**Loan Default Risk**

**Assessment**

Dhruv Sapra , Raunak Singh Khalsi

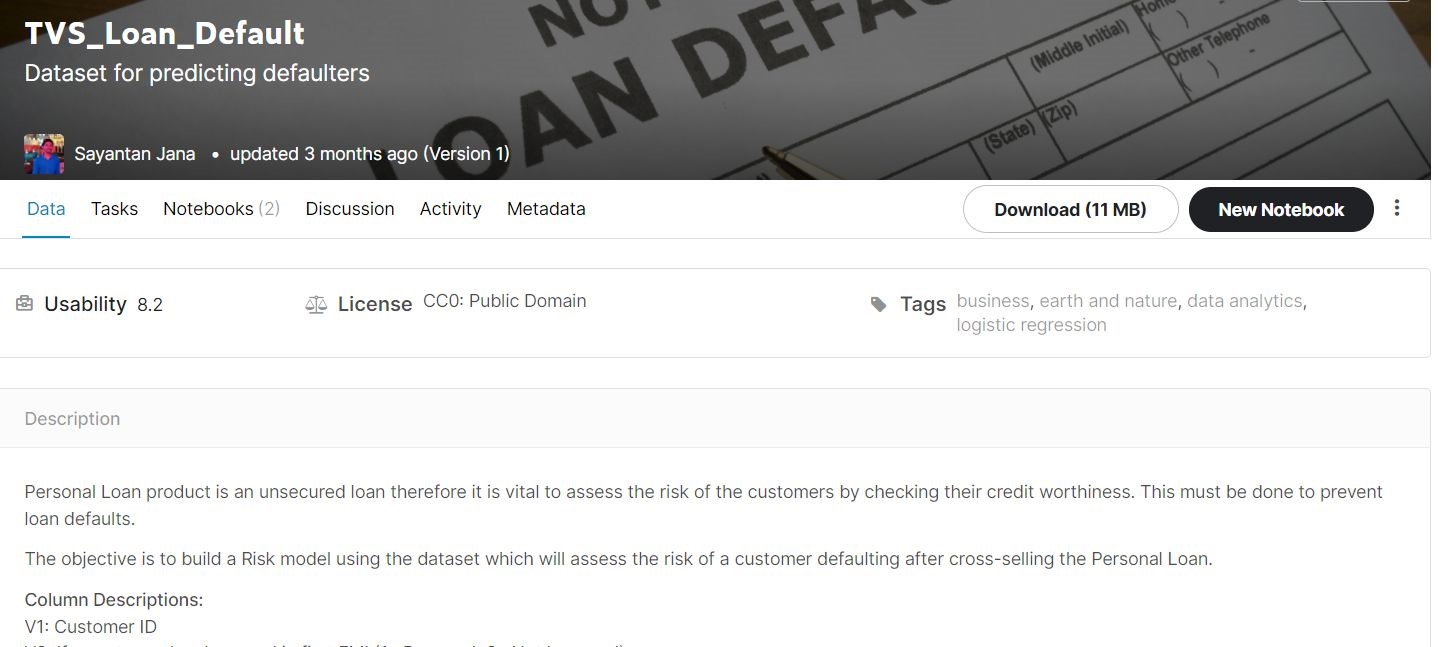
Github:

