Loan Approval Status

Homework 3 Data 622 Section 2 Group 5

Bonnie Cooper, Orli Khaimova, Leo Yi

2021-10-24

Contents

Introduction

We will be working with a dataset of loan approval status information. The task is to develop models to predict loan approval status with the given feature variables. After a preliminary exploratory data analysis, we will fit Linear Discriminant, K-Nearest Neighbors, Decision Trees and Random Forest models to a subset of the data and evaluate performance on a hold-out data set.

Import Data

To begin, the following code will import the data and R necessary libraries:

```
library(tidyr)
library(dplyr)
library(ggplot2)
library(VIM)
library(corrplot)
library(purrr)
library(scales)
library(caret)
library(Hmisc)
library(naniar)
library(rattle)

# import data
url <- 'https://raw.githubusercontent.com/SmilodonCub/DATA622/master/Loan_approval.csv'
df <- read.csv(url, header=T, na.strings="")
```

Exploratory Data Analysis

The following code will quantitatively and visually explore the nature of the loan approval dataset. We begin by describing the dataset features:

```
# convert column names to lowercase
names(df) <- lapply(names(df), tolower)
names(df)
```

```
##  [1] "loan_id"           "gender"            "married"
##  [4] "dependents"        "education"         "self_employed"
##  [7] "applicantincome"   "coapplicantincome" "loanamount"
## [10] "loan_amount_term"  "credit_history"    "property_area"
## [13] "loan_status"
```

Use `dplyr`'s `glimpse()` function to take a quick look at the data structure. Followed by `Hmisc`'s `describe()` function to return some basic summary statistics about the dataframe features:

```
# quick look at what the data structure looks like
glimpse(df)
```

```
## Rows: 614
## Columns: 13
## $ loan_id           <chr> "LP001002", "LP001003", "LP001005", "LP001006", "LP0~
## $ gender            <chr> "Male", "Male", "Male", "Male", "Male", "Male", "Mal~
## $ married           <chr> "No", "Yes", "Yes", "Yes", "No", "Yes", "Yes", "Yes"~
## $ dependents        <chr> "0", "1", "0", "0", "0", "2", "0", "3+", "2", "1", "~
## $ education         <chr> "Graduate", "Graduate", "Graduate", "Not Graduate", ~
## $ self_employed     <chr> "No", "No", "Yes", "No", "No", "Yes", "No", "No", "N~
## $ applicantincome   <int> 5849, 4583, 3000, 2583, 6000, 5417, 2333, 3036, 4006~
## $ coapplicantincome <dbl> 0, 1508, 0, 2358, 0, 4196, 1516, 2504, 1526, 10968, ~
## $ loanamount        <int> NA, 128, 66, 120, 141, 267, 95, 158, 168, 349, 70, 1~
## $ loan_amount_term  <int> 360, 360, 360, 360, 360, 360, 360, 360, 360, 360, 36~
## $ credit_history    <int> 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, NA, ~
## $ property_area     <chr> "Urban", "Rural", "Urban", "Urban", "Urban", "Urban"~
## $ loan_status       <chr> "Y", "N", "Y", "Y", "Y", "Y", "Y", "N", "Y", "N", "Y~
```

```
# summary of each field
describe(df)
```

```
## df
##
##  13  Variables      614  Observations
## --------------------------------------------------------------------------------
## loan_id
##        n  missing distinct
##      614        0      614
##
## lowest : LP001002 LP001003 LP001005 LP001006 LP001008
## highest: LP002978 LP002979 LP002983 LP002984 LP002990
## --------------------------------------------------------------------------------
## gender
##        n  missing distinct
##      601       13        2
##
## Value      Female   Male
## Frequency     112    489
## Proportion  0.186  0.814
```

```
## ----------------------------------------------------------------------------------
## married
##        n  missing distinct
##      611        3        2
##
## Value         No   Yes
## Frequency    213   398
## Proportion 0.349 0.651
## ----------------------------------------------------------------------------------
## dependents
##        n  missing distinct
##      599       15        4
##
## Value          0     1     2    3+
## Frequency    345   102   101    51
## Proportion 0.576 0.170 0.169 0.085
## ----------------------------------------------------------------------------------
## education
##        n  missing distinct
##      614        0        2
##
## Value          Graduate Not Graduate
## Frequency           480          134
## Proportion        0.782        0.218
## ----------------------------------------------------------------------------------
## self_employed
##        n  missing distinct
##      582       32        2
##
## Value         No   Yes
## Frequency    500    82
## Proportion 0.859 0.141
## ----------------------------------------------------------------------------------
## applicantincome
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      614        0      505        1     5403     4183     1898     2216
##      .25      .50      .75      .90      .95
##     2878     3812     5795     9460    14583
##
## lowest :   150   210   416   645   674, highest: 39147 39999 51763 63337 81000
## ----------------------------------------------------------------------------------
## coapplicantincome
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      614        0      287    0.912     1621     2118        0        0
##      .25      .50      .75      .90      .95
##        0     1188     2297     3782     4997
##
## lowest :     0.00    16.12   189.00   240.00   242.00
## highest: 10968.00 11300.00 20000.00 33837.00 41667.00
## ----------------------------------------------------------------------------------
## loanamount
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      592       22      203        1    146.4    79.57     56.0     71.0
##      .25      .50      .75      .90      .95
```

```
##    100.0    128.0    168.0    235.8    297.8
##
## lowest :   9  17  25  26  30, highest: 500 570 600 650 700
## --------------------------------------------------------------------------------
## loan_amount_term
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      600       14       10    0.378      342    43.83      180      294
##      .25      .50      .75      .90      .95
##      360      360      360      360      360
##
## lowest :  12  36  60  84 120, highest: 180 240 300 360 480
##
## Value         12    36    60    84   120   180   240   300   360   480
## Frequency      1     2     2     4     3    44     4    13   512    15
## Proportion 0.002 0.003 0.003 0.007 0.005 0.073 0.007 0.022 0.853 0.025
## --------------------------------------------------------------------------------
## credit_history
##        n  missing distinct     Info      Sum     Mean      Gmd
##      564       50        2    0.399      475   0.8422   0.2663
##
## --------------------------------------------------------------------------------
## property_area
##        n  missing distinct
##      614        0        3
##
## Value          Rural Semiurban     Urban
## Frequency        179       233       202
## Proportion     0.292     0.379     0.329
## --------------------------------------------------------------------------------
## loan_status
##        n  missing distinct
##      614        0        2
##
## Value          N     Y
## Frequency    192   422
## Proportion 0.313 0.687
## --------------------------------------------------------------------------------
```

