Model card approval

Loading used packages

```
library(tidyverse)
library(tidymodels)
library(plotROC)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
```

Loading data

```
application <- read_csv("application_record.csv",
                        col_types = cols(ID = col_character(),
                                         CODE_GENDER = col_character(),
                                         FLAG_OWN_CAR = col_character(),
                                         FLAG_OWN_REALTY = col_character(),
                                         CNT_CHILDREN = col_double(),
                                         AMT_INCOME_TOTAL = col_double(),
                                         NAME_INCOME_TYPE = col_character(),
                                         NAME_EDUCATION_TYPE = col_character(),
                                         NAME_FAMILY_STATUS = col_character(),
                                         NAME_HOUSING_TYPE = col_character(),
                                         DAYS_BIRTH = col_double(),
                                         DAYS_EMPLOYED= col_double(),
                                         FLAG_MOBIL = col_character(),
                                         FLAG_WORK_PHONE = col_character(),
                                         FLAG_PHONE = col_character(),
                                         FLAG_EMAIL = col_character(),
                                         OCCUPATION_TYPE = col_character(),
                                         CNT_FAM_MEMBERS = col_double()))

records <- read_csv("credit_record.csv",col_types = cols(ID = col_character(),
                                                         MONTHS_BALANCE = col_double(),
                                                         STATUS = col_character()))
# Convert variable STATUS to more useful format.
records <-
records %>%
  mutate(STATUS = case_when(
    STATUS == 0 ~ "1-29_DPD",
    STATUS == 1 ~ "30-59_PDP",
    STATUS == 2 ~ "60-89_DPD",
    STATUS == 3 ~ "90-119_DPD",
    STATUS == 4 ~ "120-149_DPD",
    STATUS == 5 ~ "150+_DPD_Write-off",
    STATUS == "C" ~ "Paid_off",
```

```
    STATUS == "X" ~ "No_loan"),
    STATUS = as_factor(STATUS))
```

Vintage analysis of delinquency

Calculate issuance date and convert negative months on book to positive value for more consistent approach

```
records <-
records %>%
  left_join(
      records %>%
        select(ID, MONTHS_BALANCE) %>%
          group_by(ID) %>%
            mutate(ISSUANCE_MONTH = min(MONTHS_BALANCE)) %>%
              select(ID, ISSUANCE_MONTH) %>%
                distinct(),
  by = "ID")
records <-
records %>%
  mutate(MONTHS_BALANCE = abs(ISSUANCE_MONTH) - abs(MONTHS_BALANCE))
```

Calculate first mounth when the borrower was found as "Bad" - 60 days past due(DPD)

```
first_bad_month <-
records %>%
  filter(STATUS %in% c("60-89_DPD", "90-119_DPD", "120-149_DPD",
                       "150+_DPD_Write-off")) %>%
  select(ID, MONTHS_BALANCE) %>%
  group_by(ID) %>%
  summarise(FIRST_BAD_MONTH = min(MONTHS_BALANCE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Add IS_BAD as flag for indicating customers as BAD if he get 60+ DPD in some date in past

