**PREDICTING THE LIKELYHOOD**

**OF E-SIGNING**

**OF LOAN**

**BASED ON FINANCIAL HISTORY**

**Presented by :-** Pranjal Agarwal

# ABSTRACT

# The company seeks to leverage this model to identify fewer quality applicants(e.g. those who are not responding to the onboarding process) and experiment with giving them different onboarding screen.

The official application begins with the lead visiting into the website after they opted to acquire it. Here, the applicant starts with the onboarding process to apply for a loan. The user begins to provide more financial information by going over every screen of the onboarding process. The first phase ends with the applicant providing his/her signature indicating all of the given information is correct.

Any of the following screens, in which the applicants are approved/denied and given the terms of the loan, is dependent on the company, not the applicant. Therefore the effectiveness of the onboarding is measured up to the moment the applicant stops having control of the application process.

We will use various machine learning classification algorithms in this project such as SVM, Decision Tree and Random Forest to achieve our objective optimally. We will also optimize our model according to the data provided.

# CONTENT

* + - * Introduction

**1.1** [Problem Statement](#_TOC_250021)

1.2 Project Description

1.3 [Objective](#_TOC_250020)

* 1. [1.4 System Requirements](#_TOC_250019)
  2. 1.5 [Libraries](#_TOC_250018)

1.6 [Scikit Learn](#_TOC_250017)

1. Dataset
   1. 2.1 [Data Visualization](#_TOC_250016)
   2. 2.2 EDA
      1. [Histogram](#_TOC_250015)
      2. Correlation between response variable
   3. 2.3 [Data Cleaning](#_TOC_250014)
      1. [One Hot Encoding](#_TOC_250013)
      2. [Splitting the data](#_TOC_250012)
      3. [Feature scaling](#_TOC_250011)
2. [Model Selection](#_TOC_250010)
   1. 3.1 [Machine Learning](#_TOC_250009)
      1. 3.2 [Testing the model](#_TOC_250008)
   2. Linear Regression
   3. [SVM (linear)](#_TOC_250007)
   4. [SVM (rbf)](#_TOC_250006)
   5. [Decision Tree](#_TOC_250005)
   6. [Random Forest](#_TOC_250004)
3. [Optimizing the model](#_TOC_250003)
   1. 4.1 K-fold
   2. 4.2 [Grid Search](#_TOC_250002)
   3. 4.3 Confusion Matrix
   4. 4.4 Final Result
4. [Conclusion](#_TOC_250001)
5. Future Scope
6. Bibliography

# 1. INTRODUCTION

# 1.1.1 Problem Statement :

To develop a model to predict 'quality' applicants, in this case, study quality applicants are those who reach a minimum threshold part of the loan application process.

# 1.1.1 Project Description :

In this project, we are going to use various type of machine learning classification algorithms to predict 'quality' applicants, in this case, we study quality applicants are those who reach a minimum threshold part of the loan application process.

# Objective :

The objective that we are going for and achieve at the completion of this project is :

* + - * Use Case / Business Case

Step one is actually understanding the business or use case with the desired outcome. Only by understanding the final objective we can build a model that is actually of use. In our case the objective is predicting the liklelihood of e-signing of loan based on financial history.

* + - * Data collection & cleaning

With understanding the context it is possible to identify the right data sources, cleansing the data sets and preparing for feature selection or engineering. The predicting model is only as good as the data source.

* + - * Feature selection & engineering

With the third step we decide which features we want to include in our model and prepare the cleansed data to be used for the machine learning algorithm.

* + - * Modelling

With the prepared data we are ready to feed our model. But to make good predictions, we firstly need to find the right model (selection) and secondly need to evaluate that the algorithm actually works.

* + - * Insights and Actions

Last but not least we have to evaluate and interpret the outcomes.

# System Requirements :

|  |  |
| --- | --- |
| Operating System | Windows 10 / Ubuntu 15 or above |
| Processor | Intel i3 7th Gen or above at 1.8Ghz |
| RAM | 4 GB DDR3 or above |
| GPU | Nvidia GTX 2GB + |
| Software Required | Python 3.7+ , Anaconda |

# Libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sn

import random

import time

The various libraries that we used to implement our model includes :

* + - * **pandas**: For data retrieval, manipulation, storage and analysis.
      * **matplotlib** : Provides an object-oriented API for embedding plots into applications.
      * **seaborn** : Data visualization library based on matplotlib, used for statistical data.
      * **time** : Provides various time-related functions.