From this output, we can summarize each dataset feature as follows:

1. `loan_id` (ordinal): each entry is a unique value, therefore this feature is not informative for loan status
2. `gender` (categorical): 2 distinct values with missing data
3. `married` (categorical): 2 distinct values with missing data
4. `dependents` (categorical): 4 distinct values with missing data
5. `education` (categorical): 2 distinct values, no missing data
6. `self_employed` (categorical): 2 distinct values with missing data
7. `applicantincome` (numeric): value range, no missing data
8. `coapplicantincome` (numeric): value range, no missing data
9. `loanamount` (numeric): value range with missing data
10. `loan_amount_term` (numeric): relatively few unique values (10) with missing data
11. `credit_history` (categorical): 2 distinct values with missing data
12. `property_area` (categorical): 3 distinct values, no missing data
13. `loan_status` (categorical): 2 distinct values, no missing data

Removing `loan_id`: this feature was found to have as many unique values as there are rows in the dataframe.

`loan_id` is a record (row) identification label, therefore, we will drop this feature from the data:

```r
# remove loan ID
df <- df %>%
  select(-loan_id)
```

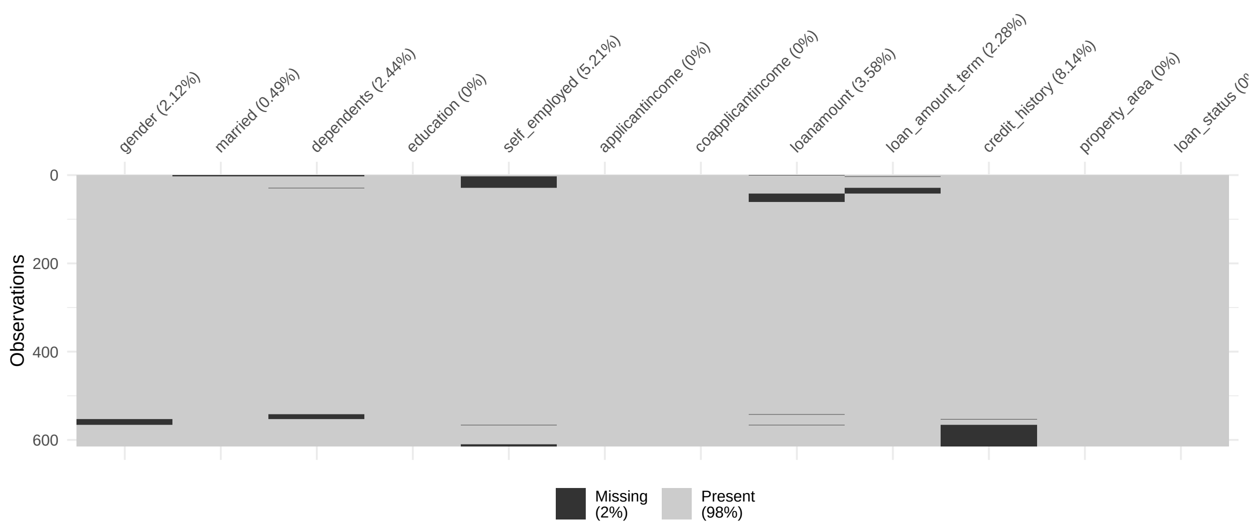
Missing Values

The output from `describe()` reveals that many of the features have missing values. Here we use `naniar`'s `miss_var_summary()` and `vis_miss()` functions to summarize and visualize the missing values in the features of the dataset:

```r
# return a summary table of the missing data in each column
miss_var_summary(df)
