Set up packages and Kaggle connection

In [4]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import StackingClassifier
import xgboost as xgb
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import cv2
import os
from tqdm import tqdm
import xgboost as xgb
from lightgbm import LGBMClassifier
from scipy.stats import jarque_bera
from scipy.special import logit, expit
```

```
/opt/anaconda3/envs/tf2/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarn
ing: numpy.ufunc size changed, may indicate binary incompatibility. Expected 1
92 from C header, got 216 from PyObject
  return f(*args, **kwds)
```

In [5]:

```
import seaborn as sns
import gc
import logging
import datetime
import lightgbm as lgb
from tqdm import tqdm_notebook
from sklearn.metrics import mean_squared_error
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split

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

```
/opt/anaconda3/envs/tf2/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarn
ing: numpy.ufunc size changed, may indicate binary incompatibility. Expected 1
92 from C header, got 216 from PyObject
  return f(*args, **kwds)
```

In [6]:
```
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, Precisi
from sklearn.metrics import classification_report
from sklearn import preprocessing
from sklearn import linear_model
from sklearn import metrics
from sklearn.naive_bayes import CategoricalNB,GaussianNB

import scikitplot as skplt
import tensorflow as tf
import random

from tensorflow.keras import backend as K
from tensorflow.keras.models import Sequential
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:
```
from google.colab import files
files.upload()
```

选取文件 未选择文件 Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

Out[3]:
```
{'kaggle.json': b'{"username":"jacobbraun","key":"392939438edcd0495f527be30174
d4ca"}'}
```

In [4]:
```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 /root/.kaggle/kaggle.json

!kaggle competitions download -c santander-customer-transaction-prediction
```

```
Downloading santander-customer-transaction-prediction.zip to /content
 99% 247M/250M [00:03<00:00, 80.1MB/s]
100% 250M/250M [00:03<00:00, 68.2MB/s]
```

In [5]:
```
!unzip santander-customer-transaction-prediction
```

```
Archive:  santander-customer-transaction-prediction.zip
  inflating: sample_submission.csv
  inflating: test.csv
  inflating: train.csv
```

Data Exploration

In [6]:

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

Check for missing values:

In [7]:

```
train.isna().sum().sum(), test.isna().sum().sum()
```

Out[7]:

```
(0, 0)
```

Isolate the features (drop ID_code and outcome variable)

In [8]:

```
x_train = train.drop(['ID_code', 'target'], axis=1)
x_test = train.drop(['ID_code'], axis=1)
```

Check correlation matrices in the training and testing sets

In [9]:

```
# Create a correlation matrix for the features
corr_mat = x_train.corr()
# Fill the diagonal with 0's since each variable is correlated with itself
np.fill_diagonal(corr_mat.values, 0)
# Take the absolute value of the correlations and find the max
corr_mat = abs(corr_mat)
corr_mat.max().max()
```

Out[9]:

```
0.009844361358419583
```

In [10]:

```
# Create a correlation matrix for the features
corr_mat_test = x_test.corr()
# Fill the diagonal with 0's since each variable is correlated with itself
np.fill_diagonal(corr_mat_test.values, 0)
# Take the absolute value of the correlations and find the max
corr_mat_test = abs(corr_mat)
corr_mat_test.max().max()
```

Out[10]:

```
0.009844361358419583
```

Check the distributions of each variable

In [11]:

```
x_train.hist(figsize=(155,200))
```

Out[11]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f7924932990>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792453b810>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924e89310>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924b01250>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924666ad0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79242afd50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79241f0410>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924228950>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924228990>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79241ea0d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924158bd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792411b210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79240d2810>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924086e10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f792404a450>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924001a50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792402eb90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923f7d690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923f32c90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923ef52d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923f298d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923ee1ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923ea3510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923e5ab10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923e1f150>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923dd4750>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923d89d50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923d4e390>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7923d03990>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923cb9f90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923cebb10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923c39510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923bf0a10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923c26f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923be54d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923b9b9d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923b50ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923b14410>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923acb910>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923a80e10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923a42350>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79239f9850>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7923a2ed50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923970290>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79239a7790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792395dc90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79239211d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79238d66d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792388cbd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792485aad0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924228d10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924055ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923e76110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923d12b10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923752490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792370ca90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f79236cf0d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79236836d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792363bcd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79235ff310>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79235b2910>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79235eaf10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79235af550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923565b50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923529190>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79234df790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923495d90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79234573d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792340e9d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79233bab10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7923386610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792333cc10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923302250>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79232b7850>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79232eee50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923231490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923267a90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79232290d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79231e06d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792319d790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923151c90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79231151d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79230ca6d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792307ebd0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7923042110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922ff9610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922fb0b10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922fe5fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922faa550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922f5ea50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922f15f50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922ed7490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922e8c990>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922e44e90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922e09810>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922dbfbd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922d84590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922d3c9d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7922cf4ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922cba410>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922cef6d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922ca4b50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922c5bfd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922c1e550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922bd19d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922b88dd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7924e89bd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79236838d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79234b3990>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79233541d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79231eda10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922a4a950>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7922a00f50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79229c4590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922979b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792293d1d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79228f37d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922929dd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79228ec410>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79228a2a10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922859fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792281d650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79227d2c50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922796290>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792274b890>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922702e90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f79226c44d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792267cad0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922640110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79225f5710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792262cd10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79225ee350>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79225a3950>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792255cf50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792251e590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79224d6b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922491b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79224530d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79224095d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79223bead0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f79223ebb10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922336510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792236ea10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922325f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79222e7450>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792229c950>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922254e50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922215390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79221cb890>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922182d90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79221452d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79220fb7d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79220b1cd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922077210>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f79220aa710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922060c10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922025150>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921fd9650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921f8eb50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921f3db90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921f0a590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921ebfa90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921e75f90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7923609650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7922949f50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79227e88d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79226092d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921d8e690>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7921d42c90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921d072d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921cbd8d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921c72ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921c36510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921c6cb10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921c2f150>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921be5750>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921b9dd50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921b60390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921b14990>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921acdf90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921a905d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921a45bd0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7921a09210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79219c0810>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921976e10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921939450>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79218f1a50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792191cb90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79218e8690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792189fc90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79218622d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79218198d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79217d0ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792178bd90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792174d2d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79217037d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f79216bacd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792167c210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921633710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792166cc10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792162e150>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79215e3650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921599b50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921546b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921512590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79214c9a90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7921481f90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79214424d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f79213f79d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f792142eed0>]],
      dtype=object)
```

Find the unique values in the training dataset:

In [12]:

```
# Training set
num_unique = x_train.nunique(axis=0)
prop_unique = x_train.nunique(axis=0)/200000
num_unique.mean(), prop_unique.mean()
```

Out[12]:

```
(97906.72, 0.4895336000000001)
```

Find the unique values in the test dataset:

In [13]:

```
# Testing set
num_unique = x_test.nunique(axis=0)
prop_unique = x_test.nunique(axis=0)/200000
num_unique.mean(), prop_unique.mean()
```

Out[13]:

```
(97419.63184079602, 0.4870981592039802)
```

Feature Engineering

In [14]:

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [15]:

```
# Generate new features to check if each value is unique within each original
# column

col_names = [f'var_{i}' for i in range(200)]
for column in tqdm(col_names):
  c = test[column].value_counts()
  u = c.index[c == 1]
  test[column + '_u'] = test[column].isin(u)
```

```
 48%|████▊     | 96/200 [00:01<00:02, 49.99it/s]/usr/local/lib/python3.7/dist-
packages/ipykernel_launcher.py:8: PerformanceWarning: DataFrame is highly frag
mented.  This is usually the result of calling `frame.insert` many times, whic
h has poor performance.  Consider joining all columns at once using pd.concat(
axis=1) instead.  To get a de-fragmented frame, use `newframe = frame.copy()`

100%|██████████| 200/200 [00:04<00:00, 48.08it/s]
```