https://github.com/code-wizard123/tvs-credit-risk-eval

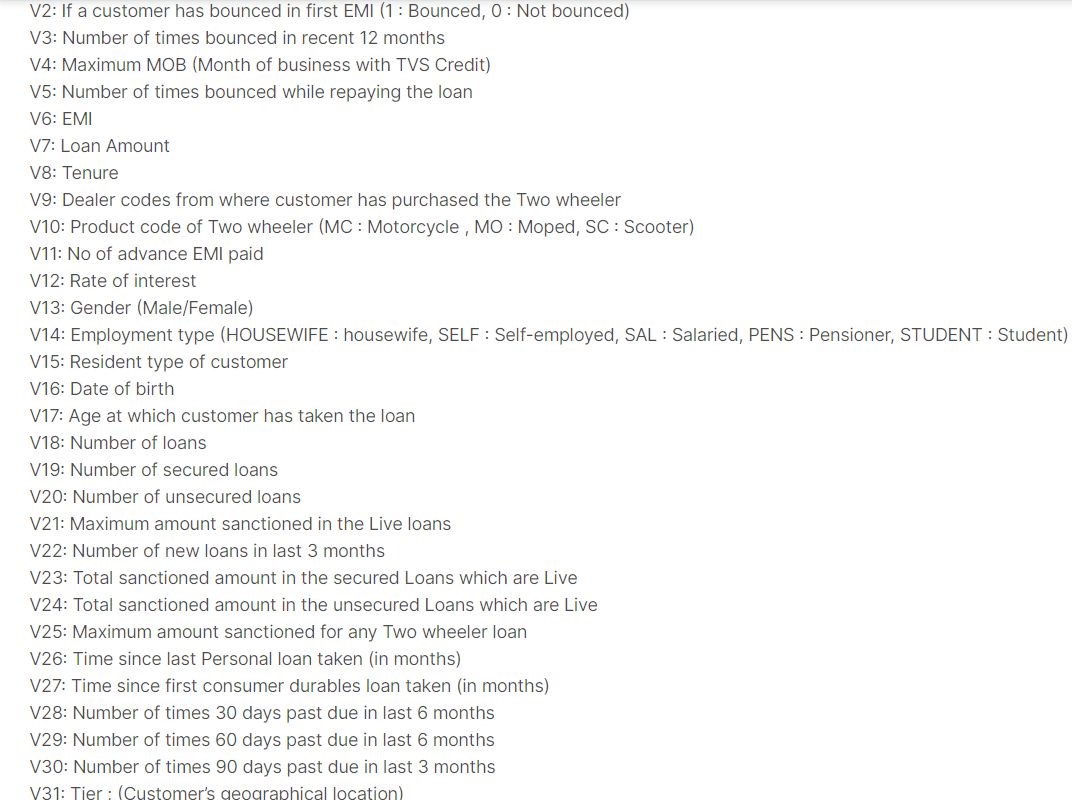
# Project Overview

* **Goal**: create loan default predictor app for TVS Credit Services
* **Plan**: try out different machine learning models
* **Process**: clean data, create and test models, then develop front end interface

# Dataset



# Column Descriptions



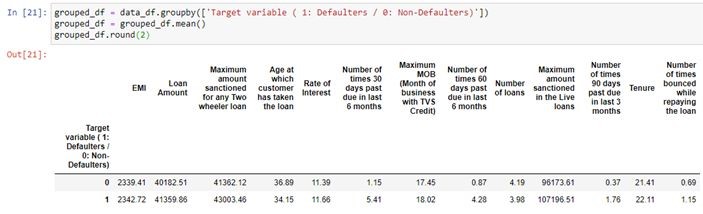
# Cleaning & Preprocessing Data



Before

After

# Deeper Dive into Data



- Number of times 30 days past due in last 6 months - Number of times 60 days past due in last 6 months - Number of times 90 days past due in last 3 months

# Deeper Dive into Data

Average default rate is just over 2%



A potential customer with a recent history of overdue loan payments is at high risk of defaulting

# Apply Classification ML Models

* Logistic Regression
* Random Forest
* Gradient-Boosted Tree/Forest
* SKLearn’s HistGradientBoostingClassifier
* XGBoost XGBClassifier