```

```
## # A tibble: 12 x 3
##    variable          n_miss pct_miss
##    <chr>              <int>    <dbl>
##  1 credit_history        50    8.14
##  2 self_employed         32    5.21
##  3 loanamount            22    3.58
##  4 dependents            15    2.44
##  5 loan_amount_term      14    2.28
##  6 gender                13    2.12
##  7 married                3    0.489
##  8 education              0    0
##  9 applicantincome        0    0
## 10 coapplicantincome      0    0
## 11 property_area          0    0
## 12 loan_status            0    0
```

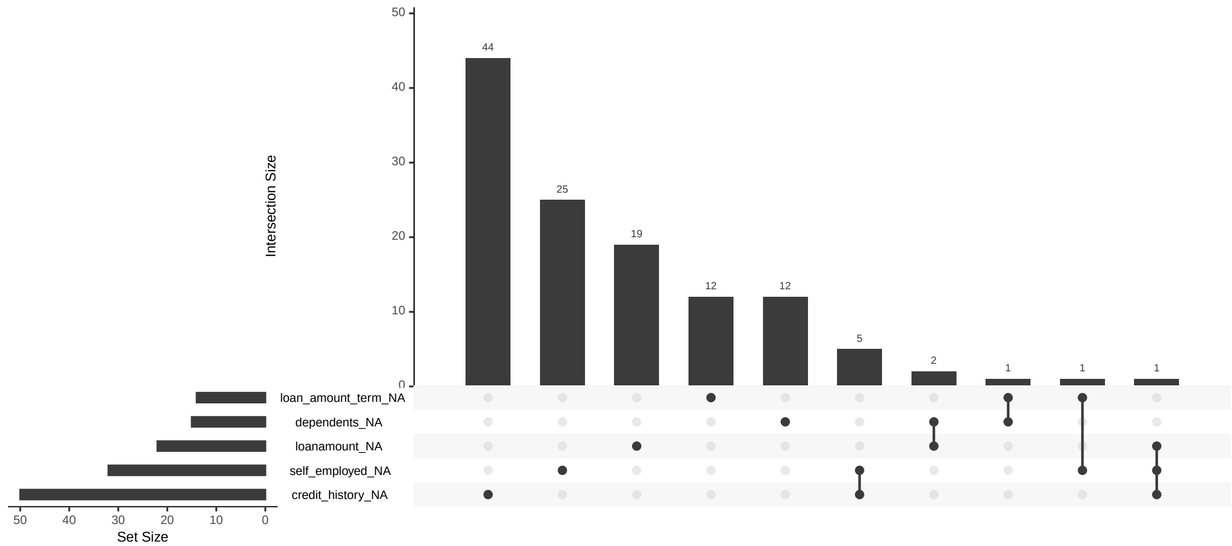
```r
# visualize the amount of missing data for each feature
vis_miss( df, cluster = TRUE )
```



The figure above shows a grouped view of the missing values in each feature column. Overall, 2% of the values are missing from the dataset. Several features have no missing values (`education`, `applicantincome`, and `coapplicantincome`). Many of the features have relatively few missing values. However, the `credit_history` is missing 8.14% of the data; a substantial amount.

Explore the missing data further by using the `gg_miss_upset()` function to show patterns correlated missing values.

```
gg_miss_upset( df )
```



The figure above shows that the vast majority of rows only have a singleton missing value; this is represented by the 5 bars in the left of the plot with only one dot to indicate the missing feature. However, a small minority or rows have 2-3 missing elements indicated by multiple, connected dots under the 5 bars to the right side of the plot.