In [16]:

```
# Add a column checking if at least one feature is unique for each row
test['unique'] = test[[column + '_u' for column in col_names]].any(axis=1)
```

In [17]:

```
# Separate out real test data and fake test data
test_real = test.loc[test.unique, ['ID_code'] + col_names]
test_fake = test.loc[~test.unique, ['ID_code'] + col_names]
```

In [18]:

```
len(test_real), len(test_fake)
```

Out[18]:

```
(100000, 100000)
```

In [19]:

```
# Combine all the 'real' data from the training and testing set so we can see
# there are any fakes once they're combined

realTrTe = pd.concat([train, test_real], axis = 0)
```

In [20]:

```
# Generate another set of binary features to check if each value is unique
# within each original feature column
for column in tqdm(col_names):
  c = realTrTe[column].value_counts()
  u = c.index[c == 1]
  realTrTe[column + '_unique'] = realTrTe[column].isin(u)*1
  test_fake[column + '_unique'] = 0
```

```
 48%|█████      | 97/200 [00:03<00:03, 26.44it/s]/usr/local/lib/python3.7/dist-
packages/ipykernel_launcher.py:6: PerformanceWarning: DataFrame is highly frag
mented.  This is usually the result of calling `frame.insert` many times, whic
h has poor performance.  Consider joining all columns at once using pd.concat(
axis=1) instead.  To get a de-fragmented frame, use `newframe = frame.copy()`

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: PerformanceWar
ning: DataFrame is highly fragmented.  This is usually the result of calling `
frame.insert` many times, which has poor performance.  Consider joining all co
lumns at once using pd.concat(axis=1) instead.  To get a de-fragmented frame,
use `newframe = frame.copy()`
  import sys
100%|██████████| 200/200 [00:08<00:00, 24.53it/s]
```

In [21]:

```
# From the combined set, isolate the real test values
test_real = realTrTe[realTrTe['ID_code'].str.contains('test')].copy()
test_real.drop(['target'], axis=1, inplace=True)

# Create a 'train' df with the new unique identifying features
# Create a 'test' df, combining the real and fake testing features
train = realTrTe[realTrTe['ID_code'].str.contains('train')].copy()
test = pd.concat([test_real, test_fake], axis=0)
```

Create training, validation and testing splits

In [22]:

```
df_train, df_val = train_test_split(train, random_state=123, train_size=0.85)

# Split the training data into X and Y dataframes
X = df_train.iloc[:, 2:]
Y = df_train['target']

X_val = df_val.iloc[:, 2:]
Y_val = df_val['target']



# Create the testing dataset for prediction
X_test = test.drop('ID_code', axis=1)

# scaler = StandardScaler()

# X = scaler.fit_transform(X)
# X_test = scaler.fit_transform(X)
```

Model Building

We tried 5 techniques and built 16 models. For brevity, we only listed best models for each technique. We can provide other model's code if in need.

Boosting -- LightGBM

Score: 0.89484

In [ ]:

```
lgbm_model = LGBMClassifier(**{
     'learning_rate': 0.04,
     'num_leaves': 31,
     'max_bin': 1023,
     'min_child_samples': 1000,
     'reg_alpha': 0.1,
     'reg_lambda': 0.2,
     'feature_fraction': 1.0,
     'bagging_freq': 1,
     'bagging_fraction': 0.85,
     'objective': 'binary',
     'n_jobs': -1,
     'n_estimators':400,
     'class_weight':{0:1, 1:0.1}})
```

In [ ]:

```
lgbm_model.fit(X, Y)
```

In [ ]:

```
y_pred_lgbm = lgbm_model.predict_proba(X_test)[:,1]
submission_lgbm = pd.DataFrame({"ID_code": test.iloc[:,0]})
submission_lgbm["target"] = y_pred_lgbm
submission_lgbm.to_csv("submission_lgbm.csv", index=False)
```

