Upvote 0 0 0 0 0

Loan Prediction

Author: Ankit Gupta
Email: ankitgupta.iiitm@gmail.com
Linkedin: linkedin.com/in/ankit-gupta-iiitm/
Kaggle: kaggle.com/guptaankit

Problem Statement

Dream Housing Finance company deals in all kinds of home loans. They have presence across all urban, semi urban and rural areas. Customer first applies for home loan after that company validates the customer eligibility for loan.

Objective: Automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process they have provided a dataset to identify the customer segments that are eligible for loan amount so that they can specifically target these customers.

Hypothesis Generation

Hypothesis generation is a process of listing all possible factors that affect the outcome. In this case, we are trying to predict if a person will be able to get a loan or not. Following are the factors which I think should be considered while predicting the loan status:

1. Applicant Income: Higher the applicant's income, higher the chances of loan approval.
2. Coapplicant Income: Higher the coapplicant's income, higher the chances of loan approval.
3. Applicant Gender: Chances of loan approval might differ for male and female applicants.
4. Loan Amount: Chances of loan approval might depend on the loan amount being requested.
5. Loan Amount Term: Loan approval might depend on the time period for which the loan is required.
6. Credit History: Applicants with a good credit history should have better chances of loan approval.
7. Property Area: Chances might be higher for those living in urban or semi urban areas.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. It is a preliminary step before more formal statistical analysis or machine learning. In this section, we will explore the given dataset to see if we can find any patterns in the data that can be used to predict the loan status.

Data Cleaning

Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. Data cleaning may be performed interactively with data wrangling tools, or batch processing through scripting.

Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive. The features chosen influence both the performance of the algorithms and the degree of complexity which is required to achieve good performance. We will try to create some new features which may be more relevant to the target variable.

Model Building: Part I

After cleaning the data and creating new features, we are now ready to start building predictive models. In this first part, I'll try to keep it simple and will use some of the most popular classification algorithms - Logistic Regression, Decision Tree and Random Forest. In the second part, I'll try to improve the accuracy by tuning the parameters of the models and also will try to use some more advanced algorithms.

Cross Validation

Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an estimate of how well the learning algorithm will do in predicting unseen instances. This is important because we want to use our model to predict the loan status of new customers. Cross validation is a statistical method for validating a predictive model. Subsets of the data are held out for use as validation sets; a model is fit to the data in the training set and evaluated on the validation set. If the model delivers good statistics on the validation set, we can trust the model to be a good predictive model.

Model Building: Part II

In the first part of this section, we tried to predict the loan status using some of the most popular classification algorithms. In this second part, we will try to improve the accuracy by tuning the parameters of the models and also will try to use some more advanced algorithms.

Conclusion

In this section, we will try to summarize the learnings from the entire exercise and also try to see what can be done to improve the accuracy.

Data Dictionary

Following is the description of the variables in the dataset:

1. Loan_ID: Unique Loan ID
2. Gender: Male/ Female
3. Married: Applicant married (Y/N)
4. Dependents: Number of dependents
5. Education: Applicant Education (Graduate/ Undergraduate)
6. Self_Employed: Self employed (Y/N)
7. ApplicantIncome: Applicant income
8. CoapplicantIncome: Coapplicant income
9. LoanAmount: Loan amount in thousands
10. Loan_Amount_Term: Term of loan in months
11. Credit_History: Credit history meets guidelines
12. Property_Area: Urban/ Semi Urban/ Rural
13. Loan_Status: Loan approved (Y/N)

Loan status is the dependent variable. Rest all are independent variables. Loan_ID is just a serial number and can be ignored. Credit_History is a binary variable, 1 for "Yes" and 0 for "No".

Import Libraries and Data Set

Following are the libraries we will use in this exercise:

1. numpy: Numpy stands for Numerical Python. It is the core library for scientific computing in Python. It contains a powerful n-dimensional array object, sophisticated broadcasting functions, tools for integrating C/C++ and Fortran code, useful linear algebra, Fourier transform, and random number capabilities.
2. pandas: Pandas is a high-performance, easy-to-use data structures and data analysis library. It provides fast, flexible, and expressive data structures designed to make working with relational or labeled data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.
3. matplotlib: Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
4. seaborn: Seaborn is a library for making attractive and informative statistical graphics in Python. It is based on matplotlib.
5. sklearn: Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

```
# Import numpy
import numpy as np
np.random.seed(2)

# Import pandas
import pandas as pd
from pandas import Series, DataFrame
from pandas.tools.rplot import *

# Import matplotlib
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline

# Import seaborn
import seaborn as sns

# Import sklearn
from sklearn import cross_validation, metrics
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_segment import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import RFECV

# Import warnings
import warnings
warnings.filterwarnings('ignore')
```

```
# Define path for data set
path = r'C:\Data\Analytics Vidhya\Loan Prediction'

# Read data set
train = pd.read_csv(path+r'\train.csv')
test = pd.read_csv(path+r'\test.csv')
```

```
# Dimensions of training data set
train.shape
```

```
(614, 13)
```

```
# Dimensions of test data set
test.shape
```

```
(367, 12)
```

```
# Preview of training data set
train.head(10)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	NaN	Urban	Y
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	0.0	Semiurban	N
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0	1.0	Urban	Y
9	LP001020	Male	Yes	1	Graduate	No	12841	10968.0	349.0	360.0	1.0	Semiurban	N