Since there are relatively few rows with multiple missing values, it would not adversely affect the power of the analysis to remove them. Imputation will be used to adress the remaining missing values.

```
# create a vector holding the sum of NAs for each row
count_na <- apply( df, 1, function(x) sum(is.na(x)))
# keep only the rows with less than 2 missing values
df <- df[count_na < 2,]
dim( df )
```

```
## [1] 601  12
```

For a simple approximation, we will use the `simputation` package[1] to fill NA values for categorical and numeric features with 'hot-deck' imputation (i.e. a values pulled at random from complete cases in the dataset).

```
# single imputation analysis
df <- bind_shadow( df ) %>%
  data.frame() %>%
  simputation::impute_rhd(., credit_history ~ 1 ) %>%
  simputation::impute_rhd(., loan_amount_term ~ 1 ) %>%
  simputation::impute_rhd(., loanamount ~ 1 ) %>%
  simputation::impute_rhd(., self_employed ~ 1 ) %>%
  simputation::impute_rhd(., gender ~ 1 ) %>%
  simputation::impute_rhd(., dependents ~ 1 ) %>%
  tbl_df()  %>%
  select( -c(13:24) )
```

Confirm that we have filled all `NA` values:

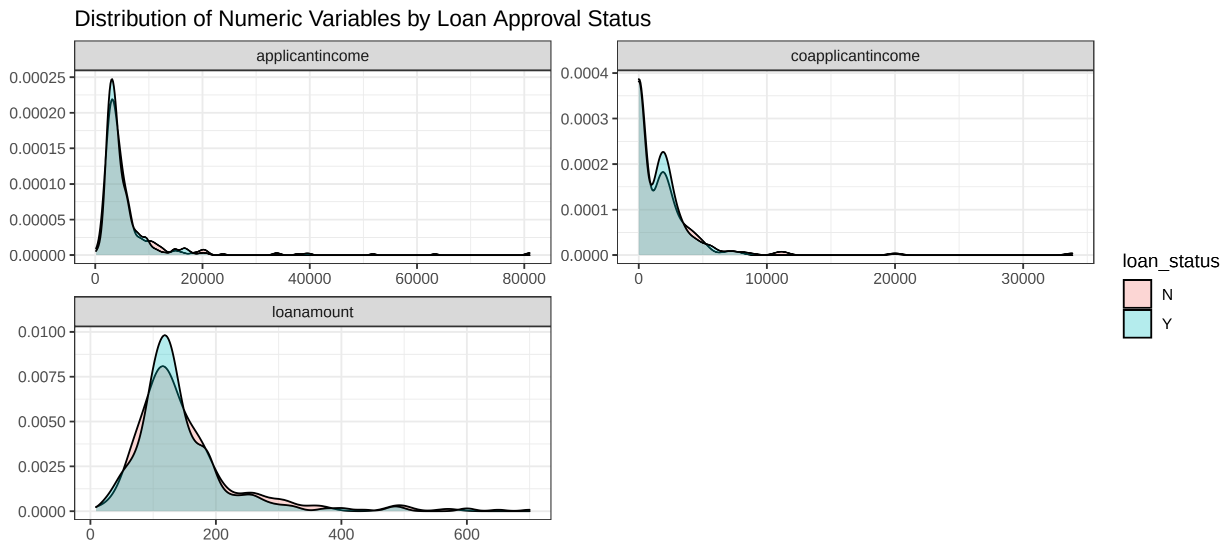
```r
# return a summary table of the missing data in each column
miss_var_summary(df)
```

```
## # A tibble: 12 x 3
##    variable          n_miss pct_miss
##    <chr>              <int>    <dbl>
##  1 gender                 0        0
##  2 married                0        0
##  3 dependents             0        0
##  4 education              0        0
##  5 self_employed          0        0
##  6 applicantincome        0        0
##  7 coapplicantincome      0        0
##  8 loanamount             0        0
##  9 loan_amount_term       0        0
## 10 credit_history         0        0
## 11 property_area          0        0
## 12 loan_status            0        0
```

Distributions of Numeric Variables

Now that the missing values have been imputed across the dataframe, we can explore the relationships of the variables in more depth. To start we visualize the distributions of the numeric variables grouped by the outcome of the target variable (`loan_status`):

```r
# numeric distributions
df %>%
  select(applicantincome, coapplicantincome, loanamount ) %>%
  bind_cols(select(df, loan_status)) %>%
  gather(var, val, -loan_status) %>%
  ggplot(aes(x = val, fill = loan_status)) +
  geom_density(alpha = .3) +
  facet_wrap(~var, scales = 'free', ncol = 2) +
  theme_bw() +
  labs(x = element_blank(),
       y = element_blank(),
       title = 'Distribution of Numeric Variables by Loan Approval Status'
       )
```



The distributions do not suggest any obviously significant differences when grouped by the target variable for any of the numeric features. It does not appear to be likely that either of these 3 features are correlated to `loan_status`. This can be confirmed with ANOVA[2]:

```
# ANOVA for applicantincome
applicantincome.aov <- aov(applicantincome ~ loan_status, data = df)
# Summary of the analysis
summary(applicantincome.aov)
```

```
##                Df      Sum Sq   Mean Sq F value Pr(>F)
## loan_status     1     1555165   1555165   0.041   0.84
## Residuals     599 22732449155 37950666
```

```
# ANOVA for coapplicantincome
coapplicantincome.aov <- aov(coapplicantincome ~ loan_status, data = df)
# Summary of the analysis
summary(coapplicantincome.aov)
```

```
##                Df     Sum Sq Mean Sq F value Pr(>F)
## loan_status     1    3676222 3676222   0.612  0.434
## Residuals     599 3600016273 6010044
```

```
# ANOVA for applicantincome
loanamount.aov <- aov(loanamount ~ loan_status, data = df)
# Summary of the analysis
summary(loanamount.aov)
```

```
##                Df  Sum Sq Mean Sq F value Pr(>F)
## loan_status     1    2099    2099   0.291   0.59
## Residuals     599 4320046    7212
```

