Loan Default Risk Assessment

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

Dhruv Sapra , Raunak Singh Khalsi

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

Data Tasks Notebooks (2) Discussion Activity Metadata

Download (11 MB) New Notebook

Usability 8.2 License CC0: Public Domain Tags business, earth and nature, data analytics, logistic regression

Description

Personal Loan product is an unsecured loan therefore it is vital to assess the risk of the customers by checking their credit worthiness. This must be done to prevent loan defaults.

The objective is to build a Risk model using the dataset which will assess the risk of a customer defaulting after cross-selling the Personal Loan.

Column Descriptions:
V1: Customer ID

Column Descriptions

V2: If a customer has bounced in first EMI (1 : Bounced, 0 : Not bounced)
V3: Number of times bounced in recent 12 months
V4: Maximum MOB (Month of business with TVS Credit)
V5: Number of times bounced while repaying the loan
V6: EMI
V7: Loan Amount
V8: Tenure
V9: Dealer codes from where customer has purchased the Two wheeler
V10: Product code of Two wheeler (MC : Motorcycle , MO : Moped, SC : Scooter)
V11: No of advance EMI paid
V12: Rate of interest
V13: Gender (Male/Female)
V14: Employment type (HOUSEWIFE : housewife, SELF : Self-employed, SAL : Salaried, PENS : Pensioner, STUDENT : Student)
V15: Resident type of customer
V16: Date of birth
V17: Age at which customer has taken the loan
V18: Number of loans
V19: Number of secured loans
V20: Number of unsecured loans
V21: Maximum amount sanctioned in the Live loans
V22: Number of new loans in last 3 months
V23: Total sanctioned amount in the secured Loans which are Live
V24: Total sanctioned amount in the unsecured Loans which are Live
V25: Maximum amount sanctioned for any Two wheeler loan
V26: Time since last Personal loan taken (in months)
V27: Time since first consumer durables loan taken (in months)
V28: Number of times 30 days past due in last 6 months
V29: Number of times 60 days past due in last 6 months
V30: Number of times 90 days past due in last 3 months
V31: Tier : (Customer's geographical location)

Cleaning & Preprocessing Data

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V23	V24	V25	V26	V27	V28	V29	V30	V31	V32
0	1	0	0	24.0	0.0	2432.0	46500.0	24.0	1385.0	SC	...	NaN	NaN	46500.0	NaN	NaN	0	0	0	TIER 1	0
1	2	0	1	24.0	1.0	1495.0	28168.0	24.0	2496.0	MO	...	NaN	NaN	28168.0	NaN	NaN	0	0	0	TIER 1	0
2	3	0	0	26.0	0.0	1707.0	38900.0	30.0	1346.0	SC	...	NaN	105000.0	38900.0	34.0	31.0	31	31	16	TIER 1	0
3	4	0	0	24.0	0.0	2128.0	42900.0	24.0	1375.0	SC	...	NaN	NaN	42900.0	NaN	NaN	0	0	0	TIER 1	0
4	5	0	0	27.0	0.0	1824.0	40900.0	30.0	4140.0	MC	...	NaN	NaN	40900.0	NaN	NaN	0	0	0	TIER 1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
119523	119524	0	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N							
119524	119525	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	3300							
119525	119526	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N							
119526	119527	0	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	31990.0	4360							
119527	119528	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	4990							

Before

After

	EMI	Loan Amount	Maximum amount sanctioned for any Two wheeler loan	Age at which customer has taken the loan	Rate of Interest	Number of times 30 days past due in last 6 months	Maximum MOB (Month of business with TVS Credit)	Number of times 60 days past due in last 6 months	Number of loans	Maximum amount sanctioned in the Live loans	Number of times 90 days past due in last 3 months	Tenure	Number of times bounced while repaying the loan
0	2432.0	46500.0	46500.0	40.0	12.75	0	24.0	0	1	0.0	0	24.0	0.0
1	1495.0	28168.0	28168.0	47.0	13.65	0	24.0	0	1	0.0	0	24.0	1.0
2	1707.0	38900.0	38900.0	31.0	12.65	31	26.0	31	9	55000.0	16	30.0	0.0
3	2128.0	42900.0	42900.0	24.0	9.50	0	24.0	0	1	0.0	0	24.0	0.0
4	1824.0	40900.0	40900.0	30.0	13.50	0	27.0	0	1	0.0	0	30.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
73527	2941.0	47900.0	47900.0	47.0	6.99	5	14.0	3	4	160400.0	0	18.0	0.0
73528	2870.0	41000.0	41000.0	35.0	3.99	0	10.0	0	1	0.0	0	15.0	0.0
73529	2720.0	28350.0	28350.0	46.0	15.10	6	10.0	6	5	32000.0	3	12.0	0.0
73530	3500.0	50000.0	50000.0	56.0	4.00	0	8.0	0	2	0.0	0	15.0	0.0
73531	2750.0	27500.0	27500.0	36.0	0.00	0	8.0	0	1	0.0	0	10.0	0.0

Deeper Dive into Data

```
In [21]: grouped_df = data_df.groupby(['Target variable ( 1: Defaulters / 0: Non-Defaulters)'])
         grouped_df = grouped_df.mean()
         grouped_df.round(2)
```

Out[21]:

Target variable (1: Defaulters / 0: Non-Defaulters)	EMI	Loan Amount	Maximum amount sanctioned for any Two wheeler loan	Age at which customer has taken the loan	Rate of Interest	Number of times 30 days past due in last 6 months	Maximum MOB (Month of business with TVS Credit)	Number of times 60 days past due in last 6 months	Number of loans	Maximum amount sanctioned in the Live loans	Number of times 90 days past due in last 3 months	Tenure	Number of times bounced while repaying the loan
0	2339.41	40182.51	41362.12	36.89	11.39	1.15	17.45	0.87	4.19	96173.61	0.37	21.41	0.69
1	2342.72	41359.86	43003.46	34.15	11.66	5.41	18.02	4.28	3.98	107196.51	1.76	22.11	1.15

- 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

30 day groups	Percent Chance of Defaulting
0-1 times 30 days past due in last 6 months	0.964512
2+ times 30 days past due in last 6 months	9.466963

60 day groups	Percent Chance of Defaulting
0-1 times 60 days past due in last 6 months	1.109901
2+ times 60 days past due in last 6 months	9.074855

90 day groups	Percent Chance of Defaulting
0-1 times 90 days past due in last 6 months	1.656719
3+ times 90 days past due in last 3 months	10.119785

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

```
from sklearn.utils import resample
# upsample minority
defaulted_upsampled = resample(defaulted,
                               replace=True,
                               n_samples=len(not_defaulted),
                               random_state=27)
```

- Performed Grid Search to improve score

```
classifier.fit(X_train, y_train)
print(f"Training Data Score: {classifier.score(X_train, y_train)}")
print(f"Testing Data Score: {classifier.score(X_test, y_test)}")
```

```
Training Data Score: 0.6375132634069126
Testing Data Score: 0.687087781328619
```

Before

```
classifier.fit(X_train, y_train)
print(f"Training Data Score: {classifier.score(X_train, y_train)}")
print(f"Testing Data Score: {classifier.score(X_test, y_test)}")
```

```
Training Data Score: 0.7754439979822226
Testing Data Score: 0.8417760250220984
```

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

```
sorted(zip(rf.feature_importances_, feature_names), reverse=True)
```

```
[(0.08890478850921141, 'EMI'),
 (0.07965104050998616, 'Maximum amount sanctioned for any Two wheeler loan'),
 (0.07884458576267889, 'Loan Amount'),
 (0.07560229260981725, 'Age at which customer has taken the loan'),
 (0.07189866677683529, 'Rate of Interest'),
```

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”

	precision	recall	f1-score	support
Non-Default	0.98	0.96	0.97	14367
Default	0.17	⇨ 0.38	0.24	340

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

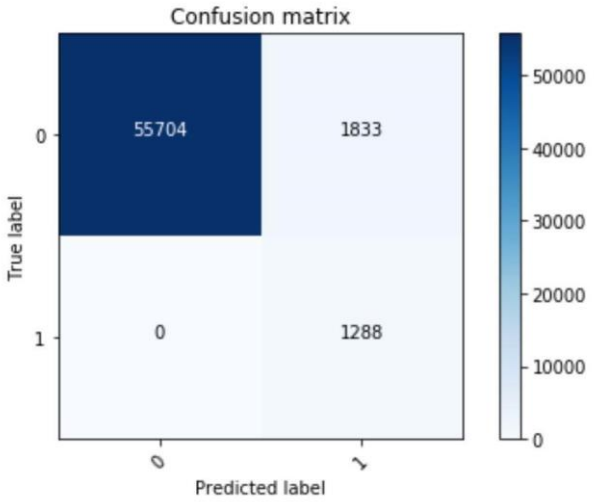
Best Model - Issues

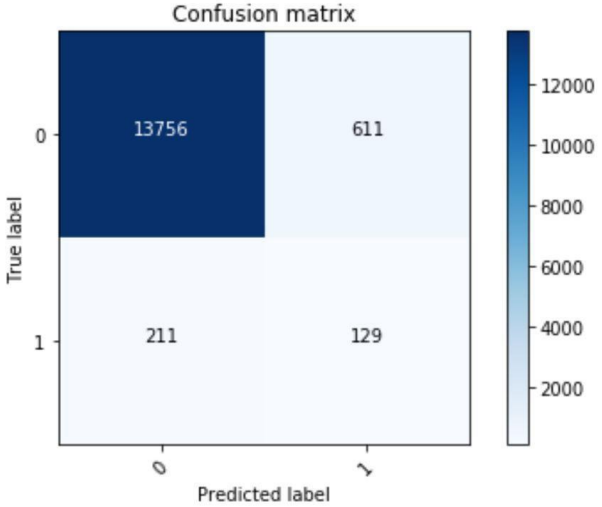
- Overfitting became an issue with training set data
- Could only rely on model's results on test set data
- Made comparison with other models difficult due to limited test dataset size
- Model technically makes money using **theoretical** maximum loan ratio of 5:1, but this ratio is an estimate
- Previous 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


Confusion matrix
55704
1833
0
1288
True label
Predicted label
0
1
50000
40000
30000
20000
10000
0


Confusion matrix
13756
611
211
129
True label
Predicted label
0
1
12000
10000
8000
6000
4000
2000