# Sci-kit learn

# 

scikit-learn is an open source Python library that implements a range of machine learning, pre-processing, cross-validation and visualization algorithms using a unified interface. Most of the functions that we are going to use here belongs to the library scikit-learn. We use different sub libraries to call various types of classes to perform all the steps that includes , data cleaning, data splitting, model fitting , optimization processes.

Important features of scikit-learn:

* + - * Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
      * Accessible to everybody and reusable in various contexts.
      * Built on the top of NumPy, SciPy, and matplotlib.
      * Open source, commercially usable – BSD license.

# 1. THE DATASET

# Data Visualization

One of the most valuable assets a company has is data. The raw dataset contains more than 300000 entries. All entries have several features.

To better understand the data we will first load it into pandas and explore it with the help of some very basic commands.

* + - * **read\_csv()** – Read a comma-separated values (.csv) file into a data frame.
      * **head( )** - Displays first 5 rows of data, unless a number is specified in parameters, in which case, that many number of rows from start is displayed.
      * **columns** - Displays all the column heads of a given data set.
      * **describe( )** - is used to view some basic statistical details like percentile, mean, std etc. of a data frame.
      * **isna( )** - Returns a Boolean value true if missing values are present, otherwise returns false.
      * **any( )** - Accepts iterable (list, tuple, dictionary etc.) as an argument and return true if any of the element in iterable is true.
      * **drop(columns = [ ])** - Deletes the column specified by column header name within the [ ] brackets.
      * **figure ( )** - Create a new figure.
      * **suptitle( )** - Add a centered title name to the figure.
      * **subplot( )** - Add subplot to the current figure.
      * **gca( )** - Stands for Get Column Access, iterates through all the values in a column.
      * **set\_visible( )** - Set the artists’ visibility.
      * **set\_title( )** - Sets a string value as the title of some data visualization instance.
      * **iloc[ ]** - Stands for index location. Used to select some rows of data from a column.
      * **pie( )** - Produces a pie chart with the define specifications.
      * **corrwith( )** - Used to compute pairwise correlation between rows or columns of two data frame objects.
      * **set( )** - used to convert any of the iterable to the distinct element and sorted sequence of iterable elements, commonly called Set.
      * **corr( )** - used to find the pairwise correlation of all columns in the data frame.
      * **triu\_indices\_from( )** - Return the indices for the upper-triangle of two-dimensional array.
      * **diverging\_palette( )** - Make a diverging palette between two colors.
      * **heatmap( )** - Plot rectangular data as a color-encoded matrix.

