FINAL PROJECT

Code

Zhimei_Chen

2022-12-11

Introduction

Background

Credit card is issued to the cardholder by the bank or credit card company. The cardholder does not need to pay cash for the consumption of the credit card. The payment will be made on the billing day. Unlike debit cards, general credit cards do not deduct money directly from the user's account. Therefore, people would like to choose to apply credit card. However, not every one got approved by the bank while applying credit card. Therefore, in this project, I am going to talk about what may affects the result of the application.

GOAL:

My goal is to predict whether the applicants can be approved with credit card under the proper background? And what conditions may be important?

Reading Suggestion

“START HERE” is the place where starting to split data and make models. Where if you understand the data already, or read the data before, it saves lots of time if you start reading there. <I explain how to connect two datasets from “Combine two datasets”>

Libraries Needed:

DATA

Credit Card Approval Prediction

https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction?select=application_record.csv (https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction?select=application_record.csv) application_record.csv contains appliers personal information, which I use as features for predicting. credit_record.csv records users’ behaviors of credit card.

read csv and clean

```
## # A tibble: 6 × 18
##        id code_gender flag_own_car flag_own_realty cnt_children amt_income_total
##     <dbl> <chr>       <chr>        <chr>                  <dbl>            <dbl>
## 1 5008804 M           Y            Y                          0           427500
## 2 5008805 M           Y            Y                          0           427500
## 3 5008806 M           Y            Y                          0           112500
## 4 5008808 F           N            Y                          0           270000
## 5 5008809 F           N            Y                          0           270000
## 6 5008810 F           N            Y                          0           270000
##   name_income_type     name_education_type           name_family_status
##   <chr>                <chr>                         <chr>
## 1 Working              Higher education              Civil marriage
## 2 Working              Higher education              Civil marriage
## 3 Working              Secondary / secondary special Married
## 4 Commercial associate Secondary / secondary special Single / not married
## 5 Commercial associate Secondary / secondary special Single / not married
## 6 Commercial associate Secondary / secondary special Single / not married
##   name_housing_type days_birth days_employed flag_mobil flag_work_phone
##   <chr>                  <dbl>         <dbl>      <dbl>           <dbl>
## 1 Rented apartment      -12005         -4542          1               1
## 2 Rented apartment      -12005         -4542          1               1
## 3 House / apartment     -21474         -1134          1               0
## 4 House / apartment     -19110         -3051          1               0
## 5 House / apartment     -19110         -3051          1               0
## 6 House / apartment     -19110         -3051          1               0
##   flag_phone flag_email occupation_type cnt_fam_members
##        <dbl>      <dbl> <chr>                     <dbl>
## 1          0          0 <NA>                          2
## 2          0          0 <NA>                          2
## 3          0          0 Security staff                2
## 4          1          1 Sales staff                   1
## 5          1          1 Sales staff                   1
## 6          1          1 Sales staff                   1
```

DATA: application_data

Observations

Code

```
## Observation number in application:  438557
```

turn to characters and factors and see if there are NAs

```
##       id             code_gender flag_own_car flag_own_realty  cnt_children
##  Length:438557      F:294440    N:275459     N:134483        Min.   : 0.0000
##  Class :character   M:144117    Y:163098     Y:304074        1st Qu.: 0.0000
##  Mode  :character                                            Median : 0.0000
##                                                              Mean   : 0.4274
##                                                              3rd Qu.: 1.0000
##                                                              Max.   :19.0000
##
##  amt_income_total              name_income_type
##  Min.   :  26100   Commercial associate:100757
##  1st Qu.: 121500   Pensioner           : 75493
##  Median : 160780   State servant       : 36186
##  Mean   : 187524   Student             :    17
##  3rd Qu.: 225000   Working             :226104
##  Max.   :6750000
##
##                    name_education_type            name_family_status
##  Academic degree              :   312   Civil marriage      : 36532
##  Higher education             :117522   Married             :299828
##  Incomplete higher            : 14851   Separated           : 27251
##  Lower secondary              :  4051   Single / not married: 55271
##  Secondary / secondary special:301821   Widow               : 19675
##
##
##           name_housing_type    days_birth     days_employed    flag_mobil
##  Co-op apartment    :  1539   Min.   :-25201   Min.   :-17531   1:438557
##  House / apartment  :393831   1st Qu.:-19483   1st Qu.: -3103
##  Municipal apartment: 14214   Median :-15630   Median : -1467
##  Office apartment   :  3922   Mean   :-15998   Mean   : 60564
##  Rented apartment   :  5974   3rd Qu.:-12514   3rd Qu.:  -371
##  With parents       : 19077   Max.   : -7489   Max.   :365243
##
##  flag_work_phone flag_phone flag_email    occupation_type   cnt_fam_members
##  0:348156        0:312353   0:391102   Laborers    : 78240   Min.   : 1.000
##  1: 90401        1:126204   1: 47455   Core staff  : 43007   1st Qu.: 2.000
##                                        Sales staff: 41098   Median : 2.000
##                                        Managers    : 35487   Mean   : 2.194
##                                        Drivers     : 26090   3rd Qu.: 3.000
##                                        (Other)     : 80432   Max.   :20.000
##                                        NA's        :134203
```

The only variable with missingness is occupation_type, which is missing 134203 observations.

Trans NA to “Dont wanna tell”

Code

turn to characters and factors

```
## # A tibble: 6 × 18
##   id      code_gender flag_own_car flag_own_realty cnt_children amt_income_total
##   <chr>   <fct>       <fct>        <fct>                  <dbl>            <dbl>
## 1 5008804 M           Y            Y                          0           427500
## 2 5008805 M           Y            Y                          0           427500
## 3 5008806 M           Y            Y                          0           112500
## 4 5008808 F           N            Y                          0           270000
## 5 5008809 F           N            Y                          0           270000
## 6 5008810 F           N            Y                          0           270000
##   name_income_type     name_education_type           name_family_status
##   <fct>                <fct>                         <fct>
## 1 Working              Higher education              Civil marriage
## 2 Working              Higher education              Civil marriage
## 3 Working              Secondary / secondary special Married
## 4 Commercial associate Secondary / secondary special Single / not married
## 5 Commercial associate Secondary / secondary special Single / not married
## 6 Commercial associate Secondary / secondary special Single / not married
##   name_housing_type days_birth days_employed flag_mobil flag_work_phone
##   <fct>                  <dbl>         <dbl> <fct>      <fct>
## 1 Rented apartment      -12005         -4542 1          1
## 2 Rented apartment      -12005         -4542 1          1
## 3 House / apartment     -21474         -1134 1          0
## 4 House / apartment     -19110         -3051 1          0
## 5 House / apartment     -19110         -3051 1          0
## 6 House / apartment     -19110         -3051 1          0
##   flag_phone flag_email occupation_type  cnt_fam_members
##   <fct>      <fct>      <fct>                      <dbl>
## 1 0          0          Don't wanna tell               2
## 2 0          0          Don't wanna tell               2
## 3 0          0          Security staff                 2
## 4 1          1          Sales staff                    1
## 5 1          1          Sales staff                    1
## 6 1          1          Sales staff                    1
```

Variables

id: Client number
code_gender: Gender [M,F]
flag_own_car: Is there a car [N,Y]
flag_own_realty: Is there a property [N,Y]
cnt_children: Number of children
amt_income_total: Annual income
name_income_type: Income category [Commercial associate, Pensioner, State servant, Student, Working]
name_education_type: Education level [Academic degree, Higher education, Incomplete higher, Lower secondary, Secondary / secondary special]
name_family_status: Marital status [Civil marriage, Married, Separated, Single / not married, Widow]
name_housing_type: Way of living [Co-op apartment, House / apartment, Municipal apartment, Office apartment, Rented apartment, With parents]
days_birth: Birthday Count backwards from current day (0), -1 means yesterday
days_employed: Start date of employment Count backwards from current day(0). If positive, it means the person currently unemployed.
flag_mobile: Is there a mobile phone [0,1]
flag_work_phone: Is there a work phone [0,1]

flag_phone: Is there a phone [0,1]
flag_email: Is there an email [0,1]
occupation_type: Occupation [Laborers, Core staff, Sales staff, Managers, Drivers, (Other) , NA's]
cnt_fam_members: Family size

trans inappropriate days of employees to 0 days

Code

```
## numeric(0)
```

Code

```
## [1] 365243 365243 365243 365243 365243 365243
```

Code

DATA: credit_data

Code

```
## # A tibble: 6 × 3
##        id months_balance status
##     <dbl>          <dbl> <chr>
## 1 5001711              0 X
## 2 5001711             -1 0
## 3 5001711             -2 0
## 4 5001711             -3 0
## 5 5001712              0 C
## 6 5001712             -1 C
```

Code

```
##        id          months_balance      status
##  Min.   :5001711   Min.   :-60.00   Length:1048575
##  1st Qu.:5023644   1st Qu.:-29.00   Class :character
##  Median :5062104   Median :-17.00   Mode  :character
##  Mean   :5068286   Mean   :-19.14
##  3rd Qu.:5113856   3rd Qu.: -7.00
##  Max.   :5150487   Max.   :  0.00
```

There is no missing data in credit.

