SELF-PAY PREDICTION MODEL

Team Mavericks: Vikas Khati, Prajjwal Kumar, Siddharth JP

INDEX

Executive Summary

- TVS Credit being a NBFC which is into lending, the monthly collections have a direct correlation to its profits.
- Collections happen in mainly two methods, i) Field agents and ii) Telecall.
- To reduce the human effort required in field we can use machine learning techniques to identify self paying customers who can be targeted by the tele-callers.
- The features are both categorical and numerical. The categorical variables, depending on the nature can be converted to numerical using one hot encoding or label encoding.
- Data pre-processing and feature engineering must be done to lower error rates and increase accuracy.
- Models such as SVM, logistic regression, PCA, Random forest are used to classify data and the accuracy rates are compared. **Highest of 93.8% is achieved**.

©
insetfair

Loading and Reading the data

- The dataset has 547308 data points and 30 columns consisting the features.
- It has 29 features out of which 18 are numerical and 11 are categorical variables.
  The last column being the target variable.
- The shape() function gives the dimensions and the head() function describes the first 5 datapoints.

```
data = pd.read_csv('DatasetModified.csv')
data.head()
data.shape
```

	CUST_ID	Each Month Last Date	DateCA	P.Type	ModelCode	DealerCode	AppDownloaded	Adv.EMIno	LoanTenure	EMI	DOB	AreaCode	AssetCost	LoanAmount	DownPayment
0	CNO000001	31-03-2019	21-09-2018	MOBILE	CD00124531	CD04818	Y	3	10	1799	16-11-1978	3054	17990	17990	5527
1	CNO000002	30-04-2019	21-09-2018	MOBILE	CD00124531	CD04818	Y	3	10	1799	16-11-1978	3054	17990	17990	5527
2	CNO000003	31-05-2019	21-09-2018	MOBILE	CD00124531	CD04818	Y	3	10	1799	16-11-1978	3054	17990	17990	5527

Data analysis: Checking for null values

- While checking for null values we observe the following 3 occurrences.

1. Values missing in the range of ~500000
   -Looking at the nature of the variables. There can be 2 cases here, either the people might have not taken the particular loan therefore we will put 0 or consider them as missing values and drop the columns due to the size.

2. Values missing in the range of ~40000:
   - We replace the values with the mean of the population.

3. Categorical variable missing values.
   - Since the only one missing is Resident type , and the number of missing values is less w.r.t to overall size , therefore we can assume it as either owned or rent.

```
data.isnull().sum()

CUST_ID                          0
Each Month Last Date             0
DateCA                           0
P.Type                           0
ModelCode                        0
DealerCode                       0
AppDownloaded                    0
Adv.EMIno                        0
LoanTenure                       0
EMI                              0
DOB                              0
AreaCode                         0
AssetCost                        0
LoanAmount                       0
DownPayment                      0
Qualifi.                         0
Employ. Type                     0
ResidentType                  3936
BouncedTimes                     0
Buss. Mon with Cust.             0
FuturePrinciple                  0
OverallMaxLoanAmount         42191
UnsecMaxLoanAmount           42222
Timelastloan                 42175
TimelastpersonalLoan        493480
TimelastLivepersonalLoan    502763
TimelastClosedpersonalLoan  526616
TimelastLiveBusinnessLoan   518331
TimelastConsumerLoan         42225
SELF_PAY                         0
```