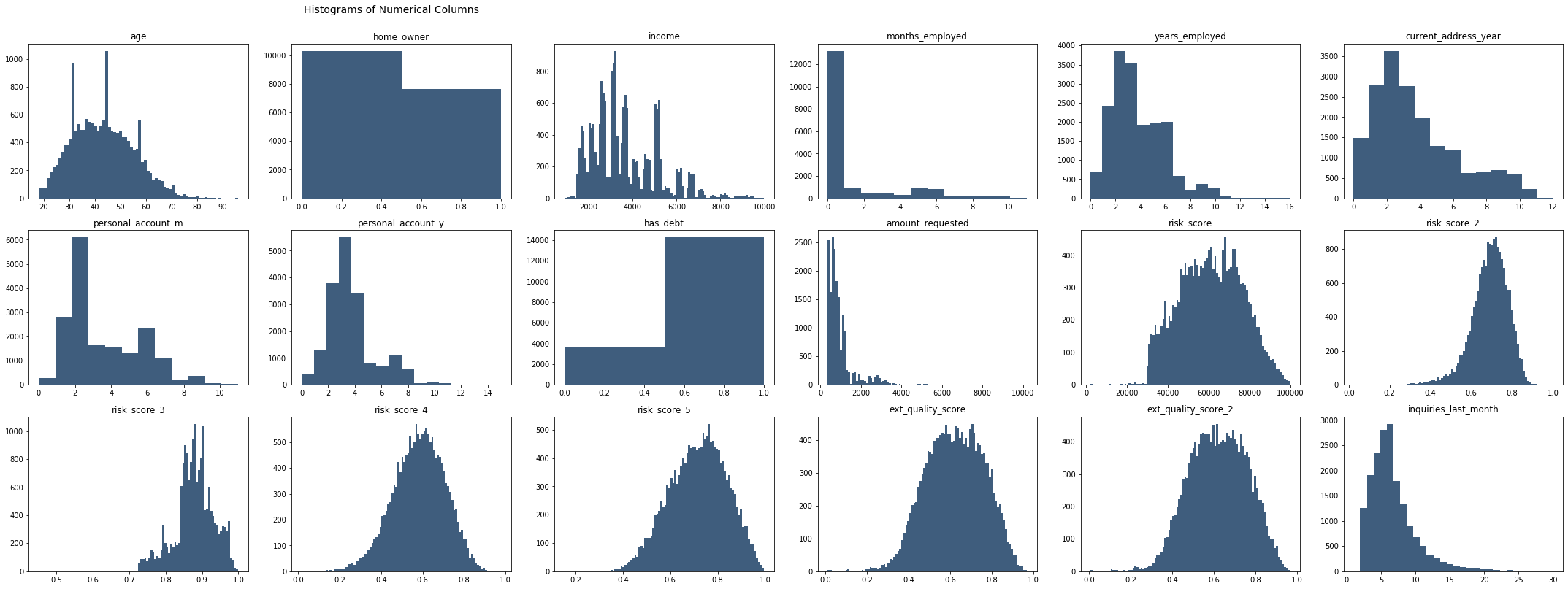
# EDA - Exploratory Data Analysis

It is a way of visualizing, summarizing and interpreting the information that is hidden in rows and column format. EDA is one of the crucial step in data science that allows us to achieve certain insights and statistical measure that is essential for the business continuity, stockholders and data scientists. It performs to define and refine our important features variable selection, that will be used in our model.

We dropped 3 columns –'entry\_id', 'pay\_schedule', 'e\_signed' as they doesn’t carry much value in our project and also will improve our model and training time.

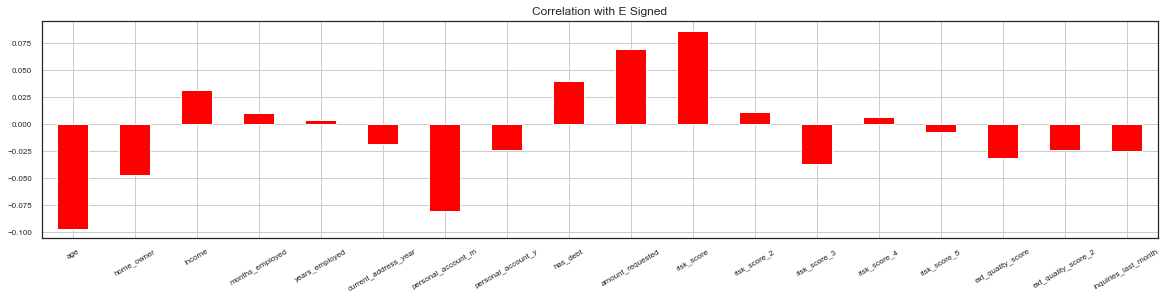
# Histogram:

Studying the histogram of necessary data :