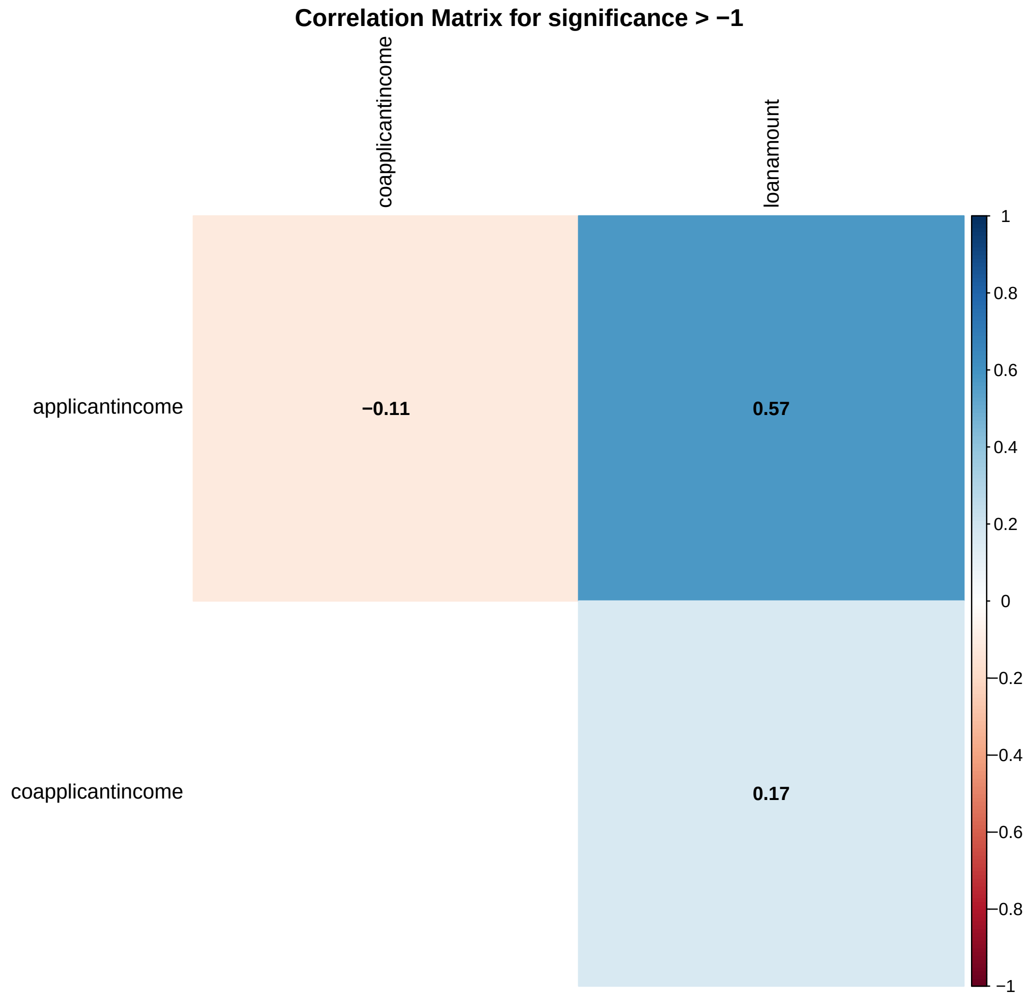
The p-values for all three ANOVA tests are very high indicating that there is no significant relationship between the features variables and the target.

Correlation of Numeric Variables

Here we can look for correlations between feature variables

```
df_numeric <- df %>%
  select(applicantincome, coapplicantincome, loanamount )
```

```r
plot_corr_matrix(df_numeric, -1)
```



We can see a strong positive correlation between the features `applicantincome` and `loanamount`. There is a weak positive correlation between `coapplicantincome` and `loanamount`. Interestingly there is a weak negative correlation between `applicantincome` and `coapplicantincome`; presumably due to a high-earning family being able to sustain with a single income.