© insetfair

Data analysis: Statistics

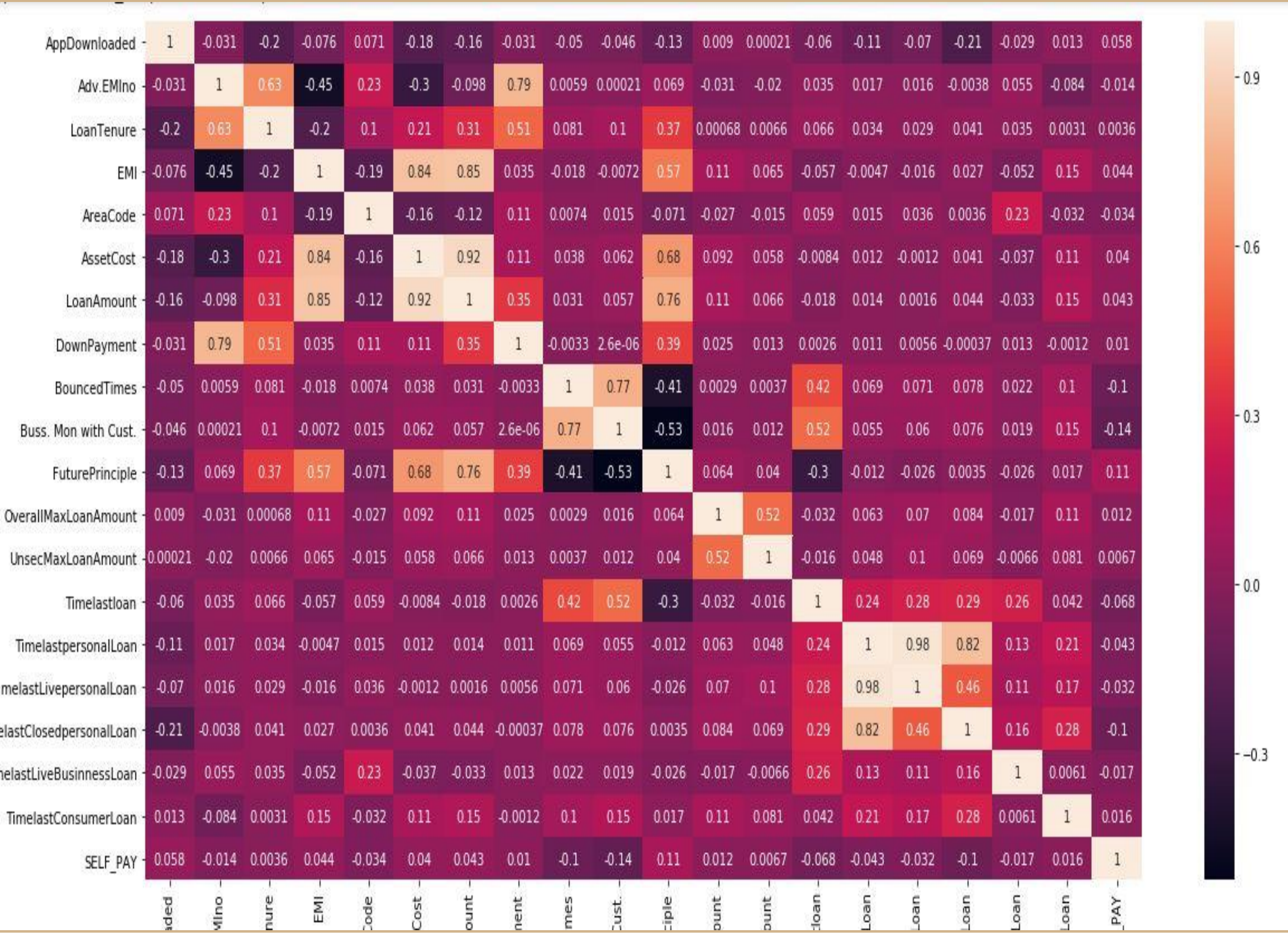
- We use the describe() function to learn about mean, std dev, min, max of the features.

```
(547308, 30)
```

```
[ ] data.describe()
```