# Correlation between Responses

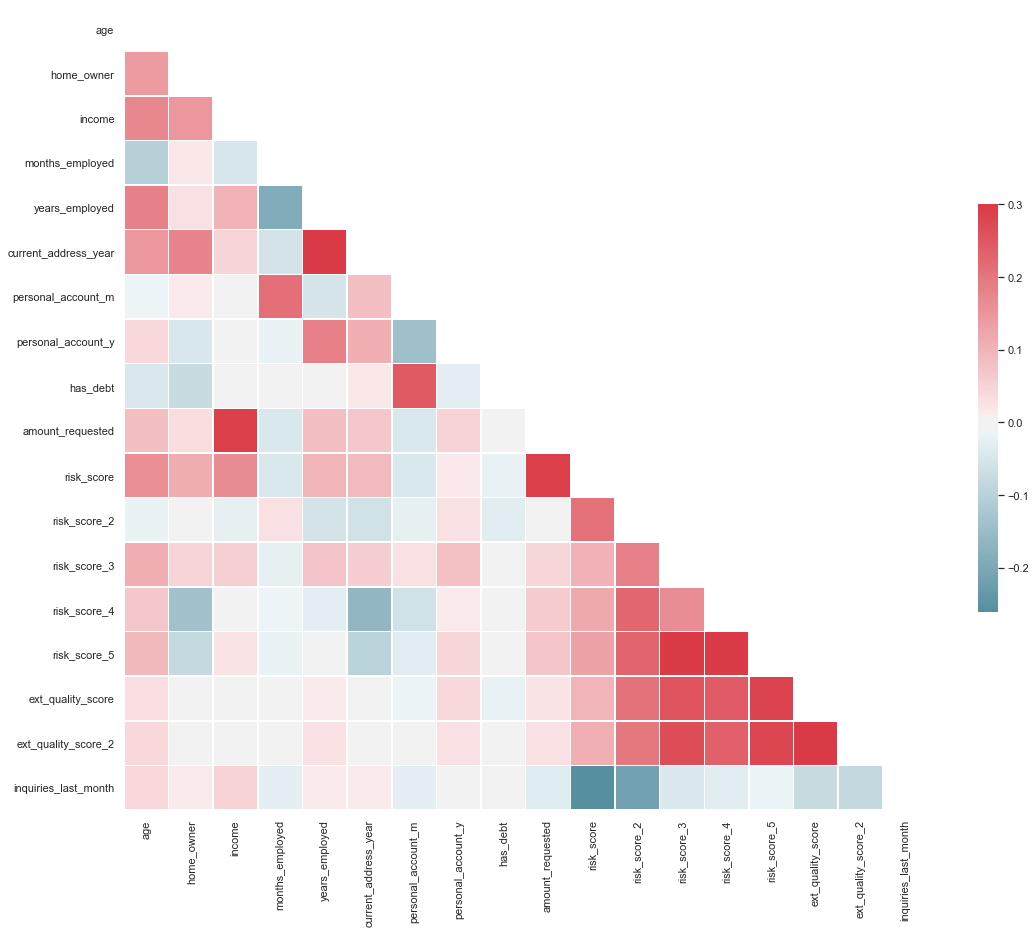
After analysing the we have histogram better understanding of how each data is related with e-signed. After that to get even more precise knowledge of some of the features we use :



The X-Axis contains all the features and how strong their correlation is with e-signed. Y-Axis represents the degree of correlation. Bars going upwards represent positive correlation and bars going downwards represent negative correlation.

A heat map is a two-dimensional representation of data in which values are represented by colors. A simple heat map provides an immediate visual summary of information. More elaborate heat maps allow the viewer to understand complex data sets.

Finally, a heat map is produced of all the features, so that the correlation of each feature to every other feature through the entire data set is represented in a single visual representation. We mask the upper half of the heat map as it is redundant due to being a mirror of the lower half. We do not include customer ID and customer e-signed value , as user ID is not an affecting feature and the e-signed is the dependent variable.



Both the axes have all the attributes set along them. A darker shade of red on a cell represents a stronger positive correlation between the two attributes intersecting on that cell, and a darker shade of blue on a cell represents a stronger negative correlation.

# Data Cleaning

In the context of data science and machine learning, data cleaning means filtering and modifying your data such that it is easier to explore, understand, and model. Filtering out the parts you don’t want or need so that you don’t need to look at or process them. Modifying the parts you do need but aren’t in the format you need them to be in so that you can properly use them.

# One Hot Encoding

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

We use One Hot Encoding in “pay\_schedule” as they contain categorical values. We use the function : *get\_dummies()* from the pandas library.

We combine 'personal\_account\_m' and 'personal\_account\_y' under a single column named 'personal\_account\_months' and drop 'personal\_account\_m' and 'personal\_account\_y' as they are no longer needed.