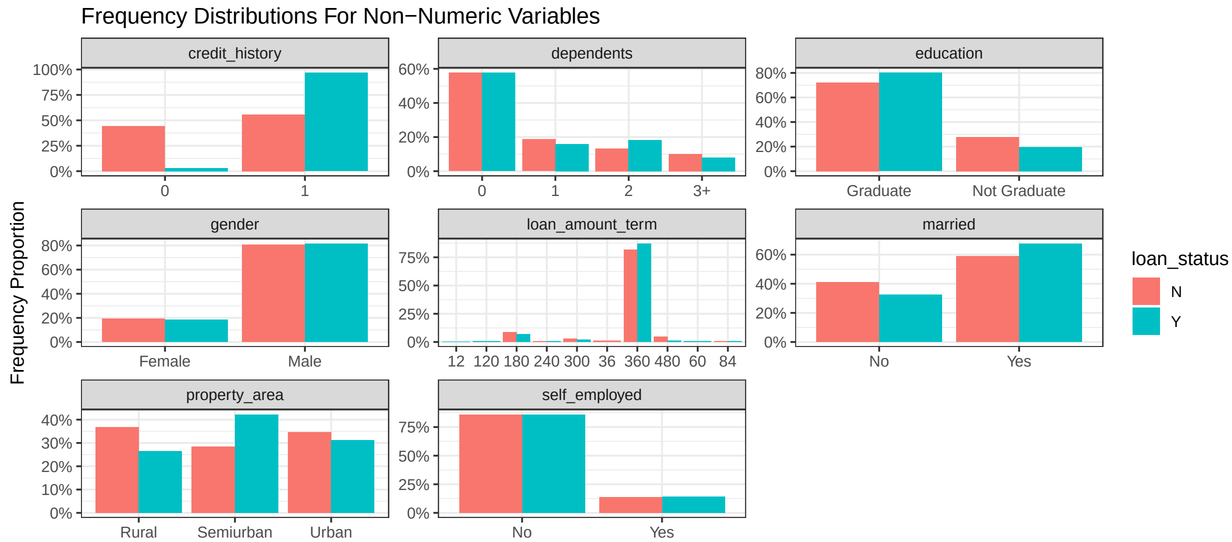
Distributions of Categorical Variables

No we turn to the categorical features to see if there are any strong relationships with the target variable. The following code will visualize the proportions of each target variable level for each level of a given feature:

```r
yes_count <- sum(df$loan_status == 'Y')
no_count <- sum(df$loan_status == 'N')

df %>%
  select(-applicantincome, -coapplicantincome, -loanamount ) %>%
  gather(var, value, -loan_status) %>%
  group_by(var, value, loan_status) %>%
  summarise(count = n(),
            .groups = 'drop') %>%
  mutate(prop = count / ifelse(loan_status == 'Y', yes_count, no_count)) %>%
  ggplot(aes(x = value, y = prop, fill = loan_status)) +
```

```r
  geom_col(position = 'dodge') +
  facet_wrap(~var, scales = 'free') +
  theme_bw() +
  labs(y = 'Frequency Proportion',
       x = element_blank(),
       title = 'Frequency Distributions For Non-Numeric Variables') +
  scale_y_continuous(labels = percent_format(accuracy = 1))
```



When interpreting the categorical bar plots, differences between `loan_status` for a given feature-level suggest that a relationship exists between a feature and the target variable. For example, we see a clear difference between the Y/N bars for `credit_history`, `married` and `property_area`. However, there is little difference for the levels of `gender` and no noticeable difference for `self_employed`.

The existence of a significant relationship between the categorical features and the target variable can be evaluated with a Chi-square test[3].

```r
cat_features <- c( 'self_employed', 'gender', 'dependents', 'loan_amount_term', 'education', 'property_a
for(feature in cat_features){print( feature ); print( chisq.test(table(df[[feature]], df$loan_status)))
```

```
## [1] "self_employed"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(df[[feature]], df$loan_status)
## X-squared = 4.9889e-29, df = 1, p-value = 1
##
## [1] "gender"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(df[[feature]], df$loan_status)
## X-squared = 0.021725, df = 1, p-value = 0.8828
##
## [1] "dependents"
##
##  Pearson's Chi-squared test
##
```

```
## data:  table(df[[feature]], df$loan_status)
## X-squared = 3.2112, df = 3, p-value = 0.3602
##
## [1] "loan_amount_term"
##
##  Pearson's Chi-squared test
##
## data:  table(df[[feature]], df$loan_status)
## X-squared = 15.579, df = 9, p-value = 0.07621
##
## [1] "education"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(df[[feature]], df$loan_status)
## X-squared = 4.3088, df = 1, p-value = 0.03792
##
## [1] "property_area"
##
##  Pearson's Chi-squared test
##
## data:  table(df[[feature]], df$loan_status)
## X-squared = 11.718, df = 2, p-value = 0.002854
##
## [1] "married"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(df[[feature]], df$loan_status)
## X-squared = 4.0097, df = 1, p-value = 0.04524
##
## [1] "credit_history"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(df[[feature]], df$loan_status)
## X-squared = 160.21, df = 1, p-value < 2.2e-16
```

From the results of the Chi-square test, only the following features have a statistically significant relation ($\alpha = 0.05$) to the target:

- `credit_history`
- `married`
- `property_area`
- `education`

We will move forward using these four features to model `loan_status`.