Variables

id: Client number
months_balance: Record month The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the previous month, and so on
status: Status (*more in "transfer status"*)

Observations

Code

```
## Observation number in credit:  1048575
```

transfer status

0: 1-29 days past due –> 1
1: 30-59 days past due –> 2
2: 60-89 days overdue –> 3
3: 90-119 days overdue –> 4
4: 120-149 days overdue –> 5
5: Overdue or bad debts, write-offs for more than 150 days –> 6
C: paid off that month –> 0
X: No loan for the month –> 0

```
## # A tibble: 6 × 3
##   id      months_balance status
##   <chr>            <dbl> <chr>
## 1 5001711              0 0
## 2 5001711             -1 1
## 3 5001711             -2 1
## 4 5001711             -3 1
## 5 5001712              0 0
## 6 5001712             -1 0
```

turn to factors

Code

```
## # A tibble: 6 × 3
##   id      months_balance status
##   <chr>            <dbl> <fct>
## 1 5001711              0 0
## 2 5001711             -1 1
## 3 5001711             -2 1
## 4 5001711             -3 1
## 5 5001712              0 0
## 6 5001712             -1 0
```

first year credit

Code

```
## # A tibble: 6 × 4
## # Groups:   id [2]
##   id      months_balance status month
##   <chr>            <dbl> <fct>  <int>
## 1 5001711             -3 1          1
## 2 5001711             -2 1          2
## 3 5001711             -1 1          3
## 4 5001711              0 0          4
## 5 5001712            -18 1          1
## 6 5001712            -17 1          2
```

Combine two datasets

How we combine?

Example

An example here who, with ID: 5008805, applied credit card and had been approved.

Code

```
## # A tibble: 12 × 4
## # Groups:   id [1]
##    id      months_balance status month
##    <chr>            <dbl> <fct>  <int>
##  1 5008805            -14 0          1
##  2 5008805            -13 1          2
##  3 5008805            -12 2          3
##  4 5008805            -11 0          4
##  5 5008805            -10 0          5
##  6 5008805             -9 0          6
##  7 5008805             -8 0          7
##  8 5008805             -7 0          8
##  9 5008805             -6 0          9
## 10 5008805             -5 0         10
## 11 5008805             -4 0         11
## 12 5008805             -3 0         12
```

Code

```
## # A tibble: 15 × 3
##         id months_balance status
##      <dbl>          <dbl> <chr>
##  1 5008805              0 C
##  2 5008805             -1 C
##  3 5008805             -2 C
##  4 5008805             -3 C
##  5 5008805             -4 C
##  6 5008805             -5 C
##  7 5008805             -6 C
##  8 5008805             -7 C
##  9 5008805             -8 C
## 10 5008805             -9 C
## 11 5008805            -10 C
## 12 5008805            -11 C
## 13 5008805            -12 1
## 14 5008805            -13 0
## 15 5008805            -14 X
```

Code

```
## # A tibble: 1 × 18
##   id      code_gender flag_own_car flag_own_realty cnt_children amt_income_total
##   <chr>   <fct>       <fct>        <fct>                  <dbl>            <dbl>
## 1 5008805 M           Y            Y                          0           427500
##   name_income_type name_education_type name_family_status name_housing_type
##   <fct>            <fct>               <fct>              <fct>
## 1 Working          Higher education    Civil marriage     Rented apartment
##   days_birth days_employed flag_mobil flag_work_phone flag_phone flag_email
##        <dbl>         <dbl> <fct>      <fct>           <fct>      <fct>
## 1     -12005         -4542 1          1               0          0
##   occupation_type  cnt_fam_members
##   <fct>                      <dbl>
## 1 Don't wanna tell               2
```

Therefore, we can combine application data with first year, first month credit data

Code

```
## Unique ID in first year credit:  45985 and in credit:  45985
```

Code

```
## Unique ID in application:  438510
```

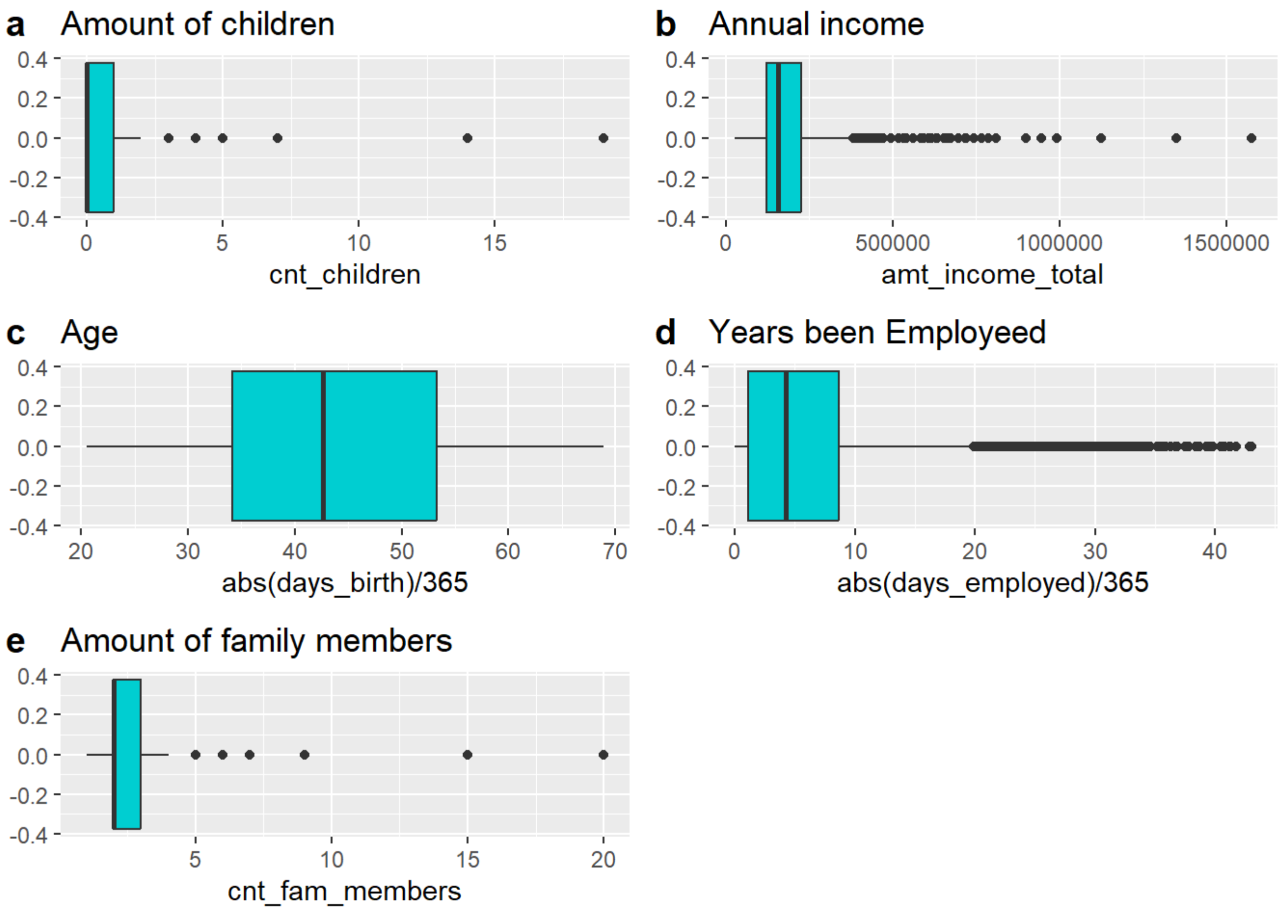
Code

```
## 36457 people has credit card with Info here.
```