	Adv.EMIno	LoanTenure	EMI	AreaCode	AssetCost	LoanAmount	DownPayment	BouncedTimes	Buss. Mon with Cust.	FuturePrinciple
count	547308.000000	547308.000000	547308.000000	547308.000000	547308.000000	547308.000000	547308.000000	547308.000000	547308.000000	547308.000000
mean	2.525258	10.627257	1854.651699	3031.169950	21096.872478	19365.248524	4403.523499	3.025395	3.788428	12352.398898
std	1.412450	2.243713	750.749861	27.873827	10360.971390	8045.232442	2770.269504	1.977241	2.194184	6951.576244
min	0.000000	6.000000	459.000000	3000.000000	5500.000000	5000.000000	0.000000	1.000000	1.000000	0.000000
25%	2.000000	10.000000	1334.000000	3007.000000	14000.000000	13591.000000	3130.000000	1.000000	2.000000	7585.120000
50%	3.000000	10.000000	1667.000000	3025.000000	18000.000000	17500.000000	4480.000000	3.000000	4.000000	11099.610000
75%	4.000000	12.000000	2167.000000	3054.000000	24000.000000	22900.000000	5894.000000	4.000000	5.000000	15599.450000
max	8.000000	36.000000	16667.000000	3115.000000	263000.000000	212668.000000	54224.000000	16.000000	16.000000	177225.810000

Correlation Heatmap




```python
Var_Corr = data.corr()

plt.figure(figsize=(20,10))

sns.heatmap(Var_Corr, xticklabels=Var_Corr.columns, yticklabels=Var_Corr.columns, annot=True)
```

- We plot a correlation map to identify highly correlated variables, which can then be removed.
- The highly correlated values can be removed manually or we can use a technique called Principal Component Analysis which takes care of high correlation.

Feature Engineering

- We feature engineer two new variables namely **Interest and SecuredLoans.**

- **Interest = LoanAmount + DownPayment – AssetCost**
- **SecuredLoans = OverallMaxLoanAmount – UnsecMaxLoanAmount**

- These new variables help unearth new dependencies of the features to the dataset.

Feature Engineering

```
[ ] Interest = data['LoanAmount']+data['DownPayment']-data['AssetCost']
    SecuredLoans = data['OverallMaxLoanAmount']-data['UnsecMaxLoanAmount']
    data.insert(13,"Interest",Interest,True)
    data.insert(24,"SecuredLoans",SecuredLoans,True)
    data.head()
```

©
insetfair

Feature Engineering

- The birth date can be converted to a numerical value of **AGE** which is further for analysis instead using it in the raw date time format.

```
data['DOB']=data['DOB'].apply(lambda x: pd.to_datetime('today').year-pd.to_datetime(x).year)

data.head()
```

© insetfair

One-Hot and Label Encoding

- Categorical variables are first converted by label encoding then into one-hot ,as one hot encoding only takes numerical values.
- For AppDownloaded , Yes is assigned with **higher value 1** and **No with 0** ,as it can be inferred from the dataset that people with apps tend to selfpay.
- Qualification is label encoded according to ratio of self paid divided by self unpaid people.
- One-hot encoding for dealer code creates about 8000 columns as , so other pre-processing methods should be preferred