Data Prep

```r
df2 <- df %>%
  select( married, property_area, credit_history, education, loan_status )

# train test split
set.seed(101)
```

```r
trainIndex <- createDataPartition(df2$loan_status,
                                  p = 0.75,
                                  list = F)

train <- df2[trainIndex,]
test <- df2[-trainIndex,]

# cross validation train control
ctrl <- trainControl(method = 'cv', number = 10)
```

LDA

Model 1: Linear Discriminant Analysis finds a linear combination of features to characterize the separation of target classes. We use the four key features variables that were selected during EDA to build our model (* `credit_history`, `married`, `property_area`, `education`).

```r
lda <- train(loan_status ~ .,
             data = train,
             method = 'lda',
             trControl = ctrl
             )

lda
```

```
## Linear Discriminant Analysis
##
## 452 samples
##   4 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 407, 407, 407, 407, 407, 407, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.79657   0.4467962
```

KNN

Model 2: K-Nearest Neighbors is a non-parametric method used here for classification

```r
knn <- train(loan_status ~ .,
             data = train,
             method = 'knn',
             trControl = ctrl,
             tuneLength = 10
             )

knn
```

```
## k-Nearest Neighbors
##
## 452 samples
##   4 predictor
```

```
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 407, 406, 407, 407, 407, 407, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.7811111  0.4097444
##    7  0.7544444  0.3140822
##    9  0.7522222  0.3022953
##   11  0.7477295  0.2712812
##   13  0.7432850  0.2496886
##   15  0.7499034  0.2637019
##   17  0.7477295  0.2565839
##   19  0.7477295  0.2565839
##   21  0.7477295  0.2526287
##   23  0.7410628  0.2262748
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

To tune the hyperparameter 'k', the model tests multiple values (we specify a tune length of 10). The highest accuracy measure that resulted from cross-validation was $k = 5$; this value is selected for our final KNN model run with the test data.

Decision Tree

Model 3: Decision Tree is a non-parameteric model that uses simple, branched decision rules to optimise classification.

```r
cart <- train(loan_status ~ .,
              data = train,
              method = 'rpart',
              trControl = ctrl
              )

cart
```

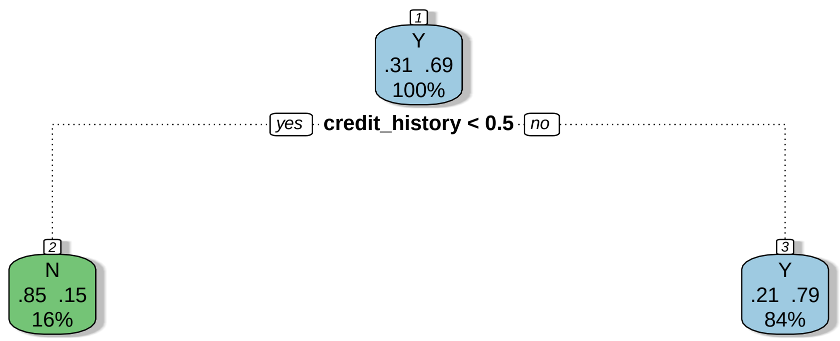
```
## CART
##
## 452 samples
##   4 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 406, 407, 407, 407, 406, 407, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.7743478  0.4109264
##   0.00177305  0.7743478  0.4109264
##   0.34751773  0.7190821  0.1448982
##
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.00177305.
```

```r
#summary( cart )
```

We can also visualize the resulting decision tree:

```r
fancyRpartPlot(cart$finalModel)
```



Rattle 2021−Oct−24 10:08:21 bonzilla

It is interesting that the decision tree does build much depth. Rather, the tree remains shallow using only `credit_history` for classification.

Random Forest

Model 4: Random Forest is an ensemble method that combines multiple decision trees to optimize classification

```r
rf <- train(loan_status ~ .,
            data = train,
            method = 'rf',
            trControl = ctrl
            )

rf
```

```
## Random Forest
##
## 452 samples
##   4 predictor
##   2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 407, 407, 407, 407, 406, 407, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7965700  0.4497066
```

```
##   3     0.7811111  0.4180403
##   5     0.7766667  0.4090197
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Model Performance