# Splitting the data

The basic idea is to divide the dataset T into two subsets – one subset is used for training while the other subset is left out and the performance of the final model is evaluated on it. The main purpose of cross- validation is to achieve a stable and confident estimate of the model performance.

The optimal ratio of splitting the data is 8:2 so we are keeping the 80% of data into the training set and the rest 20% into the testing set at *random state = 0*, so that we don’t have any randomization in our data

We save the training set into X\_train and y\_train. We save the test set into X\_test and y\_test.

# Feature Scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units.

We use the class *StandardScalar* from the library scikit learn to implement feature scaling.

The function *fit\_transform()* is then used to transform all the data in a fixed range to fit it into our model. We also use the function : *Dataframe()* to preserve the column names.

# MODEL SELECTION

# Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

# Testing the model :

We use various functions to test our model on how much time it is taking to train, how accurately it can predict the outcomes on new data. For this we use various functions that are :

**accuracy\_score( ) -** In multilabel classification, this function computes subset

**precision\_score( ) -** Compute the precision. The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives.

**recall\_score( )** - Compute the recall. The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives.

**f1\_score( ) -** Compute the F1 score, also known as balanced F-score or F-measure. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is: F1 = 2 \* (precision \* recall) / (precision + recall)

**time( )** - Returns the number of seconds passed since epoch

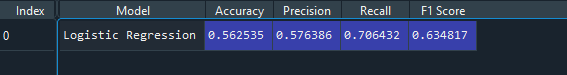
# Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

We use the Logistic Regression algorithm and fit the data into this model and check the results. We use the class *LogisticRegression* from the library *scikit* learn.



The results we achieve are :



We achieve accuracy of 56.25% which is not really good. Our next objective is to use other algorithms and compare the result.

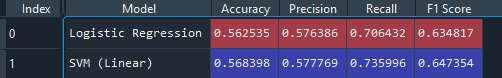
# SVM (linear)

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

We use the class *SVC* from the library scikit learn with *kernel = “linear”.*

**

The results we achieve are :



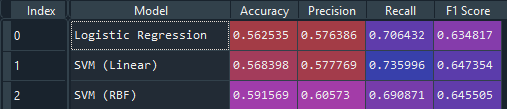
We can observe that SVM (linear) is giving better results than Logistic Regression, we can change the parameters of SVM and try to fir our data into it once again.

# SVM (rbf)

In machine learning, the radial basis function kernel, or *RBF kernel*, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification.

We use the class *SVC* from the library *scikit* learn with *kernel = “linear”.*

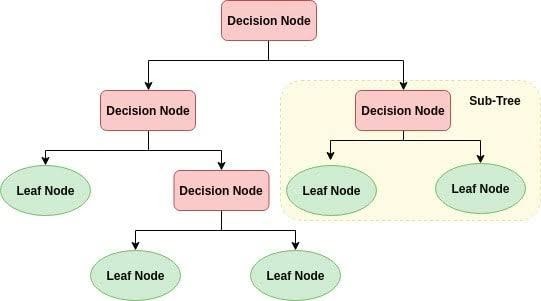
The results we achieve are :



Here when we use “rbf” as parameter in SVM we get much better results than both the previous models.

# Decision Tree

A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter.



We use the class *DecisionTreeClassifier* from the library *scikit* learn. The results we achieve are :

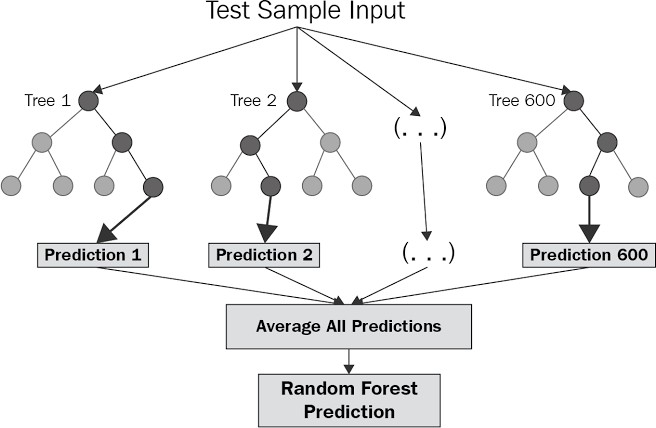




# Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

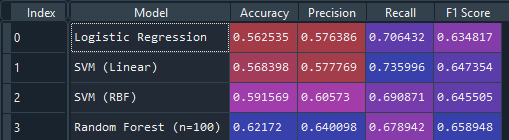
**Ensemble Learning :** *Random forests or random decision forests* are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.