```
records <-
records %>%
  left_join(first_bad_month, by = "ID") %>%
  mutate(IS_BAD = if_else(MONTHS_BALANCE >= FIRST_BAD_MONTH, 1, 0),
         IS_BAD = replace_na(IS_BAD, 0),
         FIRST_BAD_MONTH = NULL)
```

Divide issuance credits by groups 6 months

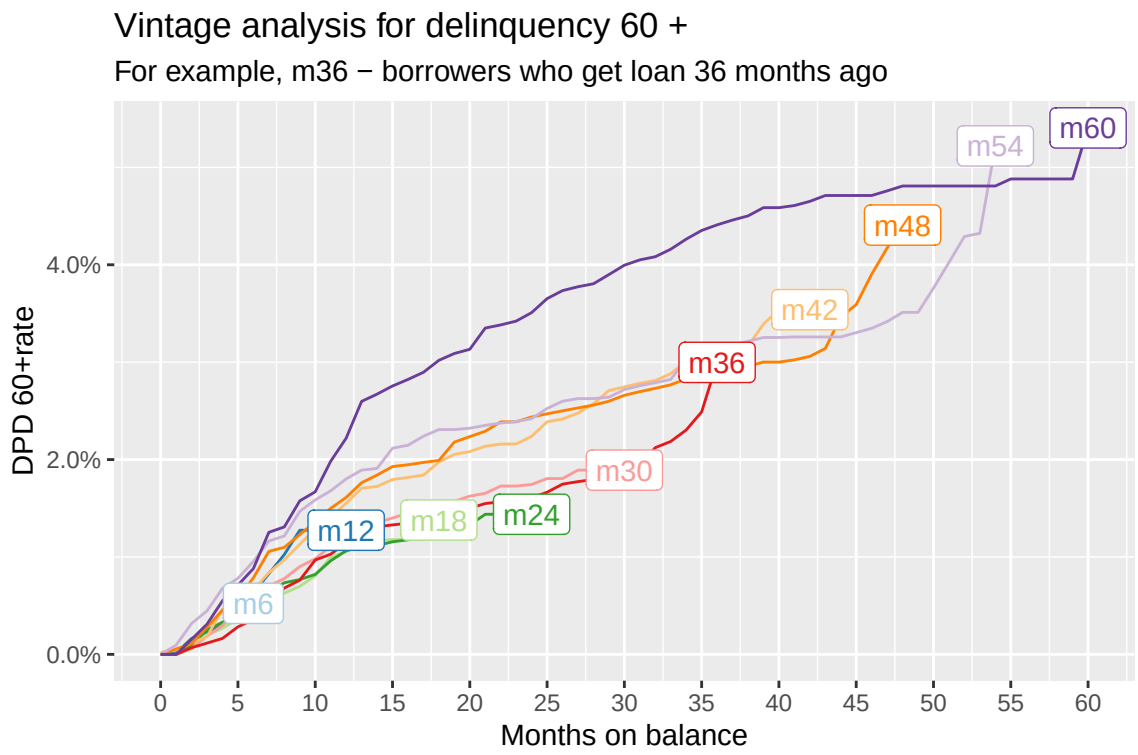
```
records <-
records %>%
  mutate(ISSUANCE_MONTH = abs(ISSUANCE_MONTH),
         ISSUANCE_PERIOD = case_when(
    ISSUANCE_MONTH >= 0 & ISSUANCE_MONTH <= 6 ~ "m6",
    ISSUANCE_MONTH > 6 & ISSUANCE_MONTH <= 12 ~ "m12",
    ISSUANCE_MONTH > 12 & ISSUANCE_MONTH <= 18 ~ "m18",
```

```
    ISSUANCE_MONTH > 18 & ISSUANCE_MONTH <= 24 ~ "m24",
    ISSUANCE_MONTH > 24 & ISSUANCE_MONTH <= 30 ~ "m30",
    ISSUANCE_MONTH > 30 & ISSUANCE_MONTH <= 36 ~ "m36",
    ISSUANCE_MONTH > 36 & ISSUANCE_MONTH <= 42 ~ "m42",
    ISSUANCE_MONTH > 42 & ISSUANCE_MONTH <= 48 ~ "m48",
    ISSUANCE_MONTH > 48 & ISSUANCE_MONTH <= 54 ~ "m54",
    ISSUANCE_MONTH > 54 & ISSUANCE_MONTH <= 60 ~ "m60")) %>%
  mutate(ISSUANCE_PERIOD = factor(ISSUANCE_PERIOD,
                                  ordered = T,
                                  levels = c("m6", "m12", "m18",
                                             "m24", "m30", "m36",
                                             "m42", "m48", "m54",
                                             "m60"
                                             )))
```

Vintage plot for 60+ DPD (no cumulative)

