# Statistics 652: Project - Lending Club Loan History Challenge

Jessica Grover

The aim of the Project is to properly categorize the loan status of authorized loans by using various machine learning algorithms. The data used is LendingClub data from 2012 to 2014. The objective is to equal or improve upon the accuracies attained in the LoanDefault-Prediction competition on Github, which is primarily a classifier-building challenge rather than a quantitative prediction model. The effective application of machine learning algorithms to this dataset might have significant ramifications for the lending sector and loan approval accuracy.

### Introduction:-

The goal of this project is to create a machine learning model that uses the logistic regression to categorize the Loan Status of authorized LendingClub loans from 2012 to 2014. We are downloading the data set from Kaggle. Collecting data, studying and preparing data, training a model on the data, assessing model performance, and improving model performance are all implemented in this project.

For categorical classification, logistic regression is the best machine learning method. Because of its simplicity and efficiency, it is one of the most used models. The model predicts the likelihood of an occurrence based on a collection of predictor factors. The goal of this project is to apply logistic regression to predict whether or not a consumer will purchase a product. We will employ a data set that includes demographic information as well as the consumers' browsing history. Our objective is to create an accurate model that can forecast the data.

### Step 1: Data collection

The core data file, which comprises data from 2007 to 2018, will be supplemented with the sanctioned loans from 2012 to 2014 when the data has been downloaded in CSV format. Some of the important variables of the data are:

loan_amnt - Loan amount of the customer.

funded_amnt - approved by the bank.

int_rate - interest rate of the loan.

installment - Installment done by customer.

annual_inc - Annual income of the customer.

## importing important libraries

```
knitr::opts_chunk$set(echo = TRUE)
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(pacman)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
p_load(tidyverse, tidymodels, naniar, DataExplorer, janitor, discrim)
```

```
lending_club_data_2012_2014 <- read_csv("lending_club_data_2012_2014_small.csv")
head(lending_club_data_2012_2014)
```

```
## # A tibble: 6 x 152
##         id member_id loan_~1 funde~2 funde~3 term  int_r~4 insta~5 grade sub_g~6
##      <dbl> <lgl>       <dbl>   <dbl>   <dbl> <chr>   <dbl>   <dbl> <chr> <chr>
## 1  3290675 NA          12175   12175   12175 36 m~   17.8     439. D     D1
## 2  1690712 NA          15000   15000   14975 36 m~   18.5     546. D     D2
## 3  1339491 NA          15000   15000   15000 36 m~   20.5     561. E     E2
## 4 13518760 NA          15000   15000   15000 36 m~    8.39    473. A     A5
## 5  6578305 NA          10000   10000   10000 36 m~    6.62    307. A     A2
## 6 16441609 NA           4000    4000    4000 36 m~   14.0     137. C     C3
## # ... with 142 more variables: emp_title <chr>, emp_length <chr>,
## #   home_ownership <chr>, annual_inc <dbl>, verification_status <chr>,
## #   issue_d <chr>, loan_status <chr>, pymnt_plan <chr>, url <chr>, desc <chr>,
## #   purpose <chr>, title <chr>, zip_code <chr>, addr_state <chr>, dti <dbl>,
## #   delinq_2yrs <dbl>, earliest_cr_line <chr>, fico_range_low <dbl>,
## #   fico_range_high <dbl>, inq_last_6mths <dbl>, mths_since_last_delinq <dbl>,
## #   mths_since_last_record <dbl>, open_acc <dbl>, pub_rec <dbl>, ...
```

## Step 2: exploring and preparing the data for modelling

An exploratory data analysis is done to get insights to the data once it has been gathered and combined. This will involve looking for any missing data, investigating the distribution of the variables, and spotting any outliers. To get the data ready for model training, data cleaning and pre processing methods including imputation, normalization, and feature engineering will be used. Here in loan status is factor with Fully Paid as level 1 and Charged Off as 0.

```
data_loan_status <- lending_club_data_2012_2014 %>%
  select(loan_amnt, funded_amnt,int_rate,installment,annual_inc,total_rec_int,last_pymnt_amnt,total_rec

data_loan_status <- data_loan_status [data_loan_status$loan_status %in% c("Fully Paid", "Charged Off"),

data_loan_status<- data_loan_status %>%
  mutate(loan_status = ifelse(loan_status == "Fully Paid",1,0),
         loan_status = as_factor(loan_status),
         term = as_factor(term),
         home_ownership = as_factor(home_ownership),
         year=as_factor(year)) %>%
  drop_na(loan_status)
```