Boosting -- CatBoost

Score: 0.89870

In [ ]:

```
from catboost import CatBoostClassifier

cat_model = CatBoostClassifier(scale_pos_weight=1/11)
cat_model.fit(X, Y)

y_pred_cat = cat_model.predict_proba(X_test)[:,1]
submission_cat = pd.DataFrame({"ID_code": test.iloc[:,0]})
submission_cat["target"] = y_pred_cat
submission_cat.to_csv("submission_cat.csv", index=False)
```

Neural Network -- CNN

Score: 0.87297

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.3,random_
```

In [ ]:

```
N_units = 400
kernel_size=2
strides=2
cnn_model = Sequential()
cnn_model.add(Conv1D(N_units, kernel_size=kernel_size, strides=strides, paddi
                     activation='relu', input_shape=(X.shape[1], 1)))
cnn_model.add(Flatten())
cnn_model.add(Dense(1, activation = 'sigmoid'))

cnn_model.summary()
```

In [ ]:

```
epochs = 10
LR_callback = keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy', patie
EarlyStop_callback = keras.callbacks.EarlyStopping(monitor='val_accuracy', pa
my_callback=[EarlyStop_callback, LR_callback]

cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc
cnn_model.fit(X_train, y_train,
              validation_data=(X_test, y_test),
              epochs=epochs,
              verbose=2,
              callbacks = my_callback,
              )
```

In [ ]:

```
y_pred_cnn = cnn_model.predict(X_test)
```

In [ ]:

```
submission_cnn = pd.DataFrame({"ID_code": test.iloc[:,0]})
submission_cnn["target"] = y_pred_cnn
submission_cnn.to_csv("submission_cnn.csv", index=False)
# score: 0.875
```

Stacking Model

Score: 0.89434

In [ ]:

```
# initializing all the base model objects with default parameters
model_1 = xgb.XGBClassifier(max_depth=25,
                            verbose=1)

model_2 = LGBMClassifier(**{
     'learning_rate': 0.04,
     'n_estimators':200})

model_3 = RandomForestClassifier()

# putting all base model objects in one list
all_models = [('xgb', model_1), ('lgbm', model_2), ('rf', model_3)]

# create meta model
final_lr = LogisticRegression(class_weight='balanced', # Help with imbalanced
                              solver='newton-cg')

# stacked model
stack = StackingClassifier(estimators=all_models,
                           final_estimator=final_lr,
                           cv=3,
                           stack_method='predict_proba',
                           passthrough=True, # Train final model on predictio
                           verbose=1)
```

In [ ]:

```
# Fit the stacking model
stack.fit(X, Y)
```

In [ ]:

```
# Generate predictions from the testing dataset
y_pred = stack.predict_proba(X_test)[:,1]
```

In [ ]:

```
# Create the submission file
submission = pd.DataFrame({"ID_code": test.iloc[:,0]})
submission["target"] = y_pred
submission.to_csv("submission.csv", index=False)
```

Ensemble 200 Models

Score: 0.88764

- each time use one original feature and corresponding frequency feature to build a model
- then use logit transfer the prediction of probability got from one model and sum up these result from 200 models
- then get mean of these result from 200 models and use exp to transfer into probability

In [ ]:

```
#features = [x for x in X_train.columns if x.startswith("var")]
features = X.columns[0:200].to_list()

pred = 0
for var in features:
    model = lgb.LGBMClassifier(**{'learning_rate': 0.05,
                                  'max_bin': 165,
                                  'max_depth': 5,
                                  'min_child_samples': 150,
                                  'min_child_weight': 0.1,
                                  'min_split_gain': 0.0018,
                                  'n_estimators': 41,
                                  'num_leaves': 6,
                                  'reg_alpha': 2.0,
                                  'reg_lambda': 2.54,
                                  'objective': 'binary',
                                  'n_jobs': -1})
    var_count_name = var + '_unique'
    model = model.fit(np.hstack([X[var].values.reshape(-1, 1),
                           X[var_count_name].values.reshape(-1, 1)]), Y.values)
    pred += logit(model.predict_proba(np.hstack([X_test[var].values.reshape(-
                      X_test[var_count_name].values.reshape(-1, 1)]))[:, 1])