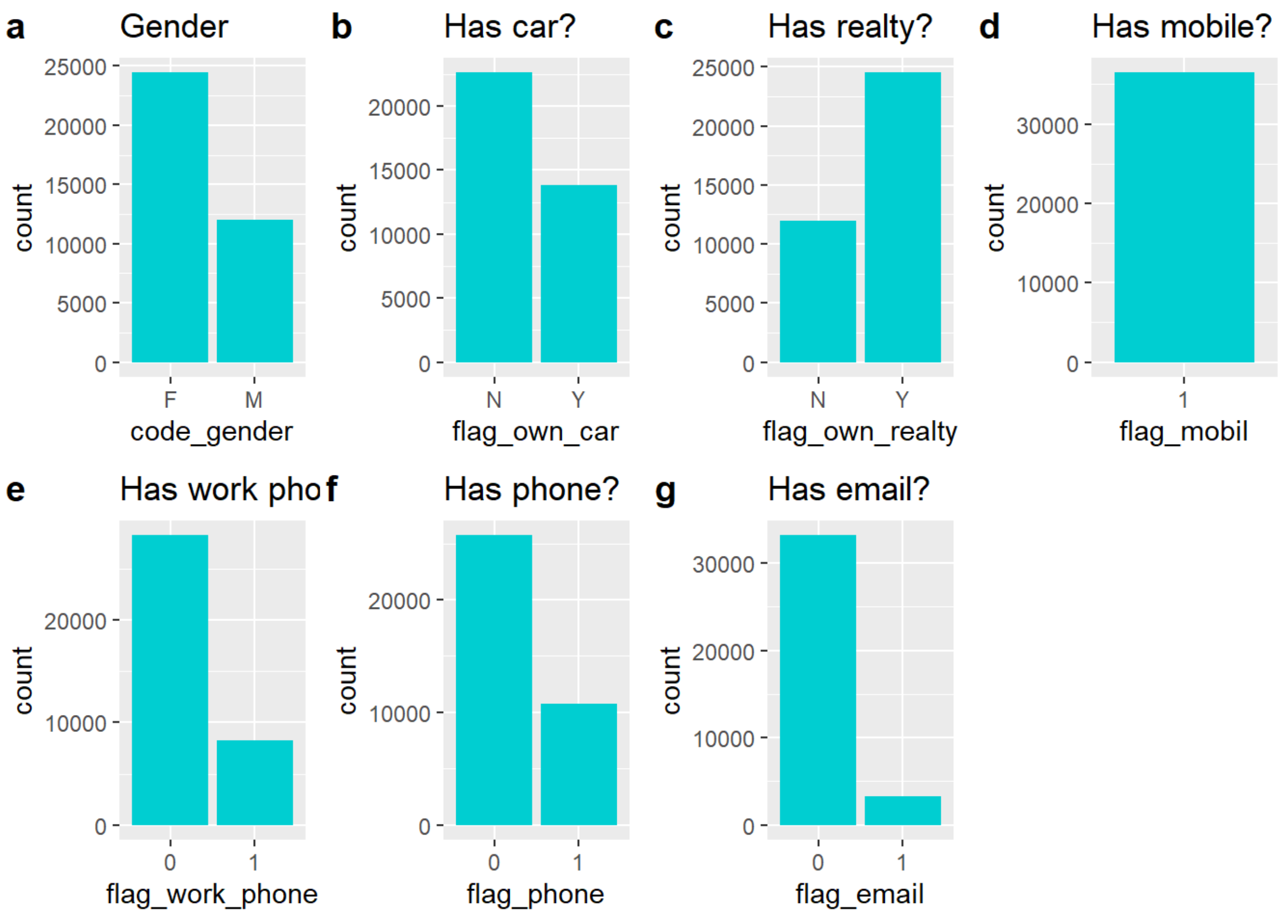
Code

Observe - graphs

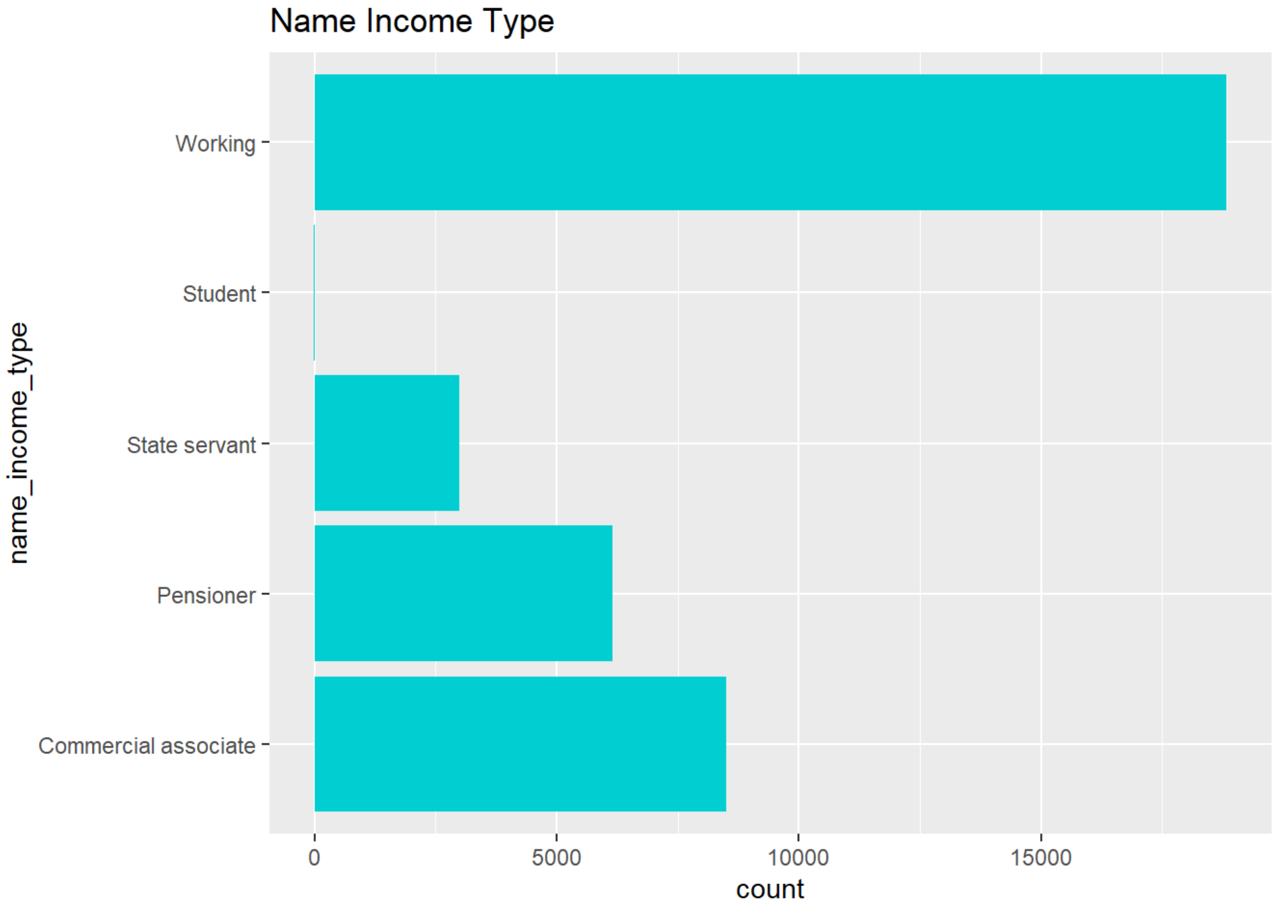
those had been approved's information (numbers)