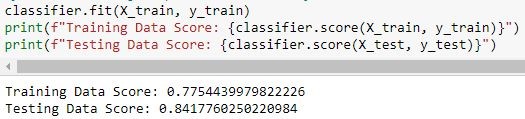
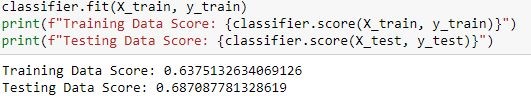
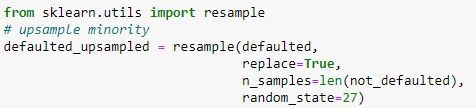
# Logistic Regression

* Used train, test, split from sklearn
* Then proceeded to **oversample** the data

○

●

Performed Grid Search to improve score



Before

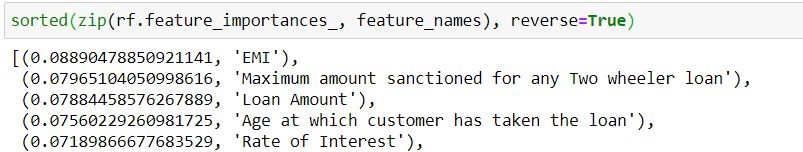
After

○

# Random Forest

First the dataset dropped the columns of “Customer ID” and “Dealer codes from where customer has purchased the Two wheeler”

Running a Decision Tree generated a p-value of around 0.95, and running Random Forest generated a p-value of around 0.97



# Gradient-Boosted ML Models

* Similar to random forests - technically tree ensembles
* Trees are built to optimize an objective function
* Each tree is built while keeping previous trees fixed
* Trees built by adding levels - a leaf is split into two leaves if tree score improves from split
* Useful for imbalanced classes in classification problems

**Scoring Metric - How to Compare Models?**

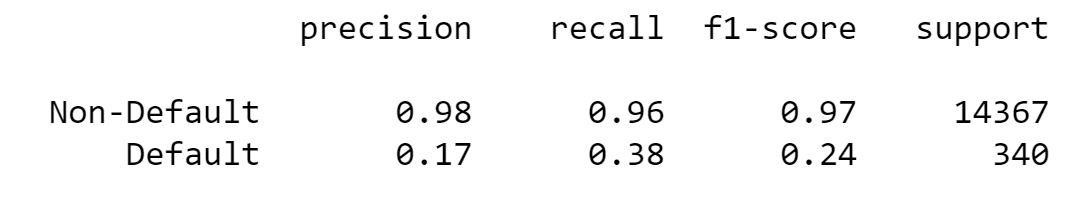
* Accuracy is not a valid comparison metric - classifiers will classify all customers as non-defaulters
* This would result in ~98% accuracy rate
* Compare by money predicted to be saved/spent when model is implemented to screen customers for personal loan offers
* Ratio of default loan worth to non-default loan worth is **theoretically** 5:1 - Must find a model that maximizes the equation:
* 5 \* (# of defaulters correctly identified) - (# of customers incorrectly identified as defaulters) - Later added a multiplicative factor to improve precision:
* (# of defaulters correctly identified)/(# of total defaulters)

# Best Model - XGBoost’s XGBClassifier

* 30-40% of all defaulters consistently found
* Loan defaulter/false negative ratio of 4.5-4.8 to 1

-

Slightly better than 5:1 ratio to “break even”



# Financial Analysis of XGBoost Model

* 5:1 Ratio : Average Completed Loan Profit ≈ 8,000 Rupees, Average Loan Amount ≈ 40,000 Rupees
* Successfully predicted 645 defaulters, but 3,055 false positives
* Utilizing the model for our sample data would have resulted in an additional 990,000 rupee profit, or $13,800

# Best Model - Issues

* Overfitting became an issue with training set data
* Could only rely on model’s results on test set data
* Made comparison with other models difficult due to limited test dataset size
* Model technically makes money using **theoretical** maximum loan ratio of 5:1, but this ratio is an estimate
* Previous best models were likely overfit but not detected due to the new scoring metric and lack of confusion matrices, so hard to compare to earlier “best” models

**Best Model - Issues cont’d** Training Results Test Results