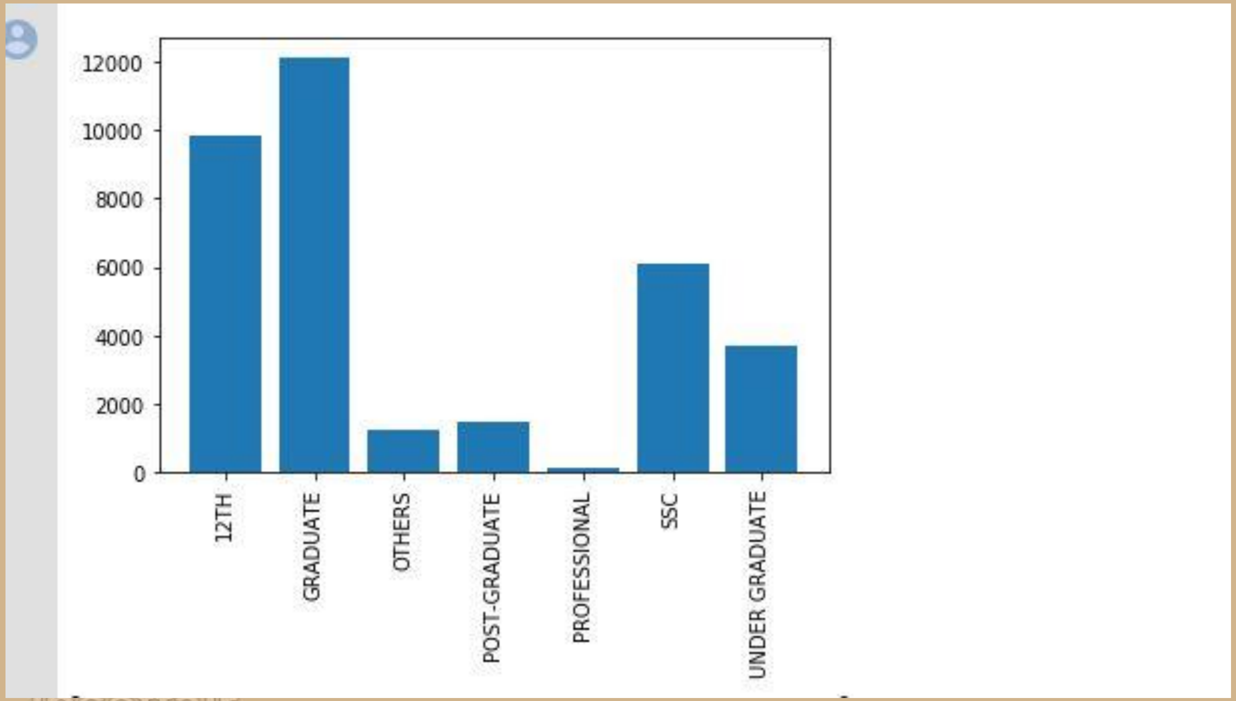
```
[ ] data.AppDownloaded.replace(('Y', 'N'), (1, 0), inplace = True)
    data.head()
```

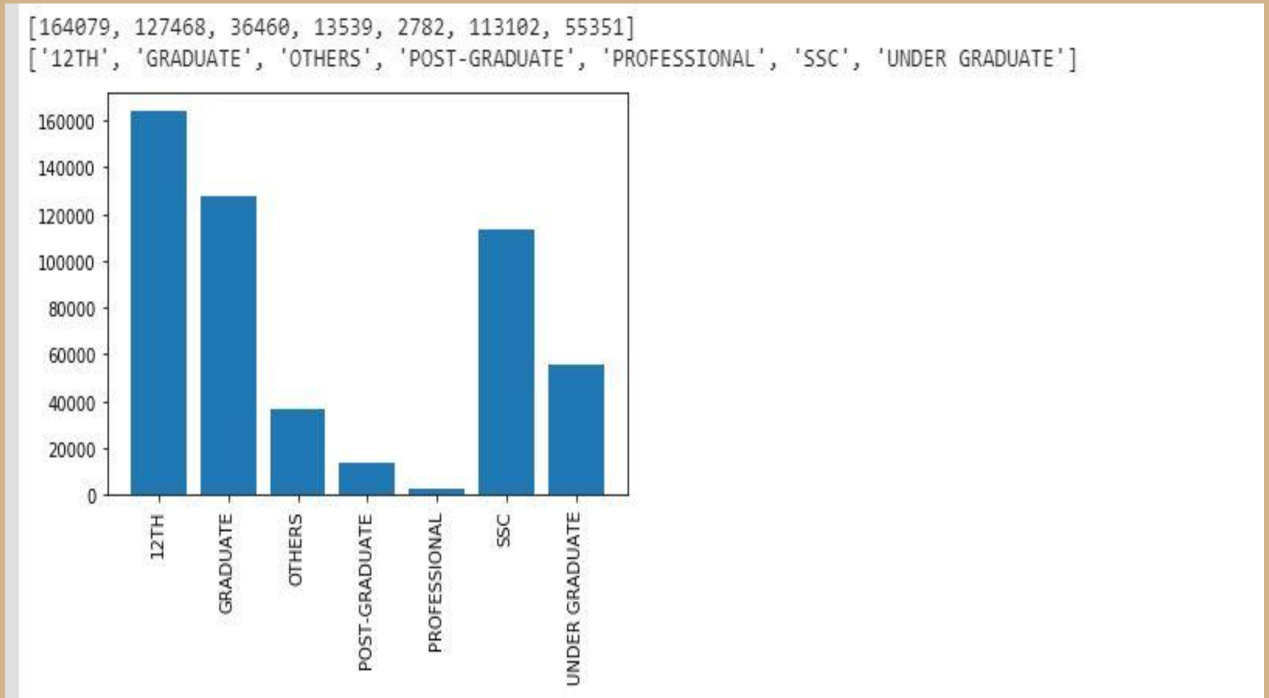
```
[ ]
    from sklearn.preprocessing import LabelEncoder, OneHotEncoder
    le =LabelEncoder()
    # X[:,3]=le.fit_transform(X[:,3])

    data['P.Type']=le.fit_transform(data['P.Type'])
    data['Employ. Type']=le.fit_transform(data['Employ. Type'])
    data['ResidentType']=le.fit_transform(data['ResidentType'])
    data.head()
```