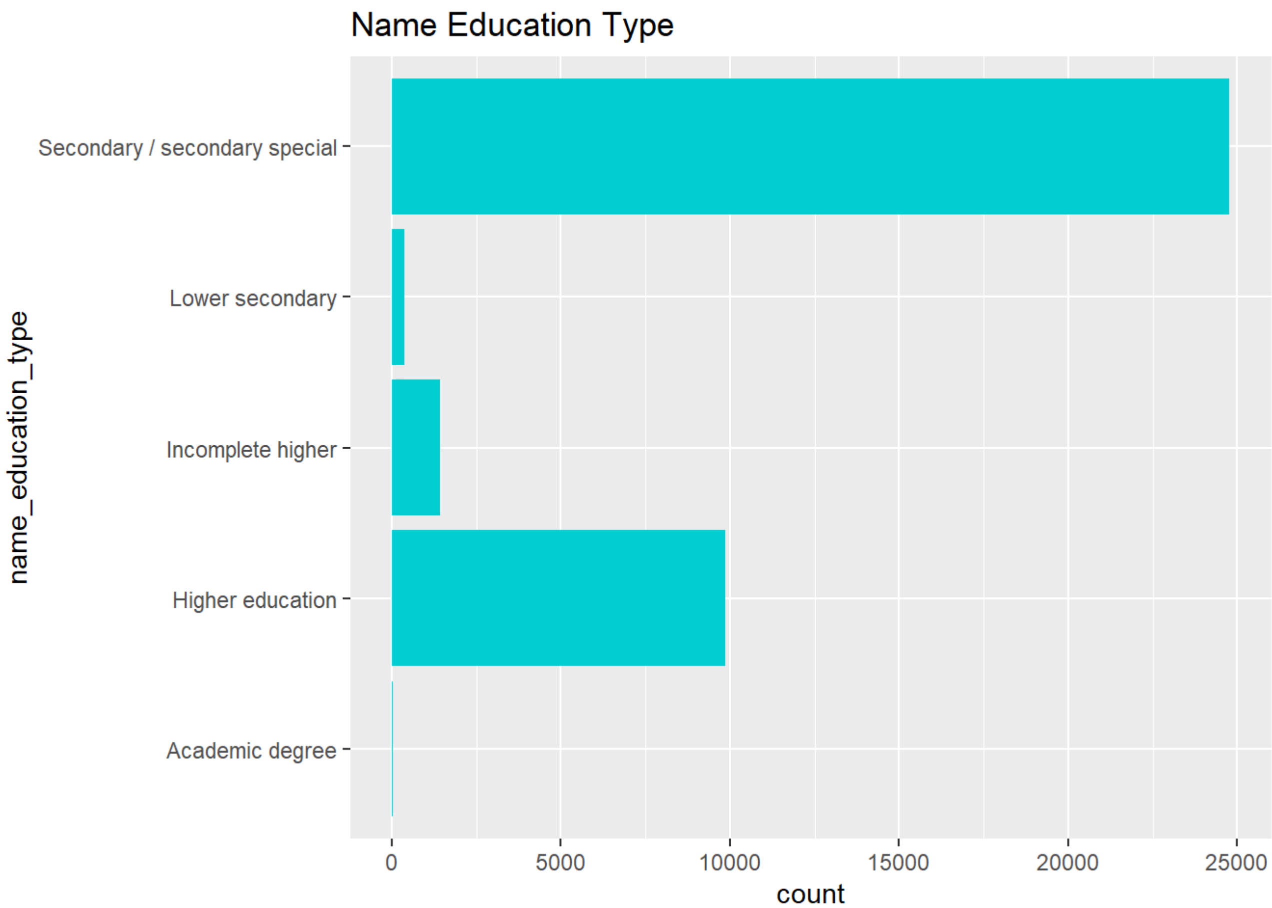


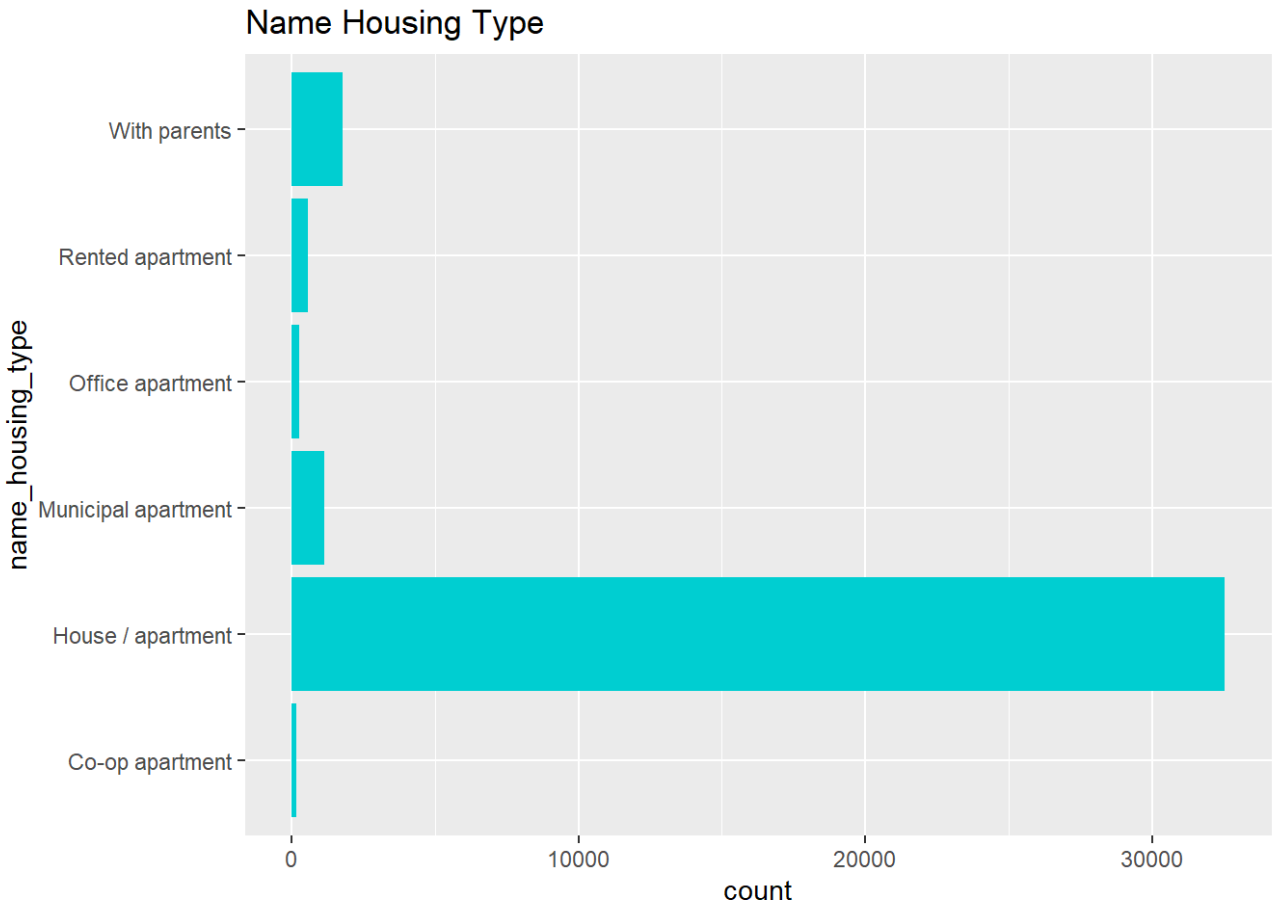
those had been approved's information (factors - YES/NO)