#Spliting the data

```
data_loan_status_split <- initial_split(data_loan_status, prop = 0.75)
data_loan_status_split
```

```
## <Training/Testing/Total>
## <7287/2429/9716>
```

#missing Values

```
vis_miss(data_loan_status)
```

#recipe

```
data_loan_status_recipe <- training(data_loan_status_split) %>%
  recipe(loan_status ~ .) %>%
  step_nzv(all_predictors()) %>%
  step_rm(term, home_ownership,year) %>%
  step_impute_median(all_numeric()) %>%
  prep()
```

#Baking test

```
data_loan_status_testing <- data_loan_status_recipe %>%
  bake(testing(data_loan_status_split))
```

#juicing

```
data_loan_status_training <- juice(data_loan_status_recipe)
```

## Step 3: training a model on the data

We will use Logistic Regression to train the training data. A collection of input factors and a binary result are modeled using logistic regression to determine the connection between them. Using the logistic function, it calculates the likelihood of the result depending on the input factors. After that, based on a threshold value, the algorithm assigns the result to one of the two potential values.

### Logistic Regression

```
data_loan_status_glm <- logistic_reg(penalty = 0.001, mixture = 0.5) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(loan_status ~ ., data = data_loan_status_training)
data_loan_status_glm
```

```
## parsnip model object
##
##
## Call:  glmnet::glmnet(x = maybe_matrix(x), y = y, family = "binomial",      alpha = ~0.5)
##
##    Df  %Dev   Lambda
## 1   0  0.00 0.213200
## 2   1  0.95 0.194200
## 3   1  1.87 0.177000
## 4   2  3.11 0.161300
## 5   2  4.67 0.146900
## 6   2  6.14 0.133900
## 7   2  7.52 0.122000
## 8   2  8.83 0.111100
## 9   2 10.05 0.101300
## 10  2 11.21 0.092280
```

```
## 11  2 12.29 0.084080
## 12  2 13.31 0.076610
## 13  2 14.27 0.069800
## 14  4 15.37 0.063600
## 15  4 16.44 0.057950
## 16  5 17.44 0.052800
## 17  5 18.41 0.048110
## 18  5 19.31 0.043840
## 19  5 20.14 0.039940
## 20  6 21.12 0.036400
## 21  6 22.27 0.033160
## 22  6 23.34 0.030220
## 23  6 24.34 0.027530
## 24  6 25.26 0.025090
## 25  6 26.13 0.022860
## 26  8 26.94 0.020830
## 27  8 27.72 0.018980
## 28 10 28.47 0.017290
## 29 10 29.18 0.015750
## 30 10 29.83 0.014360
## 31 10 30.44 0.013080
## 32 11 31.28 0.011920
## 33 11 32.09 0.010860
## 34 11 32.86 0.009894
## 35 11 33.60 0.009015
## 36 11 34.30 0.008215
## 37 11 34.97 0.007485
## 38 12 35.61 0.006820
## 39 12 36.21 0.006214
## 40 12 36.79 0.005662
## 41 12 37.33 0.005159
## 42 12 37.84 0.004701
## 43 12 38.32 0.004283
## 44 12 38.78 0.003903
## 45 12 39.20 0.003556
## 46 12 39.60 0.003240
## 47 12 39.97 0.002952
## 48 12 40.32 0.002690
## 49 12 40.65 0.002451
## 50 12 40.95 0.002233
## 51 12 41.23 0.002035
## 52 12 41.49 0.001854
## 53 12 41.74 0.001689
## 54 12 41.96 0.001539
## 55 12 42.17 0.001403
## 56 12 42.36 0.001278
## 57 12 42.53 0.001164
## 58 12 42.70 0.001061
## 59 12 42.84 0.000967
## 60 12 42.98 0.000881
## 61 12 43.10 0.000803
## 62 12 43.21 0.000731
## 63 12 43.32 0.000666
## 64 12 43.41 0.000607
```

```
## 65 12 43.50 0.000553
## 66 12 43.58 0.000504
## 67 12 43.65 0.000459
## 68 12 43.71 0.000418
## 69 12 43.77 0.000381
## 70 12 43.82 0.000347
## 71 12 43.87 0.000317
## 72 12 43.91 0.000288
## 73 12 43.95 0.000263
## 74 12 43.98 0.000239
## 75 12 44.01 0.000218
## 76 12 44.04 0.000199
## 77 12 44.06 0.000181
## 78 12 44.08 0.000165
## 79 12 44.10 0.000150
## 80 12 44.12 0.000137
## 81 12 44.13 0.000125
## 82 12 44.14 0.000114
## 83 12 44.15 0.000104
## 84 12 44.16 0.000094
## 85 12 44.17 0.000086
## 86 12 44.18 0.000078
## 87 12 44.18 0.000071
## 88 12 44.19 0.000065
## 89 12 44.19 0.000059
## 90 12 44.20 0.000054
## 91 11 44.20 0.000049
## 92 11 44.20 0.000045
## 93 11 44.20 0.000041
## 94 11 44.21 0.000037
## 95 11 44.21 0.000034
## 96 11 44.21 0.000031
## 97 11 44.21 0.000028
## 98 11 44.21 0.000026
## 99 11 44.21 0.000023
```