Confusion Matrix

Visualize example confusion matrix outcomes on the test data set for the four models:

```
# calculate predictions
test$lda <- predict(lda, test)
test$knn <- predict(knn, test)
test$cart <- predict(cart, test)
test$rf <- predict(rf, test)

table(test$loan_status, test$lda, dnn = c('approval status','LDA predictions'))
```

```
##                LDA predictions
## approval status   N   Y
##               N  23  23
##               Y   2 101
```

```
table(test$loan_status, test$knn, dnn = c('approval status','KNN predictions'))
```

```
##                KNN predictions
## approval status  N  Y
##               N 21 25
##               Y 11 92
```

```
table(test$loan_status, test$cart, dnn = c('approval status','CART predictions'))
```

```
##                CART predictions
## approval status   N   Y
##               N  23  23
##               Y   2 101
```

```
table(test$loan_status, test$rf, dnn = c('approval status','RF predictions'))
```

```
##                RF predictions
## approval status   N   Y
##               N  23  23
##               Y   2 101
```

Comparison of Model Accuracy

The confusion matrices show the results on the test data which can be used to calculate an accuracy score. However, to build a more robust estimate of each model's accuracy, we will collect metrics from multiple fits of the model. This is made very simple with a call to caret's `resamples()` function

```
results <- resamples(list(LDA = lda,  DT = cart, kNN = knn, RF = rf))
summary( results )
```

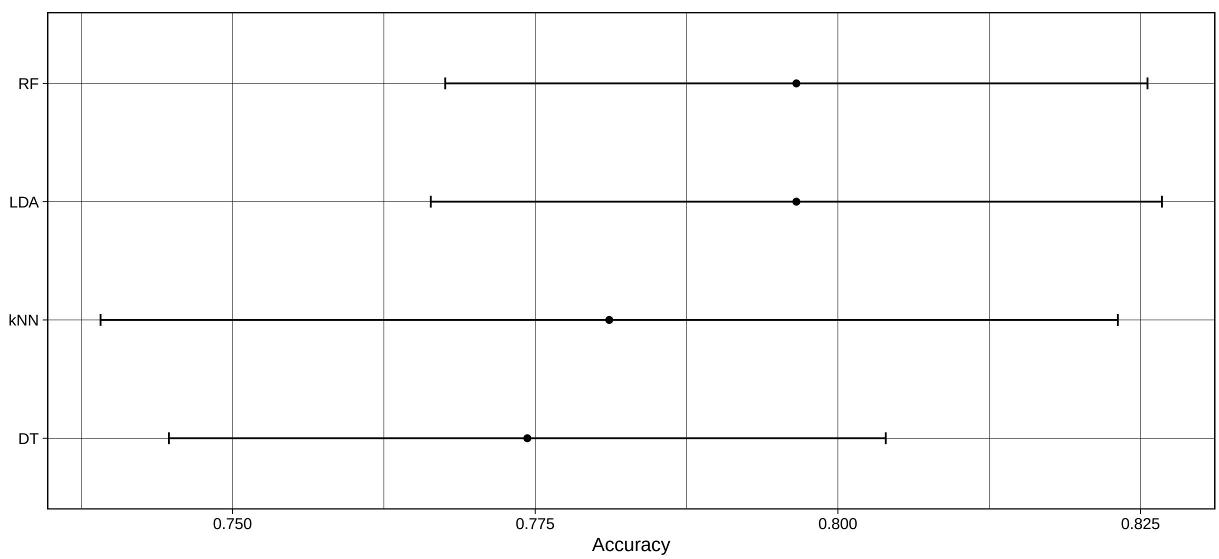
```
##
## Call:
## summary.resamples(object = results)
##
```

```
## Models: LDA, DT, kNN, RF
## Number of resamples: 10
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA 0.7333333 0.7777778 0.7913043 0.7965700 0.8166667 0.8888889    0
## DT  0.6888889 0.7487923 0.7888889 0.7743478 0.8000000 0.8222222    0
## kNN 0.6888889 0.7555556 0.7888889 0.7811111 0.8177536 0.8666667    0
## RF  0.7173913 0.7777778 0.7888889 0.7965700 0.8222222 0.8666667    0
##
## Kappa
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA 0.2263610 0.3823750 0.4359058 0.4467962 0.5018372 0.7246022    0
## DT  0.2446043 0.3607000 0.4359058 0.4109264 0.4911053 0.5081967    0
## kNN 0.1273713 0.3391188 0.4222625 0.4097444 0.5191077 0.6475196    0
## RF  0.2068966 0.3852459 0.4476456 0.4497066 0.5245688 0.6475196    0
```

From the output above, we will focus on the Accuracy measure to chose the best model of the four. We see that **Model 1: LDA** has the highest mean & median accuracy score.

We can visualize this outcome with ggplot:

```r
ggplot(results) +
  labs(y = "Accuracy") +
  theme_linedraw()
```



This graph visualizes that, although Model 1: LDA had the highest accuracy, Model 4: Random Field had very similar performance.

References

1. Dealing with Missingness: Single Imputation

2. ANOVA test with R
3. Chi-squared test of independence in R