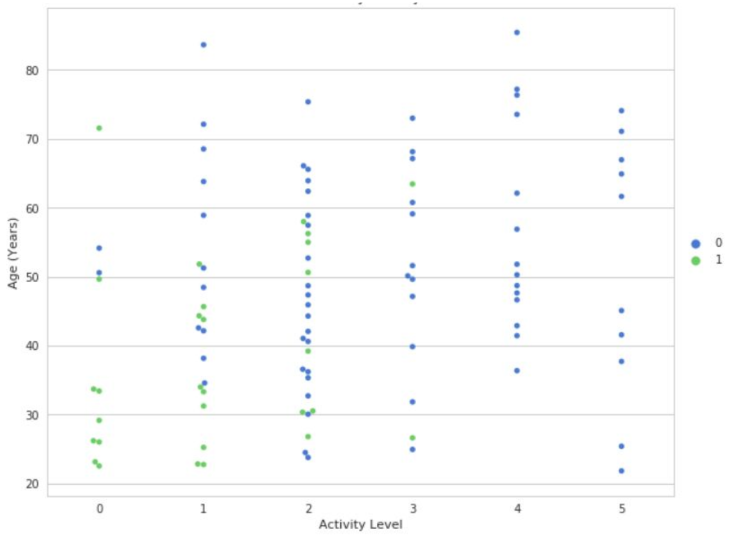
```
client_df = load_data_from_mds(vtable_name='VSAM_CLIENT', user=USERNAME, password=PASSWORD)
```

# Analytics in Mainframe zOS - IV

From the box plot and the swarm plot, we see that the mean age (in years) during which a customer might churn is a between 40-60 and during which the customer will not churn is between 30-60. The range is much wider than the one which we got from the CSV datafiles.



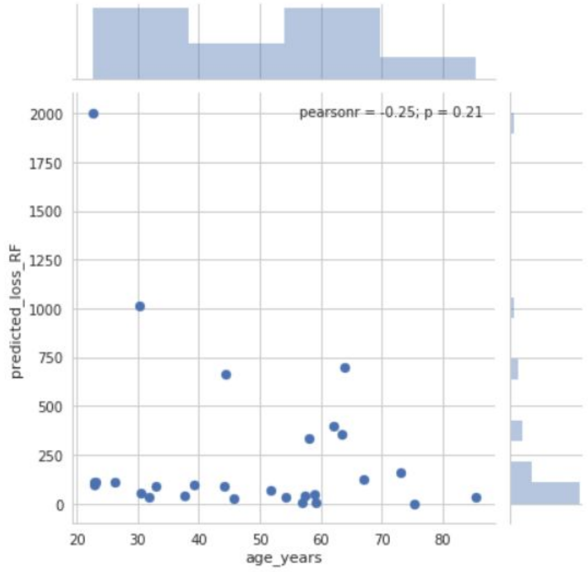
Box Plot

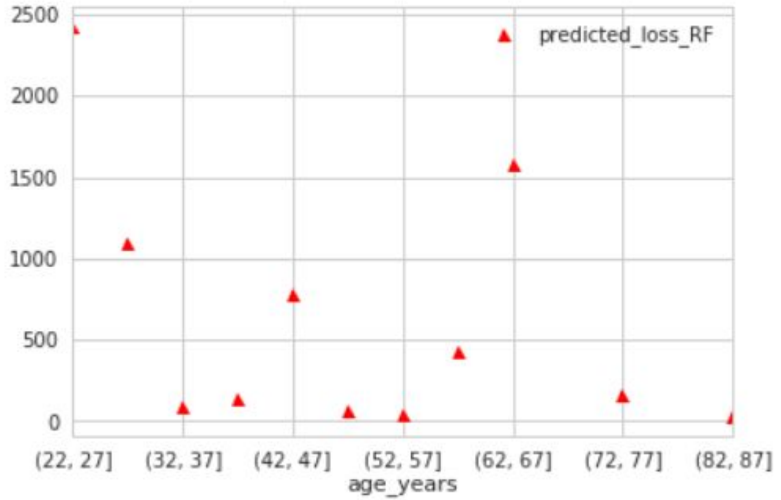


Swarm Plot

# Analytics in Mainframe zOS - V

From the graph, we see that for all the ages (in years), the predicted loss is almost between 0 - 2500.





## Conclusion

- Analytics provide business values to retain profitable customers.
- Mainframe analytics take advantage of data gravity because of which latency and data security risk is reduced.
- Insights from analytics can be plugged into transactional processing to provide customized and differentiated service to the customers.
- Last but not the least, running analytics on the Mainframe requires no special Mainframe skill other than knowing how to connect to the data tables.

## References

- https://cognitiveclass.ai/
- https://www.hackerrank.com/
- https://towardsdatascience.com/matplotlib-tutorial-learn-basics-of-pythons-powerful-plotting-library-b5d1b8f67596
- https://elitedatascience.com/python-seaborn-tutorial
- https://medium.com/clarisights/data-gravity-why-should-marketers-care-273b4d8f7a4b
- https://izoda.github.io/
- https://www.ibm.com/it-infrastructure/z/resources/trial

**Codes available on GitHub.**

# Thank You