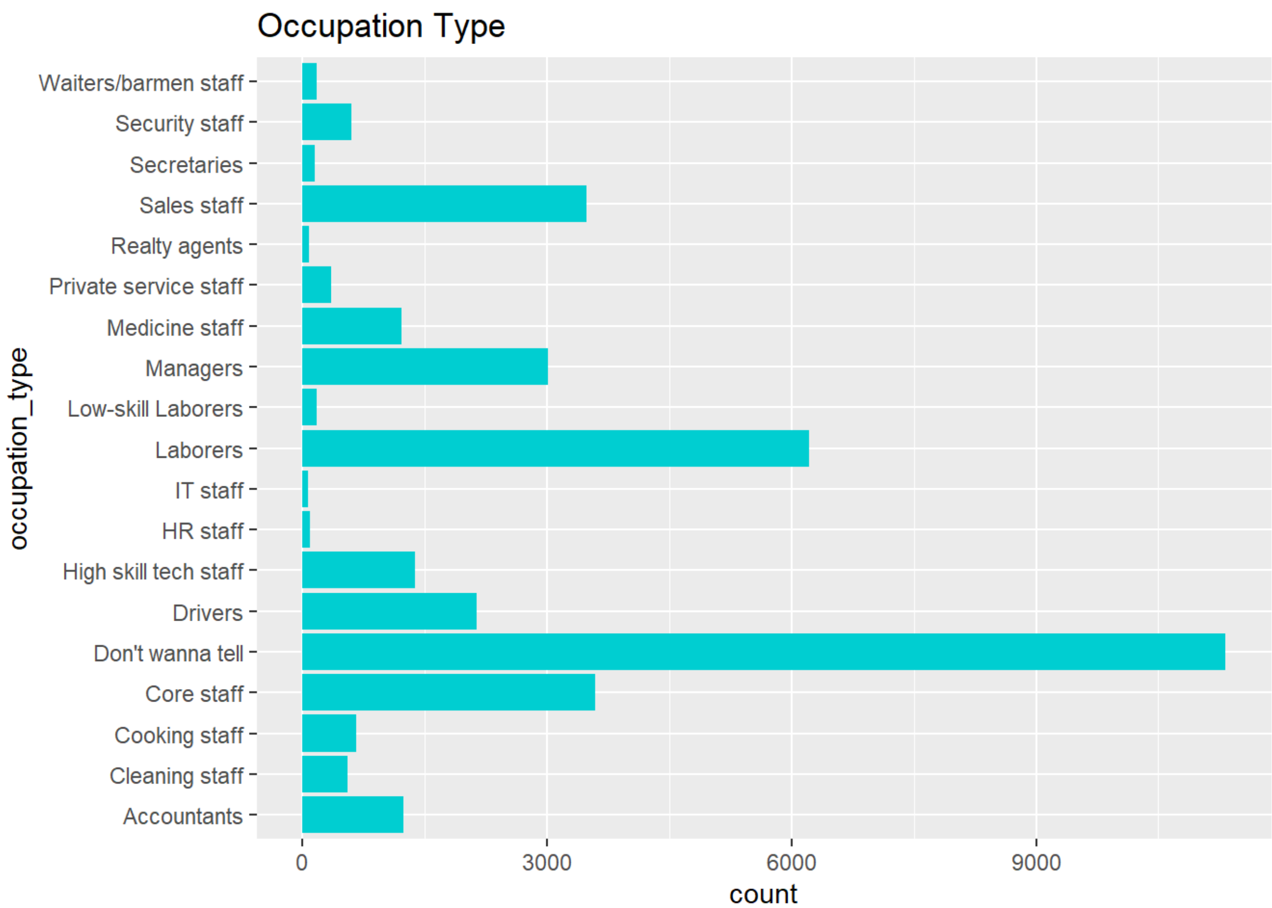


those had been approved's information (factors - REST)









Compare numbers of ID

Code

```
## 45985 45985 438510 36457
```

My understanding to this data is according to the 45985 people who have credit cards, 36457 people from those 45985 people are applied with recording the backgrounds. The bank issued credit cards to those 36457 people from 438510 applicants.

Combine and add our y

Code

```
##    Mode   FALSE    TRUE
## logical  402100   36457
```

There are 402053 people who do not have credit card, while the other 36457 people have.

Code

```
## 438510 is less than 438557
```

Finding out there are duplicated rows in mydata, since when unique id should be 43510, where we have 438557 rows of mydata, so I remove them:

Remove duplicated rows & turn logistics to factors

Code

START HERE!

Split Data

Code

```
## [1] 306957     19
```

Code

```
## [1] 131553     19
```

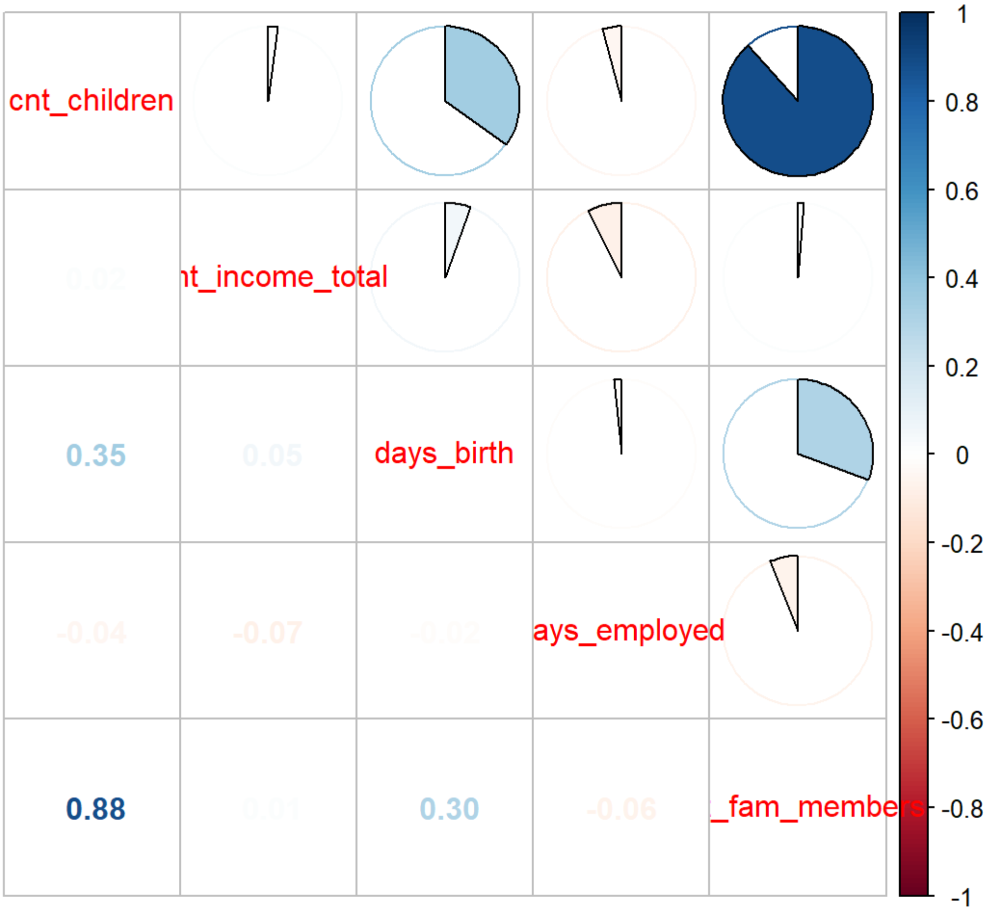
Code

Correlation Matrix

Using the training data set, I create a correlation matrix of all continuous variables.

Code

```
## Warning: Use of bare predicate functions was deprecated in tidyselect 1.1.0.
## ℹ Please use wrap predicates in `where()` instead.
##   # Was:
##   data %>% select(is.numeric)
##
##   # Now:
##   data %>% select(where(is.numeric))
```



KNN

Code

In Train

Code

```
##           true
## predicted  FALSE   TRUE
##     FALSE 280279   6284
##     TRUE    1145  19249
```

Code

```
## [1] 0.02420209
```

In Test

Code

```
##           true
## predicted  FALSE   TRUE
##     FALSE 118640   4685
##     TRUE    1989   6239
```

Code

```
## [1] 0.0507324
```

Conclusion

Both error rates in train and test are low, it’s nice. However, we can only use numeric variables for predictors in KNN model, so I use other predictors in the following models.