We use the class *RandomForestClassifier* from *sklearn.ensemble*. We are running the algorithm at

*n\_estimators* = 100 , this means that it will take average of 100 predictions from the model.

The results we achieve by using Random Forest Classifier are :



Random Forest being one the best models in classification algorithms gives immensely better results than all the previous models that is 62.17%.

Apart from accuracy Random Forest is giving better results in *precision, recall and f1 score* too.

# OPTIMIZING THE MODEL

Optimization is the most essential ingredient in the recipe of machine learning algorithms. It starts with defining some kind of loss function/cost function and ends with minimizing them using one or the other optimization routine. The choice of optimization algorithm can make a difference between getting a good accuracy in hours or days. The applications of optimization are limitless and is widely researched topic in industry as well as academia.

# K-Fold Validation

K-Fold CV is where a given data set is split into a K number of sections/folds where each fold is used as a testing set at some point. Let’s take the scenario of 10-Fold cross validation(K=10). Here, the data set is split into 10 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 10 folds have been used as the testing set.

**## K-fold Cross Validation**

**from sklearn.model\_selection import cross\_val\_score**

**accuracies = cross\_val\_score(estimator = classifier, X= X\_train, y = y\_train,**

**cv = 10)**

**print("Random Forest Classifier Accuracy: %0.2f (+/- %0.2f)" % (accuracies.mean(), accuracies.std() \* 2))**

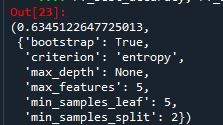
# Grid Search

A model hyperparameter is a characteristic of a model that is external to the model and whose value cannot be estimated from data. The value of the hyperparameter has to be set before the learning process begins. For example, *c in Support Vector Machines, k in k-Nearest Neighbours, the number of hidden layers in Neural Networks*.

In contrast, a parameter is an internal characteristic of the model and its value can be estimated from data. Example, *beta coefficients of linear/logistic regression or support vectors in Support Vector Machines*, *n\_estimators in Random Forest*.

Grid-search is used to find the optimal hyperparameters of a model which results in the most ‘accurate’ predictions.

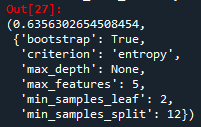
**ROUND 1:(Entropy)**



After running Grid Search for round 1 which took 3029 seconds to give the results, now we go into round 2 of Grid Search by giving even more suitable parameters from the output.

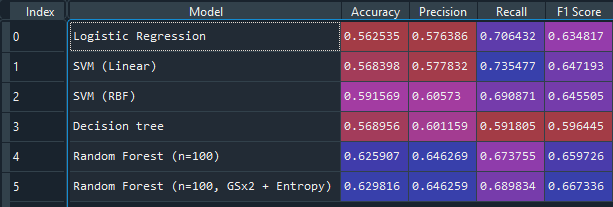
**ROUND 2: (Entropy)**

After running the grid search for the 2nd time we get the results :



It took 1004 seconds.

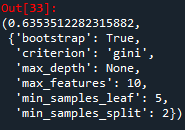
After running the grid search for the 2nd time we get the results :



We get an accuracy of 62.98% instead of 62.59% which implies that we have improved our model a bit. Though not by much but this improvement can come in handy for improving the overall result of our model.

**ROUND 3: (Gin)**

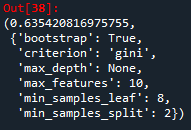
The results that we get are:



It took 1948 seconds.

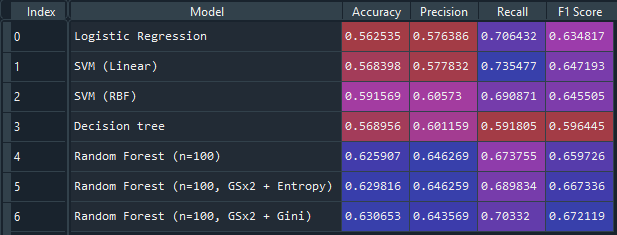
**ROUND 4: (Gin)**

Now using the result of round 3 in round 4 we get the following result:



It took 1232 seconds to execute.

