Team 2

**Members:** Amit Singh(06), Anuradha Kolarkar(07), Anushka Samnerkar(08), Arbind Kumar(09), Avdhut Gadhe(10).

IEEE-CIS Fraud Detection

**Problem Statement:**

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.  
  
Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.  
  
While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful**.**

**Evlaluation:**

Submission File   
For each SK\_ID\_CURR in the test set, you must predict a probability for the TARGET variable. The file should contain a header and have the following format:   
  
SK\_ID\_CURR,TARGET   
100001,0.1   
100005,0.9   
100013,0.2   
etc.

**Aim:**

Our aim was to train machine learning model to predict the probability that an online transaction was fraudulent.

**About Dataset**

The data is broken into two files identity and transaction, which are joined by TransactionID each for train and test. Thus we have 4 files — train transactions, train identity, test transactions, and test identity. It is important to note that all transaction does not have corresponding identity information.

**Transaction data**

Transaction id: Id related to the transaction

TransactionDT: timedelta from a given reference datetime (not an actual timestamp)

TransactionAMT: transaction payment amount in USD

ProductCD [Categorical]: product code(the product for each transaction)

card1–6 [Categorical]: payment card related information like card type, country etc

addr1, addr2 [Categorical]: address information

dist1,dist2: some distance information

P\_emaildomain [Categorical]: email domain of purchaser.

R\_emaildomain [Categorical]: email domain of the recipient.

C1-C14: counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.

D1-D15: time delta, such as days between the previous transactions, etc.

M1-M9 [Categorical]: match, such as names on card and address, etc.

Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations.

**Identity data**

Transaction id: Id related to the transaction

DeviceType [Categorical]: Type of device used for the transaction

DeviceInfo [Categorical]: More information about device used

id 1–38 [Categorical+numeric]:

network connection information,browser information etc (id 12–38 are categorical information)

Note: Actually we don't have access to exact information about columns. Mostly it is because of security purposes as we are handling transaction data.

**Business Objective**

The main business objective of finding fraud transactions can be following:If a fraud transaction is found out, the company should immediately block that card.We should be able to predict the probability of fraud transactionWe should not predict fraud transactions as nonfraud. Also the vice versa. So precision and recall should be taken care of.

**Agenda**

Importing Necessary Libraries

Data Loading, Understanding, and Cleaning

Data Preprocessing

ML Modeling

Prediction Submission

**Importing Necessary Libraries**

We imported various libraries for various purposes like splitting, merging, treating imbalanced data and encoding also for plotting different graphs.

The libraries we imported are as follows:

Pandas

Numpy

Matplotlib

Seaborn

Sklearn

SimpleImputer

StandardScaler

IMblearn

SMOTE

RandomForestClassifier

**Data Loading, Understanding, and Cleaning**

Since the data is big in size, we will use function to reduce its memory for fast processing and consuming less storage. For that purpose we created a function named “Helper function”.

After loading and optimizing the data it got reduced

drastically.

**# loading train\_transaction data**

Memory usage of dataframe is 1775.15 MB

Memory usage after optimization is: 487.16 MB

Decreased by 72.6%

**# Loading train\_identity data**

Memory usage of dataframe is 45.12 MB

Memory usage after optimization is: 10.00 MB

Decreased by 77.8%

**# Merging transaction and identity train data**

After merging the row and columns are:

(590540, 434)

**# Loading test data**

For Transaction:

Rows,columns -> (506691, 393)

Memory usage of dataframe is 1519.24 MB

Memory usage after optimization is: 425.24 MB

Decreased by 72.0%

For Identity:

Rows,columns -> (144233, 41)