Recipe

Code

Logistic Regression Model

Code

Accuracy

Generate predictions using logistic regression model and training data

Code

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.917
```

Fit the model to the testing data

Code

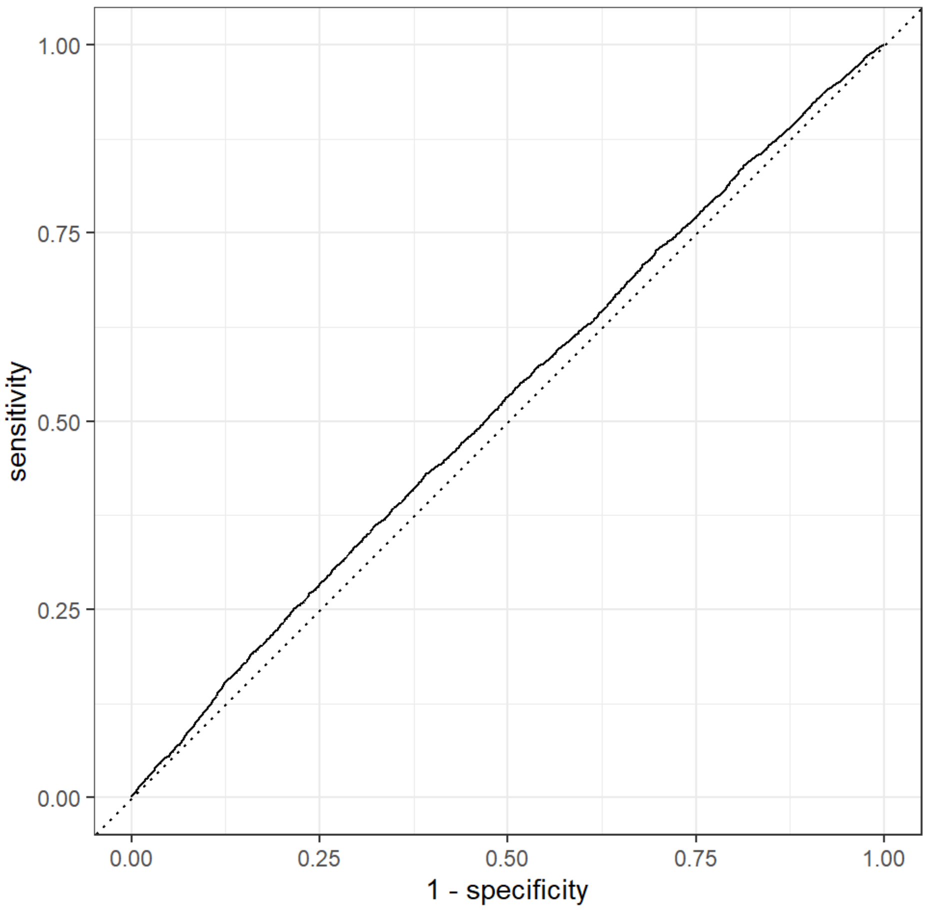
```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.917
```

Heatmap

Code

ROC Curve

Code

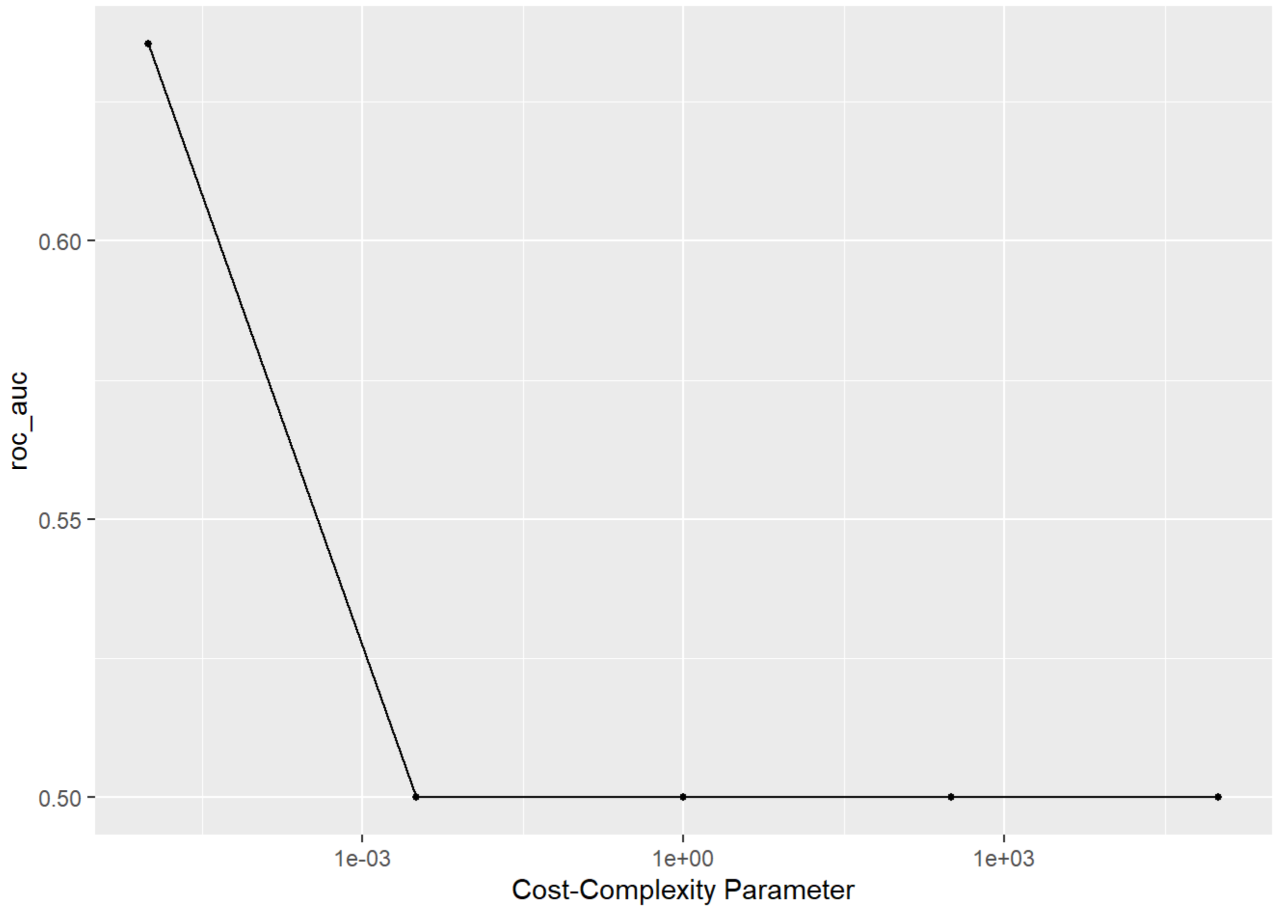


Decision Tree Model

Code

####tune & autoplot

Code



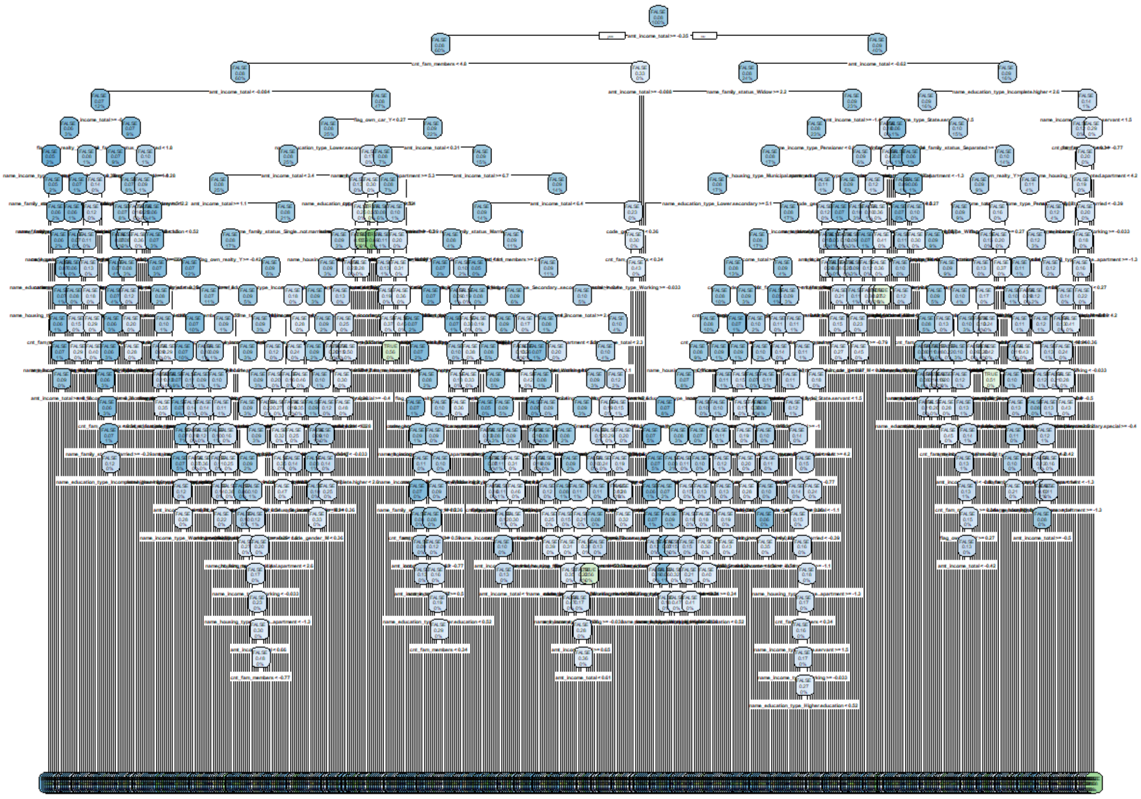
Best-performing pruned decision tree

Code