After running the grid search for the 4th time we get the results :



**Initially the accuracy was 62.59% but now after using the hyperparameters produce by the fourth round we are**

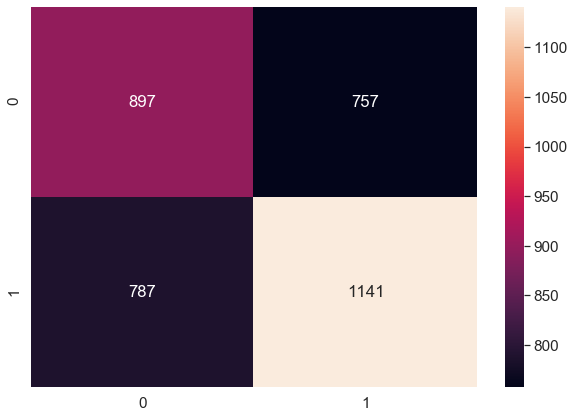
**getting an accuracy of 63.06% which implies that we have improved our model a lot as in a data of 300000**

**enteries an improvement of 1% means an improvement of 3000 results.**

**Confusion Matrix:**

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

:



True Positive predictions are- 879

True Negative predictions are-757

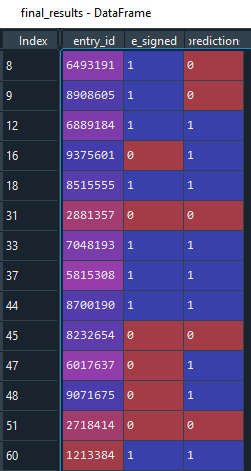
False Positive predictions are-787

False Negative predictions are-1141

Total correct predictions are: 2038

Total incorrect predictions are:1544

**FINAL RESULTS:**

****

# CONCLUSION

Our model has given us an accuracy of 63.06%. With this, we have an algorithm that can help predict whether or not a user will complete the E-signing step of the loan application. One way to leverage this model is to target those predicted to not reach the e-sign phase with customized onboarding. This means that when the lead arrives from the p2p, they may receive a different onboarding experience based on how likely they are to finish the general onboarding process. This can help the company minimize how many people drop off from the funnel. This funnel of screens is as effective as we, as a company, build it. Therefore, user drop-off in this funnel falls entirely on our shoulders. So, with new onboarding screens built intentionally to lead users to finalize the loans application, we can attempt to get more than 40%of those predicted to not finish the process to complete the e-sign step. If we can do this, then we can drastically increase profit. Many lending companies provide hundreds of loan everyday, gaining money for each one. As a result, if we can increase the number of loan takers, we can increase the profits. All with a simple model!

The algorithms used for this modes was Logistic Regression, SVM, Decision Tree Classifier, Random Forest.

The method of preparation and selection of features is one of the biggest aspect of the success of this model. The Random Forest which yielded the best results was optimized even further using k-fold validation and grid search method and best parameters was selected.

**Future scope**

Our model gave a final prediction accuracy of 63% on the dataset. Which might be satisfactory for some p2p companies but we could further try to increase the accuracy of our model using a few more optimization techniques. We could further try to better the execution time of the model. It could be bettered by somehow manipulating our given data set.

Not only it can be used to study whether a customer will end up e-signing for loan or not we could even use this model in future to predict which person applying for the loan is likely to pay back loan on time based on the financial history of the person. It could even be used to decide what is the maximum amount which could be given to a person which he or she is likely to pay back on time in the future.

# BIBLIOGRAPHY

Reference Used :

* + - * Python :
      * <https://www.w3schools.com/python/python_reference.asp>
      * Python Libraries :
      * https://numpy.org
      * <https://pandas.pydata.org/>
      * https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html
      * Sci-Kit Learn :
      * https://scikit-learn.org/stable/getting\_started.html
      * Seaborn :
      * https://seaborn.pydata.org/introduction.html