# Approach document - India ML Hiring Hackathon 2019

## Problem - Loan Delinquency Prediction

Given the information like mortgage details, borrowers related details and payment details, our objective is to identify the delinquency status of loans for the next month given the delinquency status for the previous 12 months (in number of months)

So this is basically a Binary Classification problem to be solved using Machine Learning

## Problem solving Approach:

This is my first participation in a ML hackathon. While I have tried following a systematic approach to solve the problem, given the limited time, I had available to work – my approach was to quickly get a decent solution up and running. Obviously the solution is not optimal in terms of performance metric and there is scope for improvement.

The Jupyter notebook contains enough comments to explain the logic behind the code.

1. Exploratory Data Analysis
   - Identify the Feature Columns and Target Columns
   - Check if the target class distribution is imbalanced
   - Check Features for categories and data types
     1. Numerical
     2. Categorical
     3. Temporal
   - Check Features for Cardinality and missing values
   - Check Features for data distribution and outliers
   - Check for correlation between features and target

2. Perform Feature Engineering
   - Temporal feature
   - One hot encoding Categorical Features

3. Model Training and Validation
   - Split training and validation data
   - Trained and Evaluated different tree based ensemble classifiers
   - Finally selected the best estimator model based on results of hyper tuning.

4. Submission to AV
   - Predict delinquency on the Test Data

**Concluding Remarks:**

- The biggest improvement in scores come with a good Feature engineering and selection.
    - Created a temporal feature Elapsed Days for payment of first payment and new Average credit score – in case of more than 1 borrower.
    - I think we could get some performance improvement by dropping and or/consolidating some other features
- Hyper-tuning helps to improve the scores – but it will not change scores dramatically.
- Oversampling the dataset to handle the class imbalance – improved the performance at least during the local validation. However unfortunately did not implement it since did not get time to thoroughly validate the performance with oversampling.
- Spot checked Decision Tree and ensemble algorithms like Random Forest, Adaboost and XGBoost for performance and selected XGBoost based on it's superior score in cross-validation and after quickly trying out oversampling the minority class. As mentioned above – I did not include oversampling to handle imbalanced data.