```
## # A tibble: 1 × 7
##   cost_complexity .metric .estimator  mean     n std_err .config
##             <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1         0.00001 roc_auc binary     0.635     5 0.00261 Preprocessor1_Model1
```

Rpart-Plot

Code

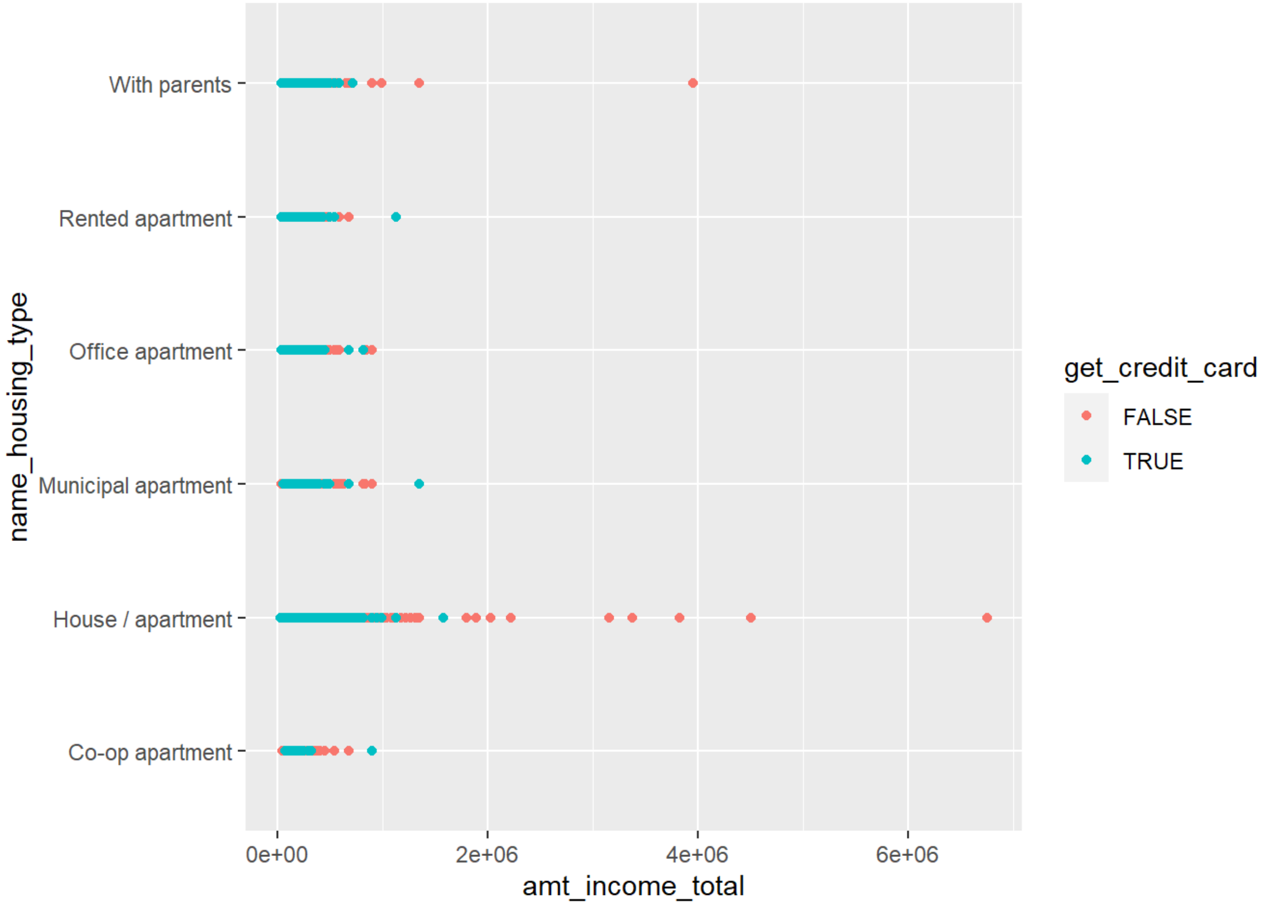


It is overfitting.

SVM

Plot

Code



SVM works too slow so I take less train data and test data to fit this model.

Split Data2

Code

```
## [1] 3069   19
```

Code

```
## [1] 1316   19
```

Code

recipe2

(Only change the size of the data)

Code

model & workflow

Code

tune

Code

```
## ! Fold1: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

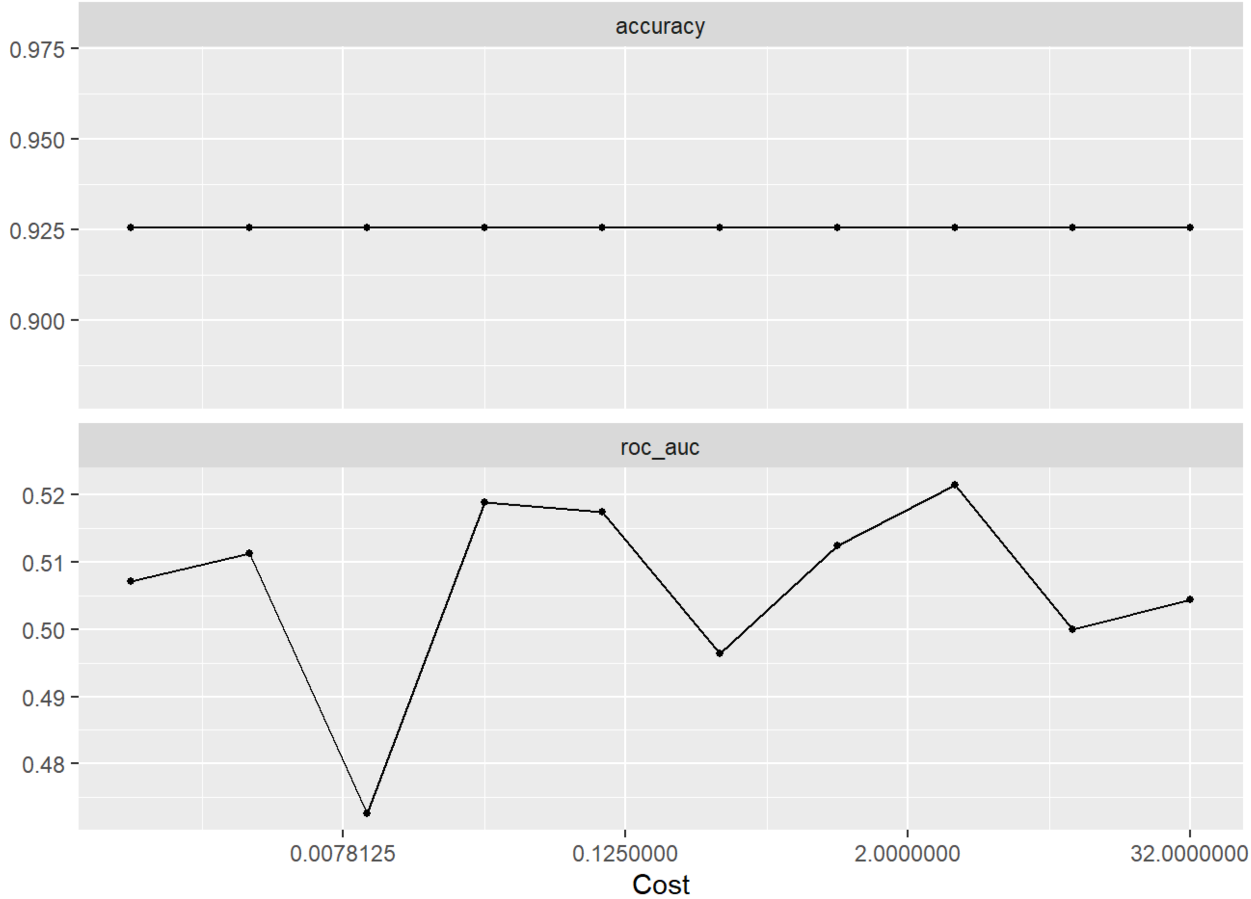
```
## ! Fold2: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

```
## ! Fold3: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

```
## ! Fold4: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

```
## ! Fold5: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

Code



Select the Best

Code

```
## maximum number of iterations reached -0.0008357573 -0.0008383165
```

Fit to test

Code

```
##           Truth
## Prediction FALSE TRUE
##      FALSE  1226   90
##      TRUE      0    0
```

It predicts everything FALSE, so that this model is not good.