```
# Preview of test data set
test.head(10)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001026	Male	Yes	0	Graduate	Yes	5000	1800	208.0	360.0	1.0	Urban
3	LP001028	Male	Yes	2	Graduate	No	2000	2384	100.0	360.0	1.0	Urban
4	LP001043	Male	Yes	0	Not Graduate	No	2000	0	84.0	360.0	1.0	Rural
5	LP001046	Male	Yes	1	Graduate	No	5116	1779	106.0	360.0	1.0	Urban
6	LP001047	Male	Yes	0	Not Graduate	No	2268	2083	94.0	360.0	1.0	Semiurban
7	LP001050	NaN	Yes	2	Not Graduate	No	3065	NaN	188.0	360.0	NaN	Rural
8	LP001054	Male	Yes	0	Graduate	Yes	3000	3416	56.0	180.0	1.0	Urban
9	LP001055	Female	No	1	Not Graduate	No	2083	4083	160.0	360.0	NaN	Semiurban

```
# Check data types
train.dtypes
```

```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome      float64
LoanAmount             float64
Loan_Amount_Term       float64
Credit_History         float64
Property_Area         object
Loan_Status           object
dtype: object
```

```
# Check data types
test.dtypes
```

```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome      int64
LoanAmount             float64
Loan_Amount_Term       float64
Credit_History         float64
Property_Area         object
dtype: object
```

```
# Check data types
train.dtypes.value_counts
```

```
object    13
int64      5
float64    4
dtype: object
```

```
# Check data types
test.dtypes.value_counts
```

```
object    12
int64      4
float64    4
dtype: object
```

```
# Check missing values
train.isnull().sum
```

```
Loan_ID               0
Gender                13
Married               3
Dependents            15
Education             0
Self_Employed         32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            22
Loan_Amount_Term      14
Credit_History        50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
# Check missing values
test.isnull().sum
```

```
Loan_ID               0
Gender                11
Married               0
Dependents            10
Education             0
Self_Employed         23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History        29
Property_Area         0
dtype: int64
```

```
# Check unique values
train.Gender.unique()
```

```
array[Male, Female, nan]
```

```
# Check unique values
train.Dependents.unique()
```

```
array[0, 1, 2, 3+, nan]
```

```
# Check unique values
train.Education.unique()
```

```
array[Graduate, Not Graduate]
```

```
# Check unique values
train.Self_Employed.unique()
```

```
array[No, Yes, nan]
```

```
# Check unique values
train.Property_Area.unique()
```

```
array[Urban, Rural, Semiurban]
```

```
# Check unique values
train.Loan_Status.unique()
```

```
array[Y, N]
```

```
# Check unique values
train.Credit_History.unique()
```

```
array[1., 0., nan]
```

```
# Check unique values
train.Loan_Amount_Term.unique()
```

```
array[360., 120., 240., 180., 60., 300., 480., nan, 36., 84., 12.]
```

```
# Check unique values
train.ApplicantIncome.unique()
```

```
array[5849, 4583, 3000, 2583, 6000, 5417, 2333, 3036, 4006, 12841,
5351, 5000, 2500, 2083, 6250, 3200, 2857, 1774, 9328, 1500,
1889, 2708, 1828, 3667, 3748, 3980, 3716, 4348, 3381, 6754,
5204, 1699, 1848, 4805, 3099, 5179, 5296, 4772, 2984, 2167,
9323, 3707, 2071, 1768, 7677, 2064, 5191, 2492, 2375, 3620,
3617, 6783, 3365, 3713, 6080, 8133, 7901, 4731, 2267, 2354,
1916, 4283, 3083, 7682, 3959, 3877, 1299, 3773, 8072, 7583,
4192, 2137, 8980, 4548, 1844, 2773, 3687, 2226, 1633, 2666,
7333, 3846, 4095, 3418, 3092, 3666, 1321, 2744, 2929, 4666,
3333, 3555, 3415, 3635, 5063, 3812, 3884, 5046, 2630, 5799,
2365, 2890, 3573, 3890, 2340, 3796, 3010, 2555, 4333, 2268,
2995, 3102, 1666, 2652, 2779, 4010, 5633, 2807, 5064, 2610,
3678, 3626, 4333, 3811, 3901, 3643, 2891, 3173, 2931, 3232,
2773, 3039, 1837, 3077, 3043, 2887, 2682, 2708, 3777, 3250,
4333, 2960, 4691, 3910, 3771, 3250, 3268, 3261, 4233, 2479,
3687, 1987, 2563, 2173, 2333, 3418, 4339, 1709, 1944, 1709,
2383, 3062, 3087, 3229, 1780, 3029, 2305, 2085, 3188, 1907,
3091, 3797, 2610, 1911, 3232, 2401, 2575, 2067, 2400, 3091,
2053, 2781, 2589, 3167, 4708, 2333, 3418, 3500, 4616, 3384,
3492, 3444, 3608, 1853, 3018, 4054, 3133, 3604, 1025, 3246,
3687, 3405, 6080, 2064, 1578, 3008
```