```
plt_60_dpd <-
  # transform data and calculate vintages
  records %>%
  group_by(ISSUANCE_PERIOD, MONTHS_BALANCE) %>%
  summarise(n = n(), NUMBER_OF_BAD = sum(IS_BAD))%>%
  mutate(BAD_RATE = NUMBER_OF_BAD / n) %>%
  select(-c(n, NUMBER_OF_BAD)) %>%
  mutate(BAD_RATE_CUM = cummax(BAD_RATE))

plt_60_dpd <-
  # make a basic plot for vintages
  plt_60_dpd %>%
  ggplot(aes(x = MONTHS_BALANCE, y = BAD_RATE_CUM, color = ISSUANCE_PERIOD))+
  geom_line()

plt_60_dpd <-
  # change scales of x variable, color and y variable
  plt_60_dpd +
  scale_x_continuous(breaks = seq(from = 0, to = 60, by = 5))+
  scale_color_brewer(palette = "Paired")+
  scale_y_continuous(labels = scales::percent)

plt_60_dpd <-
  # add notation for axes and delete legend for additional free space on graph
  plt_60_dpd+
  xlab(label = "Months on balance")+
  ylab(label = "DPD 60+rate")+
  theme(legend.position = "none")+
  guides(color = guide_legend("Issuance months ago"))

# calculate statistics for plot
plt_60_dpd_stat <- records %>%
        group_by(ISSUANCE_PERIOD, MONTHS_BALANCE) %>%
        summarise(n = n(), NUMBER_OF_BAD = sum(IS_BAD))%>%
        mutate(BAD_RATE = NUMBER_OF_BAD / n) %>%
        select(-c(n, NUMBER_OF_BAD)) %>%
        mutate(BAD_RATE_CUM = cummax(BAD_RATE))%>%
```

```
        group_by(ISSUANCE_PERIOD)%>%
        summarise(max_dpd = max(BAD_RATE_CUM))

plt_60_dpd_stat <- left_join(plt_60_dpd_stat, records %>%
            select(ISSUANCE_PERIOD, MONTHS_BALANCE) %>%
            group_by(ISSUANCE_PERIOD)%>%
            summarise(MONTHS_BALANCE = max(MONTHS_BALANCE)), by = "ISSUANCE_PERIOD")
plt_60_dpd_stat <- plt_60_dpd_stat %>%
  rename(BAD_RATE_CUM = max_dpd)
# add title and labels for plot

plt_60_dpd <-
  plt_60_dpd +
  geom_label(data=plt_60_dpd_stat, aes(label=ISSUANCE_PERIOD))+
  ggtitle("Vintage analysis for delinquency 60 +",
          subtitle = "For example, m36 - borrowers who get loan 36 months ago")

plt_60_dpd
```



Data preparation

Let's introduce Default criteria as days past due = 60 plus, based on conservative principles. Usually 90 plus or other unlikely to pay criteria are triggers of default , but for current analytical purposes 60 plus delinquency appear suitable.

```
application <-
left_join(application, first_bad_month, by = "ID") %>%
```

```
  mutate(IS_DEFAULT = if_else(is.na(FIRST_BAD_MONTH), 0, 1),
         FIRST_BAD_MONTH = NULL,
         IS_DEFAULT = factor(IS_DEFAULT, levels = c(1, 0), labels = c("Default", "Non_Default")))
```

Create variable AGE

```
application <-
application %>%
  mutate(TODAY = as.Date("2020-01-01"), #this is hypothetical day of analysis
         BIRTH_DATE = TODAY + DAYS_BIRTH,
         AGE = as.numeric((round((TODAY - BIRTH_DATE)/365,0)))) %>%
  mutate(TODAY = NULL, BIRTH_DATE = NULL)
```

Create variable working experience