Random Forest Model

Code

grid

Code

tune

Code

```
## Warning: The `...` are not used in this function but one or more objects were
## passed: 'gird', 'metric'
```

```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

```
## Warning: Column(s) have zero variance so scaling cannot be used:
## `name_income_type_Student`. Consider using `step_zv()` to remove those columns
## before normalizing
```

```
## ! Fold1: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

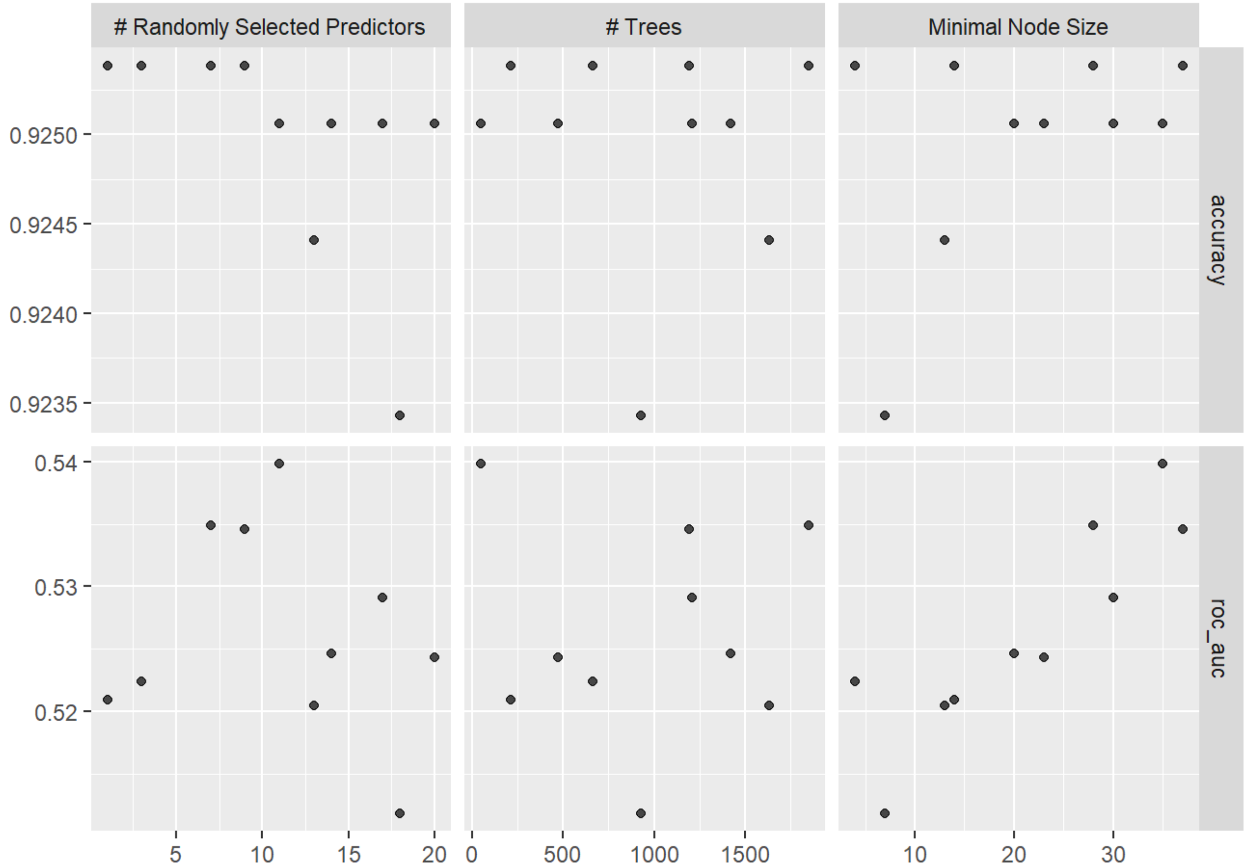
```
## ! Fold2: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

```
## ! Fold3: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

```
## ! Fold4: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

```
## ! Fold5: preprocessor 1/1: Column(s) have zero variance so scaling cannot be used: `na
me_income_typ...
```

Code



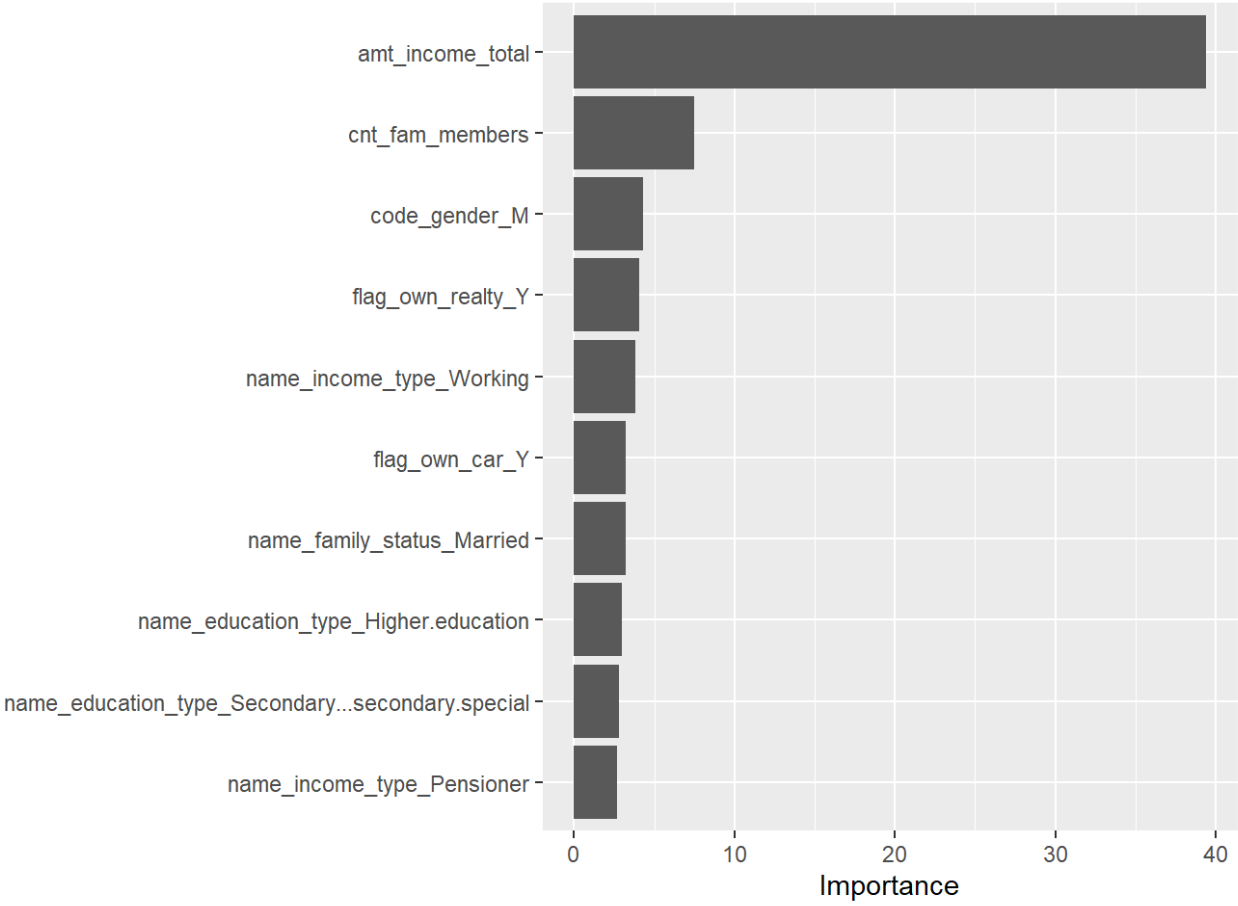
The main limitation of random forest is that a large number of trees can make the algorithm too slow and ineffective for real-time predictions. It takes almost an hour to do 6/10 of this step. Therefore, as SVM model, I choose 1 percent of the whole data to fit.

select best random forest

Code

```
## Warning: Column(s) have zero variance so scaling cannot be used:
## `name_income_type_Student`. Consider using `step_zv()` to remove those columns
## before normalizing
```

Code



Variable amt_income_total (amount of year income) is the most useful.

Conclusion

Approach GOAL?

Recall: My goal is to predict whether the applicants can be approved with credit card under the proper background? And what conditions may be important?
I think my models are not that good to predict whether the applicants can be approved under the background, though some of the models did better than others. And the numeric conditions are more important because the KNN model uses only numeric variables and it works the best.

Models

KNN

KNN works good because both mean squared errors for train data and test data are small.

Logistic Regression

Though the accuracy is about 0.91, I think this model is not good enough. The number looks good just because of there are many people got rejected from the application. ROC curve shows that it is not good enough.

Decision Tree

Decision Tree is not good enough because it is too much overriding. A good decision tree should between underiding and overriding. The situation of a small change in the data can cause a large change in the structure of the decision tree causing instability happens in my model.

SVM

It takes too long, so I take 1 percent of the original data to do. The model directly predict everything FALSE, which why I think it is not good.

Random Forest Tree

I finished the code for random Forest Tree, but it is too big, where I take 1 percent of the original data to do as SVM. I got that variable amt_income_total (amount of year income) is the most useful.

Possible Reasons & May Improved in the future:

Why some models are not good?
TOO MANY useless predictors. There are too many predictors which make the model not accurate enough. Therefore, I will try to make fewer predictors to make the models fit better.