One-Hot and Label Encoding

- Qualification plotted against SelfPay = 1 & SelfPay = 0





Data Split and Scaling

- The training and test set are split in the ratio of **4:1**, in order to train the models and verify it using the various classification algorithms.

- First before fitting the model we scale the data to maintain uniformity amongst features.

```
[ ] # Splitting the dataset into the Training set and Test set
    from sklearn.model_selection import train_test_split
    Dep_train, Dep_test, Indep_train, Indep_test = train_test_split(Dep, Indep, test_size = 0.2, random_state = 0)
```

```
[ ] # Feature Scaling
    from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    Dep_train = sc.fit_transform(Dep_train)
    Dep_test = sc.transform(Dep_test)
```

insetfair

Principal Component Analysis

- **Principal component analysis** (**PCA**) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called **principal components**.

- For this particular dataset , we extract the features into 5 distinct principal components.

```
[ ]  # Applying PCA
     from sklearn.decomposition import PCA
     pca = PCA(n_components = 5)
     Dep_train = pca.fit_transform(Dep_train)
     Dep_test = pca.transform(Dep_test)
     explained_variance = pca.explained_variance_ratio_
```

©
insetfair

Training Models

- We use the dataset to train models of

  Support Vector Machines,
  Decision trees,
  Random forest,
  Logistic Regression,
  KNN Classifier,
  Gaussian NB

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn import svm, tree
    from sklearn.linear_model import LogisticRegression
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.naive_bayes import GaussianNB
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score, confusion_matrix

    #Analyzing which classifier is the best for the task using confusion matrix and checking the accuracy
    classifiers = []
    model_1 = svm.SVC()
    classifiers.append(model_1)
    model_2 = tree.DecisionTreeClassifier()
    classifiers.append(model_2)
    model_3 = RandomForestClassifier()
    classifiers.append(model_3)
    model_4 = LogisticRegression()
    classifiers.append(model_4)
    model_5 = KNeighborsClassifier(n_neighbors = 5)
    classifiers.append(model_5)
    model_6 = GaussianNB()
    classifiers.append(model_6)
```

© insetfair

Accuracy

- We observe that SVM produces the highest accuracy rate.
- We further use it trained model to predict values of new datasets in the feature.

	Name of model	Accuracy
0	SVM	93.796934
1	Decision Tree	88.472712
2	Random Forest	93.362994
3	Logistic Regression	93.794193
4	Knn	93.479016
5	Naive Bayes	93.711973

Best classifier is SVM and its accuracy is 93.79693409585062

© insetfair

Conclusion

- The data has been analysed and machine learning techniques have been applied to train an appropriate model which can be deployed in TVSCredit's system to improve efficiency and in turn increase profits.

- The model for 93% of the time correctly classifies the defaulters.

©
insetfair

Thank You