### Step 4: evaluating model performance

The result demonstrates the usefulness and interpretability of logistic regression as a technique for classification issues. The model's accuracy is 89%, its Kappa was 0.52 and confusion matrix, ROC curve is given below. These results show how logistic regression may be used to forecast binary events.

#acuracy

```
data_loan_status_glm %>%
  predict(data_loan_status_testing) %>%
  bind_cols(data_loan_status_testing) %>%
  metrics(truth = loan_status, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.895
## 2 kap      binary         0.544
```

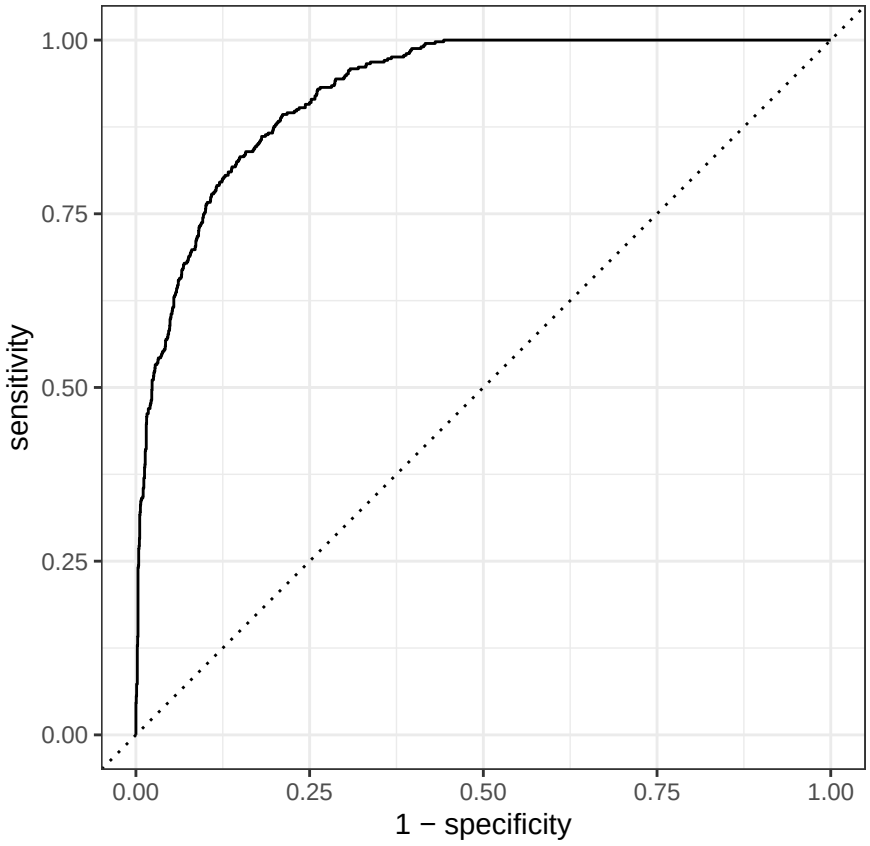
## confusion matrix

```
data_loan_status_glm %>%
  predict(data_loan_status_testing) %>%
  bind_cols(data_loan_status_testing) %>%
  conf_mat(truth = loan_status, estimate = .pred_class)
```

```
##            Truth
## Prediction    0    1
##          0  191   36
##          1  220 1982
```

