DIKSCHA SAPRA:
dikschasapra@gmail.com

# ReadMe FILE:

This code uses python 3.6 and dependencies including numpy, pandas, matplotlib, seaborn and sci-kit-learn (sklearn), imblearn.
It contains 2 modules: loans_repayment_preprocessing.py and loan_repayment_model_training.py.
First have to run module 1, which will generate file: loans_preprocessed.csv, which would be used by Module 2.

## MODULE 1: FILE NAME: loans_repayment_preprocessing.py

### LOADING DATA:

First module checks our dataset, the datatype of each othe column, shape of the data set and checks the predictor variable: not.fully.paid (which is binary 0,1).

### Q A) If information for certain borrowers is not available in the dataset, what will you do? Why?

### NULL VALUE TREATMENT:

Data contains null value which has to be treated before any model can be built. Null value treatment is done by looking at a column, and the type of imputation that needs to be done.

1. Pub.rec: null values replaced by mode (0)
2. inq.last.6mnths: null values replaced by mode (0)
3. log.annual.inc: Replacing with mean since, std is very less
4. delinq.2yrs: null values replaced by mode (0)
5. revol.util: null values replaced by mode (0)
6. days.with.cr.line: We can see that std is comparable to mean, and there is no clear mode replacement for this column. The null values only consist of 0.3% of the total data, and hence are removed.

The rest of the questions are answered in the second module.

## MODULE 2: FILE NAME: loan_repayment_model_training.py

### LABEL ENCODING:

Before applying any other technique we have to encode our column with string datatype: purpose.
Using sklearn.preprocessing.LabelEncoder to change it into categorical data.

### TARGET VARIABLE:

We split our data set into two parts – x &y. X (independent features) contains all the features and y (dependent feature) contains the target variable: not.fully.paid

### OVERSAMPLING:

Note: in case of error or run on terminal/command line: pip install imblearn
The data is highly imbalanced, and hence it could be difficult to train due to biasness of negative class (class 0). Hence we need to oversample column 1 by using imblearn.over_sampling.SMOTE. SMOTE is Synthetic Minority Oversampling Technique.

### TRAINING AND TESTING:

DIKSCHA SAPRA:
dikschasapra@gmail.com

We split our data into two parts (in ratio 80:20). 80% of the data is used to train our model and we predict the loan repayment status in that group. (sklearn.model_selection.train_test_split)

**MODEL TRAINING:**
For our given data set, we get best performance on RandomForestClassifier (sklearn.ensemble.RandomForestClassifier). We train it by feeding x_train and y_train. And then predict on x_test.

**Q d) What is the best suited evaluation metric for this kind of dataset? Why?**

**MODEL PERFORMANCE:**
Achieving Around 90% accuracy, 83% sensitivity, 98% specificity, 0.8 Kappa score and 0.8 Mathew's correlation coefficient (all may vary slightly from run to run of the code).
For repayment dectection, we have to have highly sensitive data, since we have to decrease the false negatives more than false postives: i.e.: No repayment calls not detected wrongly are more costly than non correct repayment calls wrongly detected.
Accuracy is not an accurate predictor, since data is highly imbalanced, even if the model is bad, it will predict 0 more easily and will have high accuracy.
Cohen's kappa and Mathews correlation both are good metrics. Mathew's correlation tells how good a binary classifier is. Our score (0.8) is a good score.

**Q b) Identify the 5 best borrowers that will repay the entire amount. Support your work with proper performance metrics.**

**PROBABILITY OF CORRECT PREDICTION:**
**(Assuming that we have to find the best borrowers in testing set only, because if we search even for training set, we will find that data is extremely overfitted and biased).**
We can detect the probability of a row belonging to a particular class by using the object of the model that has been trained.

*probs = rf.predict_proba(x_test)*

Rf is the name of the object of the model, predict_proba give a 2 d array giving the probability that each row will belong to class 0 or 1. We have to find the best borrowers, so considering class 0, we sort the list of class 0 to find the best five borrowers all of which have a predictive probability of 1.
**The indexes of the selected borrowers are [1650 1482 1146  502  669]**

**Q c) What parameters affect loan repayment the most (List down Top 5)**

**FEATURE IMPORTANCE:**
We can detect the most important features by using the object of the model that has been trained.

*features = sorted(rf.feature_importances_)[::-1].*

Rf is the name of the object of the model. Sorted sorts the features in ascending order, [::-1] reverses our list to sort it in descending order. The names of columns according to these feature rankings are then stored in a list (found using np.argsort() which returns the indices of the sorted array) and the same is plotted using matplotlib and seaborn.
**The best 5 features are:** inq.last.6mths, credit.policy, purpose, int.rate, fico

DIKSCHA SAPRA:
dikschasapra@gmail.com

## RETROSPECTIVE (WHAT COULD HAVE BEEN DONE BETTER?)

While seeing feature importances, we are seeing features from the training set. The lime package for interpretability allows us to see what features are important for each of the row of our testing data. What features may have been responsible for non repayment of one person might not be the same for the other.