```
application <-
application %>%
  mutate(WORK_EXP = case_when(
    DAYS_EMPLOYED == 365243 & AGE >= 50 ~ AGE - 20, # this is pensioners
    DAYS_EMPLOYED != 365243 ~ round(abs(DAYS_EMPLOYED)/365,0),
    TRUE ~ 0))
```

Delete variables with zero dispersion and collinear with other variables

```
application <-
application %>%
  mutate(DAYS_BIRTH = NULL,
         DAYS_EMPLOYED = NULL,
         FLAG_MOBIL = NULL)
```

Convert NA in occupation type to class “No work” for keep conservative

```
application <-
application %>%
  mutate(OCCUPATION_TYPE = replace_na(OCCUPATION_TYPE, "No work"))
```

Model building

let’s consider distribution of defualts in dataset. Data is very unbalanced.

```
table(application$IS_DEFAULT)
```

```
##
##     Default Non_Default
##         616      437941
```

For further steps let’s decide which strategy will be used: 1.Using data as is 2.Using subsampling for decrease majority set For solve question above, build two baseline models with resampling and without using logistic regression

```
#NAME_INCOME_TYPE: #student --> Working
#OCCUPATION_TYPE: #HR staff, Realty agents --> Managers
application <-
application %>%
  mutate(NAME_INCOME_TYPE = if_else(NAME_INCOME_TYPE == "Student",
                                    "Working",
                                    NAME_INCOME_TYPE),
         OCCUPATION_TYPE = if_else(OCCUPATION_TYPE %in% c("HR staff", "Realty agents"),
                                   "Managers",
                                   OCCUPATION_TYPE))
```

```
application <-
application %>%
  mutate_at(.vars = c("CODE_GENDER","FLAG_OWN_CAR","FLAG_OWN_REALTY",
                      "NAME_INCOME_TYPE", "NAME_EDUCATION_TYPE",
                      "NAME_FAMILY_STATUS","NAME_HOUSING_TYPE",
                      "FLAG_WORK_PHONE","FLAG_PHONE","FLAG_EMAIL",
                      "OCCUPATION_TYPE"), as_factor)
```

Creat subsample with proportion of defaulted borrowers as 30/70

```
set.seed(123)
application_shrinked <-
bind_rows(
  # all defaulted borrowers
  application %>%
  filter(IS_DEFAULT == "Default"),
  # non defaulted sample
  application %>%
    filter(IS_DEFAULT == "Non_Default") %>%
    slice_sample(n = 2000)
  )
application_shrinked <-
application_shrinked %>%
  slice_sample(n = nrow(application_shrinked))
```

Divide data to test and train samples

```
set.seed(123)
data_split <- initial_split(application_shrinked, prop = 3/4, strata = IS_DEFAULT)
train_data <- training(data_split)
test_data  <- testing(data_split)
```

Using tidy-models approach let's build random forest model

```
# Create recepie
application_rec <-
  recipe(IS_DEFAULT ~ ., data = train_data) %>%
  update_role(ID, new_role = "ID")

# Make model
application_rf_model <-
```

```
rand_forest(trees = 1000) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification")

# Make cross-validation sample
set.seed(123)
application_folds <- vfold_cv(v = 10,data = train_data)

# Make workflow
application_wf <-
  workflow()%>%
  add_model(application_rf_model)%>%
  add_recipe(application_rec)

# Estimate model using cross-validation
set.seed(123)
application_cross_validation <-
application_wf %>%
  fit_resamples(application_folds, metrics = metric_set(accuracy,
                                                        precision,
                                                        recall,
                                                        roc_auc))
```

Pull calculated metrics on 10-folds cross-validation