# ROC

```
data_loan_status_glm %>%
  predict(data_loan_status_testing, type = "prob") %>%
  bind_cols(data_loan_status_testing) %>%
  roc_curve(loan_status, .pred_0) %>%
  autoplot()
```




## Step 5: Improving model performance

Techniques like feature selection and hyperparameter tweaking will be used to enhance the model's performance. At each stage, the model's performance will be assessed to see if it has improved.

### All Models and its accuracy

| Model | Accuracy |
|---|---|
| Null model | 82% |
| KNN | 85% |
| GLM | 89% |
| Naive Bayes | 72% |

## Null model

```
data_loan_status_null <- null_model() %>%
  set_engine("parsnip") %>%
  set_mode("classification") %>%
  fit(loan_status ~ ., data = data_loan_status_training)
data_loan_status_null
```

```
## parsnip model object
##
## Null Regression Model
## Predicted Value: 1
```

## accuracy of null model

```
data_loan_status_null %>%
  predict(data_loan_status_testing) %>%
  bind_cols(data_loan_status_testing) %>%
  metrics(truth = loan_status, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.831
## 2 kap      binary         0
```

## knn

```
data_loan_status_knn <- nearest_neighbor(neighbors = 11) %>%
  set_engine("kknn") %>%
  set_mode("classification") %>%
  fit(loan_status ~ ., data = data_loan_status_training)
data_loan_status_knn
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = loan_status ~ ., data = data, ks = min_rows(11,     data, 5))
##
## Type of response variable: nominal
## Minimal misclassification: 0.1379168
## Best kernel: optimal
## Best k: 11
```

## accuracy knn

```
data_loan_status_knn %>%
  predict(data_loan_status_testing) %>%
  bind_cols(data_loan_status_testing) %>%
  metrics(truth = loan_status, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.860
## 2 kap      binary         0.375
```

## Naive Bayes

```
data_loan_status_nb <- naive_Bayes(Laplace = 1) %>%
  set_engine("klaR") %>%
  set_mode("classification") %>%
  fit(loan_status ~ ., data = data_loan_status_training)
```

## accuracy NB

```
data_loan_status_nb %>%
  predict(data_loan_status_testing) %>%
  bind_cols(data_loan_status_testing) %>%
  metrics(truth = loan_status, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.746
## 2 kap      binary         0.313
```

# GLM

```
data_loan_status_glm <- logistic_reg(penalty = 0.001, mixture = 0.5) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(loan_status ~ ., data = data_loan_status_training)
```

# accuracy GLM

```
data_loan_status_glm %>%
  predict(data_loan_status_testing) %>%
  bind_cols(data_loan_status_testing) %>%
  metrics(truth = loan_status, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.895
## 2 kap      binary         0.544
```

## Conclusion:-

In conclusion, using input characteristics, we constructed and assessed a logistic regression model to forecast the likelihood of a loan status. Our results demonstrate that logistic regression, is a useful and understandable strategy for classification difficulties. The performance of the model can be enhanced in the future by investigating different classification techniques and incorporating further characteristics.