Memory usage of dataframe is 45.12 MB

Memory usage after optimization is: 10.00 MB

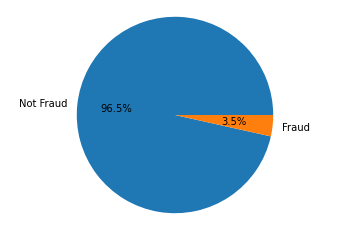
Decreased by 77.8%

**# Duplicates check in train data**

0

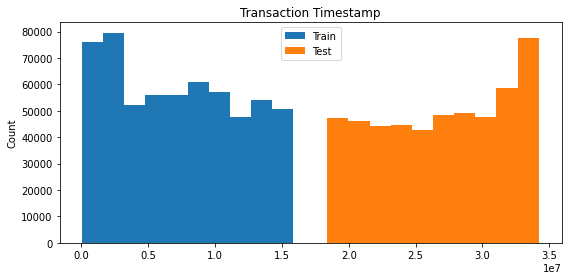
So, we have total 434 columns including the dependent variable as 'isFraud'. The training data consists of 5,90,540 samples and test data has 5,06,691 records. We also checked that there are no duplicate records. Let us investigate distribution of the dependent variable.

**# Class imbalance check**



As one can expect, this is a class imbalance problem. We will apply SMOTE (Synthetic Minority Over-sampling Technique) to deal with class imbalance in later steps. Let us understand the distribution of the timestamp column.

**# Timestamp of train and test data**



We can notice that the timestamp of the test data is ahead of the timestamp of the train data. Therefore, while training machine learning model, we need to perform time-based splitting to create training and validation sets. Let us deal with the missing values first.  
There are considerable number of columns with high missing values. We'll use only those columns that has at least 80% data which leaves 20% to the missing values that can be fillied.

**# Missing values check**

(1097231, 432)

**# Dependent variable**

(590540,)

**# Dropping columns with more than 20% missing values**

(1097231, 180)

We are left with 180 columns out of 432 after removing features with more than 20% missing values. We also have removed 'TransactionID' column as it does not hold any importance in the prediction. Let us now fill all the missing values. For numerical columns, we will use median value and for categorical column, we will use the most frequent category to fill the missing values.

**# Filtering numerical data**

(1097231, 176)

**# Filtering categorical data**

(1097231, 4)

**# Filling missing values by median for numerical columns**

(1097231, 176)

**# Filling missing values by most frequent value for categorical columns**

(1097231, 4)

**# Verifying missing values**

Total missing values: 0

(1097231, 180)

Data Preprocessing

Now, we have dealt with the missing values. Let us perform categorical encoding.

**# One-hot encoding**

(1097231, 245)

**# Separating train and test data**

(590540, 245)

(506691, 245)

**# Time-based train validation splitting with 20% data in validation set**

X\_train.shape, X\_val.shape, y\_train.shape, y\_val.shape

(472432, 245) (118108, 245) (472432,) (118108,)

**# Standardization**

Standardization is basically scaling the whole range of dataset to the range of 0 to 1.

**# Class imbalance check**

0 455833

1 16599

dtype: int64

**# Applying SMOTE to deal with the class imbalance by oversampling**

(911666, 245) (911666,)

Out[17]:

0 455833

1 455833

dtype: int64

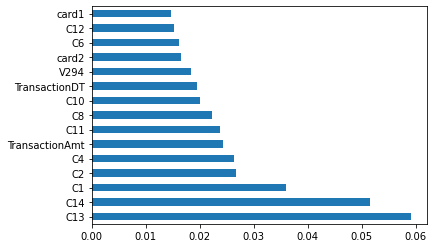
ML Modeling

I tried PCA to reduce dimension but it resulted in relatively poor performance so we will go with this dimensions only.  
After hyperparameter tuning, following parameters are selected.

# Random Forest Classifier

Validation AUC=0.8880198476630929

**# Feature importances**



Prediction Submission

In [20]:

linkcode

**# Predicting for the test data**

We used Random Forrest Classifier to predict the probability of the fraud and mentioned it in the ‘**IsFraud**’ column.

**Conclusion:**

As expected we can see that class is heavily imbalanced. Here 96.5% of transactions are not fraud and the rest 3.5% of transactions are fraudulent.

We also performed PCA for dimension decomposition but It resulted in relatively poor performance so we did hyper-parameter tuning and selected few dimensions and applied Random Forest Classifier. We chose the Area under the ROC curve (AUC) which came out to be **88%** as a metric for our ML problem and calculated the probability.

These are few rows which shows the fraud probability along with its associated Transaction ID.

TransactionID isFraud0 3663549 0.0480821 3663550 0.0334862 3663551 0.2002843 3663552 0.0699694 3663553 0.185755