```
collect_metrics(application_cross_validation)
```

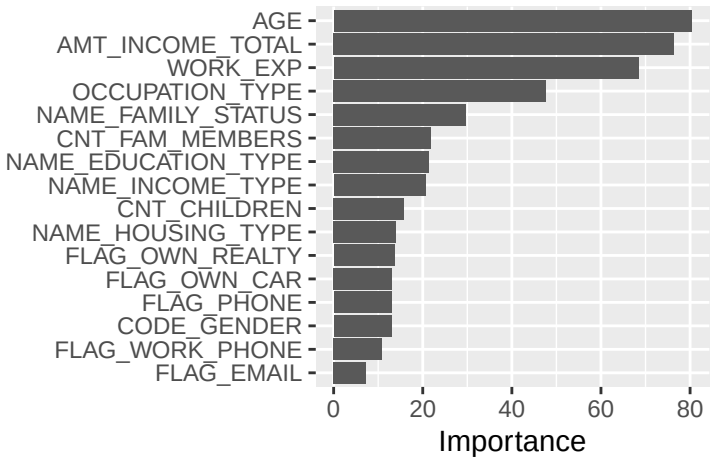
```
## # A tibble: 4 x 6
##   .metric   .estimator  mean     n std_err .config
##   <chr>     <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary     0.827    10 0.00925 Preprocessor1_Model1
## 2 precision binary     0.877    10 0.0267  Preprocessor1_Model1
## 3 recall    binary     0.310    10 0.0184  Preprocessor1_Model1
## 4 roc_auc   binary     0.730    10 0.0107  Preprocessor1_Model1
```

Estimate performance on test sample

```
# Fit model for evaluation on test data
set.seed(123)
application_fitted <-
  application_wf %>%
  fit(data = train_data)
```

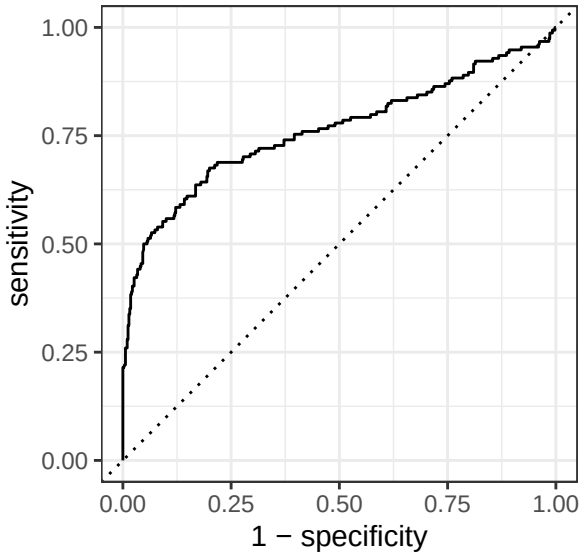
Create plot for variable importance

```
application_wf %>%
  last_fit(data_split) %>%
  pluck(".workflow", 1) %>%
  pull_workflow_fit() %>%
  vip::vip(num_features = 20)
```



Make ROC-curve

```
application_fitted %>%
  predict(new_data = test_data)%>%
  bind_cols(predict(application_fitted,new_data = test_data, type = "prob"))%>%
  bind_cols(test_data %>% select(IS_DEFAULT)) %>%
  roc_curve(IS_DEFAULT, .pred_Default) %>%
  autoplot()
```



Calculate basic metrics

```
application_prediction_rf_01 <-
  application_fitted %>%
  predict(new_data = test_data)%>%
  bind_cols(predict(application_fitted,new_data = test_data, type = "prob"))%>%
  bind_cols(test_data %>% select(IS_DEFAULT))
multi_metric <- metric_set(accuracy, precision, recall)
bind_rows(
  multi_metric(application_prediction_rf_01,
```

```
              truth = IS_DEFAULT, estimate = .pred_class),
  roc_auc(application_prediction_rf_01, IS_DEFAULT, .pred_Default))
```

```
## # A tibble: 4 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 accuracy  binary         0.824
## 2 precision binary         0.882
## 3 recall    binary         0.292
## 4 roc_auc   binary         0.762
```