#pd.DataFrame({"ID_code": test_id, "target": pred}).to_csv("submission3.csv",
b = pd.DataFrame({"ID_code": test['ID_code'], "target": pred})
```

In [ ]:

```
b['target'] = np.exp(b['target']/200)
```

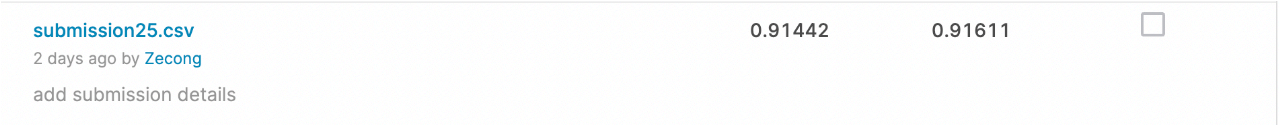
In [ ]:

```
b.to_csv("submission8.csv", index = False)
```

Best Model

Ensemble 200 Stacking Models

Score: 0.91611



submission25.csv 0.91442 0.91611 □
2 days ago by Zecong
add submission details

In [ ]:

```
#features = [x for x in X_train.columns if x.startswith("var")]
features = X.columns[0:200].to_list()

pred = 0
for var in features:
    print(var)
    model_1 = CatBoostClassifier(scale_pos_weight=1/11)
    #model_1 = LogisticRegression(class_weight='balanced',solver='newton-cg',

    model_2 = xgb.XGBClassifier()
    #model_3 = RandomForestClassifier(n_estimators=250, min_samples_split=20)
    model_4 = LGBMClassifier()

    var_count_name = var + '_unique'
    all_models = [('cat', model_1), ('xgb', model_2), ('lgbm', model_4)]
    #all_models = [('lr', model_1), ('xgb', model_2), ('rf', model_3), ('lgbm
     # create meta model
    final_lr = LogisticRegression(class_weight='balanced', solver='newton-cg'

    # stacked model
    stack = StackingClassifier(estimators=all_models,
                           final_estimator=final_lr,
                           cv=None,
                           stack_method='predict_proba',
                           n_jobs=-1,
                           passthrough=True, # Train final model on predictio
                           verbose=1)

    model = stack.fit(np.hstack([X[var].values.reshape(-1, 1),
                      X[var_count_name].values.reshape(-1, 1)]), Y.values)
    #print(model.predict_proba(np.hstack([X_test[var].values.reshape(-1, 1),X

    val_pred += model.predict_proba(np.hstack([X_val[var].values.reshape(-1,
                  X_val[var_count_name].values.reshape(-1, 1)]))[:, 1]
    val_pred_prob = val_pred/200
    pred += model.predict_proba(np.hstack([X_test[var].values.reshape(-1, 1),
                  X_test[var_count_name].values.reshape(-1, 1)]))[:, 1]
    #pred_prob = pred/200
```

In [ ]:

```
b = pd.DataFrame({"ID_code": test['ID_code'], "target": pred/200})
```

In [ ]:

```
b.to_csv("submission25.csv", index = False)
```

Business Value Analysis

draw ROC curve on validation dataset

In [23]:

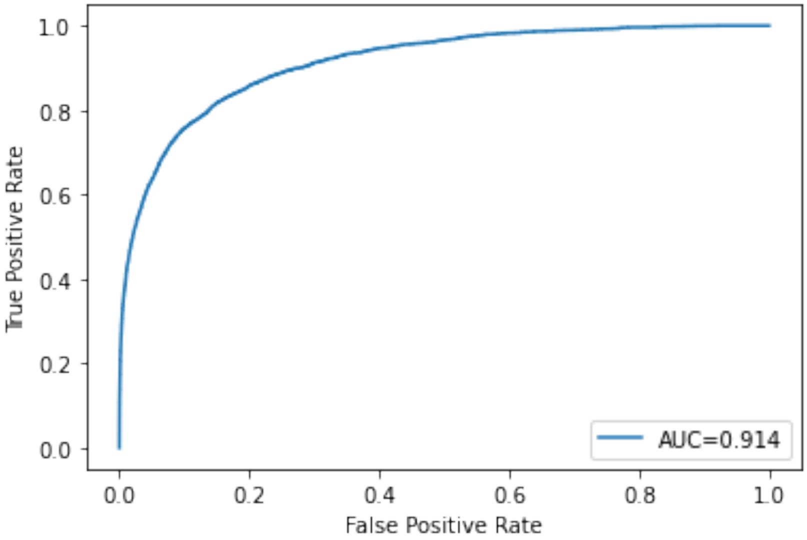
```
fpr, tpr, thresholds = metrics.roc_curve(Y_val, val_pred_prob, pos_label=1)
auc_score = round(metrics.auc(fpr, tpr), 4)
print('AUC score:', auc_score)
```

```
AUC score: 0.9138
```

In [16]:

```
auc = metrics.roc_auc_score(Y_val, val_pred_prob)

#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(round(auc,3)))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
```



In [102…

```
cost_matrix = np.array([[10, -100], [-20, 100]])
```

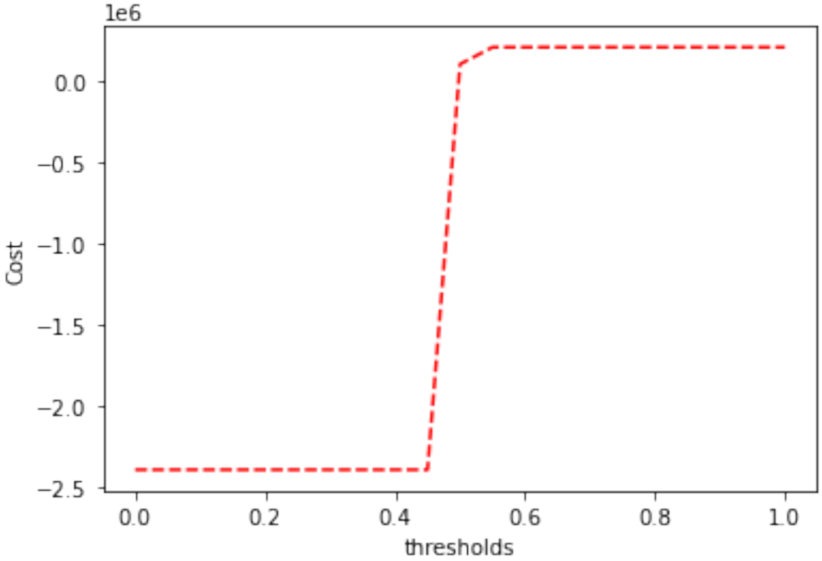
In [103…

```
Cost_List=np.linspace(0, 1.0, num=21)
thresholds = np.linspace(0, 1.0, num=21)
index=0

for t in thresholds:
    predict_thre = np.where(val_pred_prob > t, 1, 0)  ##prediction based on t
    clf_matrix = confusion_matrix(Y_val, predict_thre)
    Cost_List[index] = clf_matrix[0][0]*cost_matrix[0][0]+clf_matrix[0][1]*co
    index+=1
    #print(predict_thre)
#y_pred = np.where(y_pred_prob>=0.5, 1, 0)
plt.figure(1)
plt.plot(thresholds, Cost_List, 'r--')
plt.xlabel("thresholds")
plt.ylabel("Cost")
plt.show()

cost_tb = pd.DataFrame({'threshold':thresholds, 'cost':Cost_List}).sort_value
min_thres = cost_tb['threshold'].iloc[0]
min_cost = cost_tb['cost'].iloc[0]

print('Minimal cost is: {} with using threshold {}'.format(min_cost, min_thre
```



```
Minimal cost is: -2394800.0 with using threshold 0.0
```

In [ ]: