Homework 4 (Group 5)

Auto Insurance Predictions

Maria A Ginorio

r Sys.Date()

Contents

Overview . . . 2

Dataset . . . 3

1. Data Exploration . . . 4

    Objective . . . 4

    Data Overview . . . 4

    Cleaning . . . 6

    Distributions . . . 8

    Outliers . . . 10

    Relationships . . . 11

    Skeweness . . . 12

    Correlation . . . 20

2. Data Preparation . . . 21

    Missing Data . . . 21

    Correlation . . . 23

    Preprocess . . . 24

3. Building Models . . . 31

    Logistic Regression . . . 31

    Random Forest . . . 31

    Boosted Tree . . . 31

    LM LINEAR Regression . . . 31

    GLM LINEAR Regression . . . 31

3.2 Workflow Creation . . . 32

    Logistic Regression . . . 32

    Random Forest . . . 32

    Boosted Tree . . . 33

    LM LINEAR Regression . . . 33

    GLM LINEAR Regression . . . 33

. 34
4. Evaluate Models 35
Logistic Regression 35
Random Forests 38
Xboost 38
LINEAR REGRESSION 39
LM LINEAR REGRESSION 39
GLM LINEAR REGRESSION 39
Final Selection 40
References 41
Apendix 41

Overview

In this homework assignment, you will explore, analyze and model a data set containing approximately 8000 records representing a customer at an auto insurance company. Each record has two response variables.

The first response variable

TARGET_FLAG, is a 1 or a 0.

- A “1” means that the person was in a car crash.
- A “0” means that the person was not in a car crash.

The second response variable is TARGET_AMT.

- This value is “0” if the person did not crash their car.
- But if they did crash their car (1), this number will be a value greater than zero (*) >0

Dataset

VARIABLE NAME	DEFINITION	THEORETICAL EFFECT
INDEX	Identification Variable (do not use)	None
TARGET_FLAG	Was Car in a crash? 1=YES 0=NO	None
TARGET_AMT	If car was in a crash, what was the cost	None
AGE	Age of Driver	Very young people tend to be risky. Maybe very old people also.
BLUEBOOK	Value of Vehicle	Unknown effect on probability of collision, but probably effect the payout if there is a crash
CAR_AGE	Vehicle Age	Unknown effect on probability of collision, but probably effect the payout if there is a crash
CAR_TYPE	Type of Car	Unknown effect on probability of collision, but probably effect the payout if there is a crash
CAR_USE	Vehicle Use	Commercial vehicles are driven more, so might increase probability of collision
CLM_FREQ	# Claims (Past 5 Years)	The more claims you filed in the past, the more you are likely to file in the future
EDUCATION	Max Education Level	Unknown effect, but in theory more educated people tend to drive more safely
HOMEKIDS	# Children at Home	Unknown effect
HOME_VAL	Home Value	In theory, home owners tend to drive more responsibly
INCOME	Income	In theory, rich people tend to get into fewer crashes
JOB	Job Category	In theory, white collar jobs tend to be safer
KIDSDRIV	# Driving Children	When teenagers drive your car, you are more likely to get into crashes
MSTATUS	Marital Status	In theory, married people drive more safely
MVR_PTS	Motor Vehicle Record Points	If you get lots of traffic tickets, you tend to get into more crashes
OLDCLAIM	Total Claims (Past 5 Years)	If your total payout over the past five years was high, this suggests future payouts will be high
PARENT1	Single Parent	Unknown effect
RED_CAR	A Red Car	Urban legend says that red cars (especially red sports cars) are more risky. Is that true?
REVOKED	License Revoked (Past 7 Years)	If your license was revoked in the past 7 years, you probably are a more risky driver.
SEX	Gender	Urban legend says that women have less crashes then men. Is that true?
TIF	Time in Force	People who have been customers for a long time are usually more safe.
TRAVTIME	Distance to Work	Long drives to work usually suggest greater risk
URBANICITY	Home/Work Area	Unknown
YOJ	Years on Job	People who stay at a job for a long time are usually more safe

1. Data Exploration

Objective

- build a multiple linear regression and a binary logistic regression models on the training data
- predict the probability that a person will crash their car and
- predict the amount of money it will cost if the person does crash their car

Data Overview

Lets first look at the raw data values by using the skim package

Table 1: Data summary

Name	df
Number of rows	8161
Number of columns	26
Column type frequency:	
character	14
numeric	12
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
INCOME	445	0.95	2	8	0	6612	0
PARENT1	0	1.00	2	3	0	2	0
HOME_VAL	464	0.94	2	8	0	5106	0
MSTATUS	0	1.00	3	4	0	2	0
SEX	0	1.00	1	3	0	2	0
EDUCATION	0	1.00	3	13	0	5	0
JOB	526	0.94	6	13	0	8	0
CAR_USE	0	1.00	7	10	0	2	0
BLUEBOOK	0	1.00	6	7	0	2789	0
CAR_TYPE	0	1.00	3	11	0	6	0
RED_CAR	0	1.00	2	3	0	2	0
OLDCLAIM	0	1.00	2	7	0	2857	0
REVOKED	0	1.00	2	3	0	2	0
URBANICITY	0	1.00	19	21	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
INDEX	0	1.00	5151.87	2978.89	1	2559	5133	7745	10302.0
TARGET_FLAG	0	1.00	0.26	0.44	0	0	0	1	1.0
TARGET_AMT	0	1.00	1504.32	4704.03	0	0	0	1036	107586.1
KIDSDRIV	0	1.00	0.17	0.51	0	0	0	0	4.0
AGE	6	1.00	44.79	8.63	16	39	45	51	81.0
HOMEKIDS	0	1.00	0.72	1.12	0	0	0	1	5.0

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
YOJ	454	0.94	10.50	4.09	0	9	11	13	23.0
TRAVTIME	0	1.00	33.49	15.91	5	22	33	44	142.0
TIF	0	1.00	5.35	4.15	1	1	4	7	25.0
CLM_FREQ	0	1.00	0.80	1.16	0	0	0	2	5.0
MVR_PTS	0	1.00	1.70	2.15	0	0	1	3	13.0
CAR_AGE	510	0.94	8.33	5.70	-3	1	8	12	28.0

From the description seen by the skim package we can observe we have two variables that should be transformed into factors since they have (1) or (0) values. `chas & target.`

Cleaning

First we have decided to change the name of certain variables to aid the understanding of their meaning.

Symbol $ type char

We have 4 variables that have the dollar sign in front of them and they are considered character when they should be numeric. We will use str_remove for this process.

Variables	Removed	Changed
BLUEBOOK	"$" , ","	dbl
HOME_VAL	"$" , ","	dbl
INCOME	"$" , ","	dbl
CLAIM_TOT	"$" , ","	dbl

Symbol z_, <

We will proceed to remove symbols found ie. "z_No"

Variables	Removed
Married	z_
Sex	z_
Job	z_
Car_type	z_
Urbancity	z_, /, Highly Urban, Highly Rural
Education	z_ & <

Discrete vs. Continuous

Variables	TYPE
TARGET_FLAG	factor
SINGLE_PARENT	factor
MARRIED	factor
SEX	factor
EDUCATION	factor
JOB	factor
CAR_USE	factor
CAR_TYPE	factor
RED_CAR	factor
REVOKED	factor
URBANCITY	factor

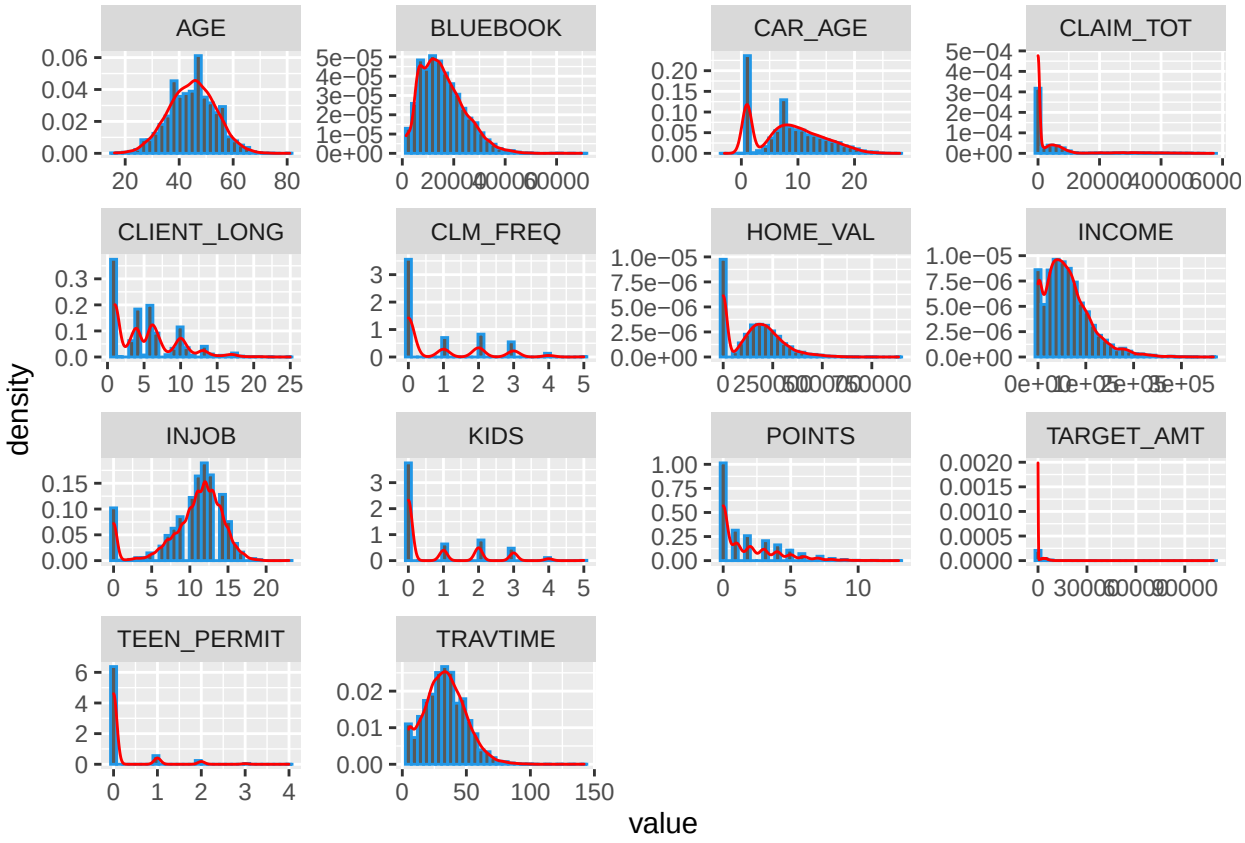
```
## tibble [8,161 x 25] (S3: tbl_df/tbl/data.frame)
##  $ TARGET_FLAG : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 2 2 1 ...
##  $ TARGET_AMT  : num [1:8161] 0 0 0 0 0 ...
##  $ TEEN_PERMIT : num [1:8161] 0 0 0 0 0 0 0 1 0 0 ...
##  $ AGE         : num [1:8161] 60 43 35 51 50 34 54 37 34 50 ...
##  $ KIDS        : num [1:8161] 0 0 1 0 0 1 0 2 0 0 ...
##  $ INJOB       : num [1:8161] 11 11 10 14 NA 12 NA NA 10 7 ...
##  $ INCOME      : num [1:8161] 67349 91449 16039 NA 114986 ...
##  $ SINGL_PARENT: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
##  $ HOME_VAL    : num [1:8161] 0 257252 124191 306251 243925 ...
##  $ MARRIED     : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 1 2 2 1 1 ...
##  $ SEX         : Factor w/ 2 levels "M","F": 1 1 2 1 2 2 2 1 2 1 ...
```

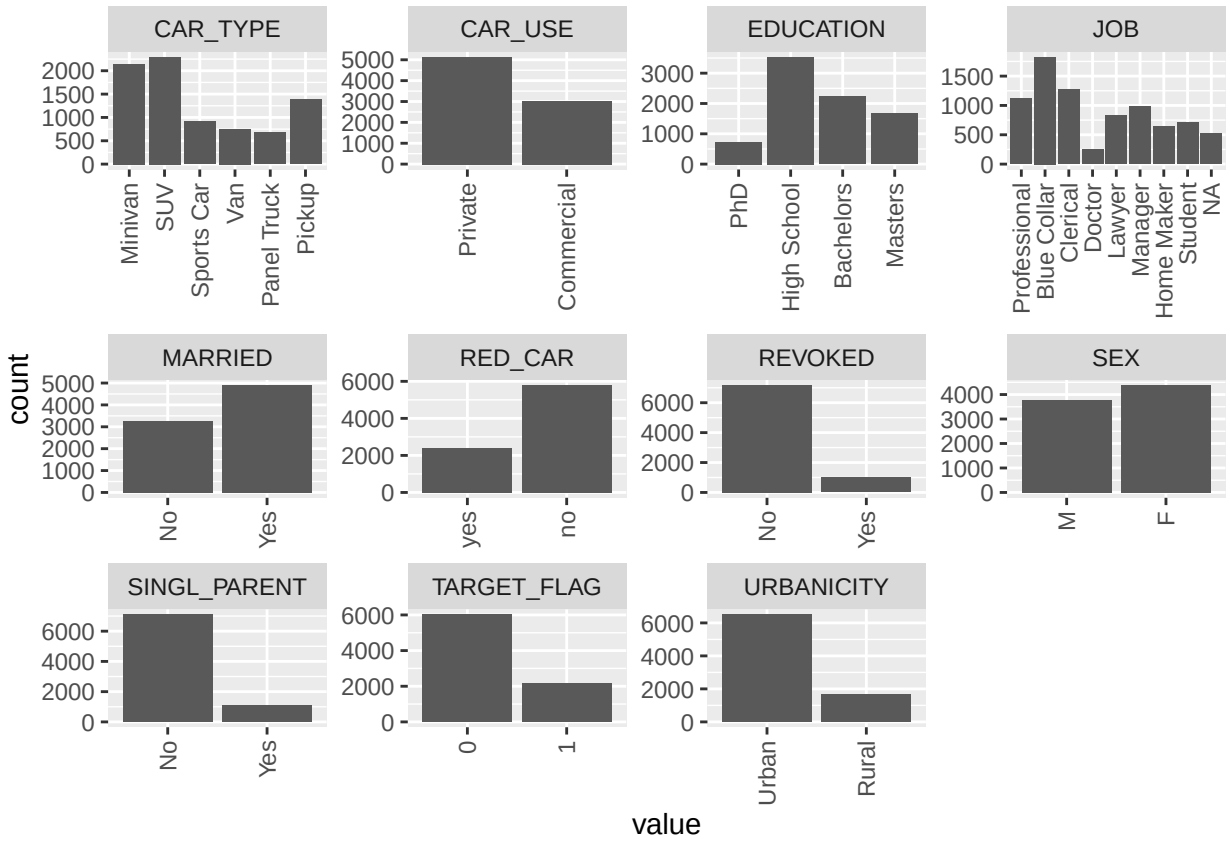
```
##  $ EDUCATION   : Factor w/ 4 levels "PhD","High School",..: 1 2 2 2 1 3 2 3 3 3 ...
##  $ JOB         : Factor w/ 8 levels "Professional",..: 1 2 3 2 4 2 2 2 3 1 ...
##  $ TRAVTIME    : num [1:8161] 14 22 5 32 36 46 33 44 34 48 ...
##  $ CAR_USE     : Factor w/ 2 levels "Private","Commercial": 1 2 1 1 1 2 1 2 1 2 ...
##  $ BLUEBOOK    : num [1:8161] 14230 14940 4010 15440 18000 ...
##  $ CLIENT_LONG : num [1:8161] 11 1 4 7 1 1 1 1 1 7 ...
##  $ CAR_TYPE    : Factor w/ 6 levels "Minivan","SUV",..: 1 1 2 1 2 3 2 4 2 4 ...
##  $ RED_CAR     : Factor w/ 2 levels "yes","no": 1 1 2 1 2 2 2 1 2 2 ...
##  $ CLAIM_TOT   : num [1:8161] 4461 0 38690 0 19217 ...
##  $ CLM_FREQ    : num [1:8161] 2 0 2 0 2 0 0 1 0 0 ...
##  $ REVOKED     : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 2 1 1 ...
##  $ POINTS      : num [1:8161] 3 0 3 0 3 0 0 10 0 1 ...
##  $ CAR_AGE     : num [1:8161] 18 1 10 6 17 7 1 7 1 17 ...
##  $ URBANICITY  : Factor w/ 2 levels "Urban","Rural": 1 1 1 1 1 1 1 1 1 2 ...
```

Distributions

We will first explore the data looking for issues or challenges (i.e. missing data, outliers, possible coding errors, multicollinearlity, etc). Once we have a handle on the data, we will apply any necessary cleaning steps. Once we have a reasonable dataset to work with, we will build and evaluate three different Logistic models that predict probabilities and cost.

In this case we will observe the distribution and frequency of numeric variables first.





The distribution of our variables can also alert us of of unusual patterns, in this case we have observed the prevalence of kurtosis for certain variables like:, `BLUEBOOK, CLAIM_TOT`, `INCOME, TRAVTIME` are skewed to the right.

After creating independent histograms for each variable we have found 3 variables that appear to be bi-modal. We notice that the graphs of this variables have two distinct humps or peaks with a valley separating them. We could attribute this observations to possibly different groups. We find that CAR_AGE and CLIENT_LONG, HOME_VAL are bi-modal.

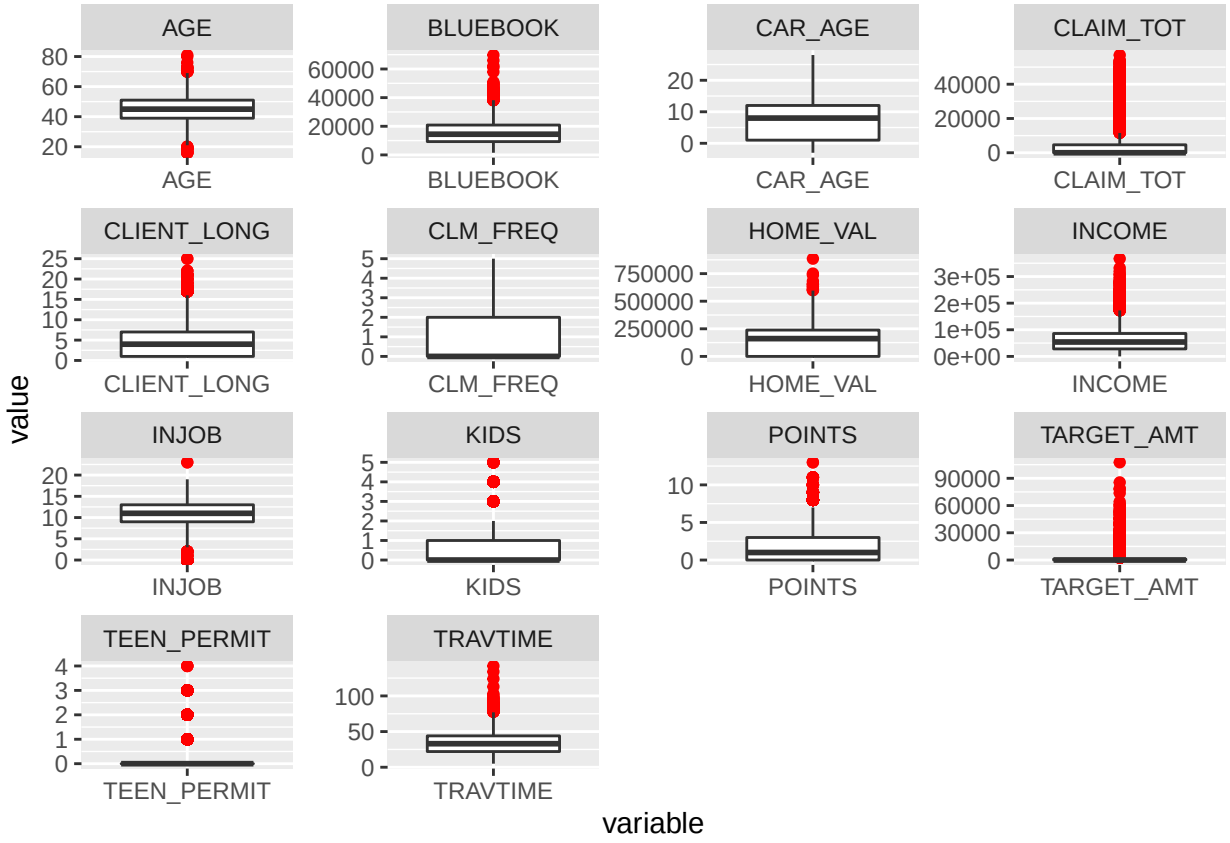
Outliers

In addition to histogram graph of our variable we thought it was pertinent to take a look at our variables using a boxplot. It will help us quickly visualize the distribution of the values in the dataset and see where the five number summary values are located.

In addition, we will be able to create a clear picture of the median values and the spreads across all the distributions. One of the most important observation we will obtain from this graph however, is outlier detection.

Find outliers in red below:

```
## Warning: Removed 1879 rows containing non-finite values (stat_boxplot).
```



Indication of outliers is present in almost all variables except CLM_FREQ

A key is whether an outlier represents a contaminated observation or a rare case.

Are these data points unusual or different in some way from the rest of the data? We will have to consider removing this and refit the data if we consider they could be affecting our results.

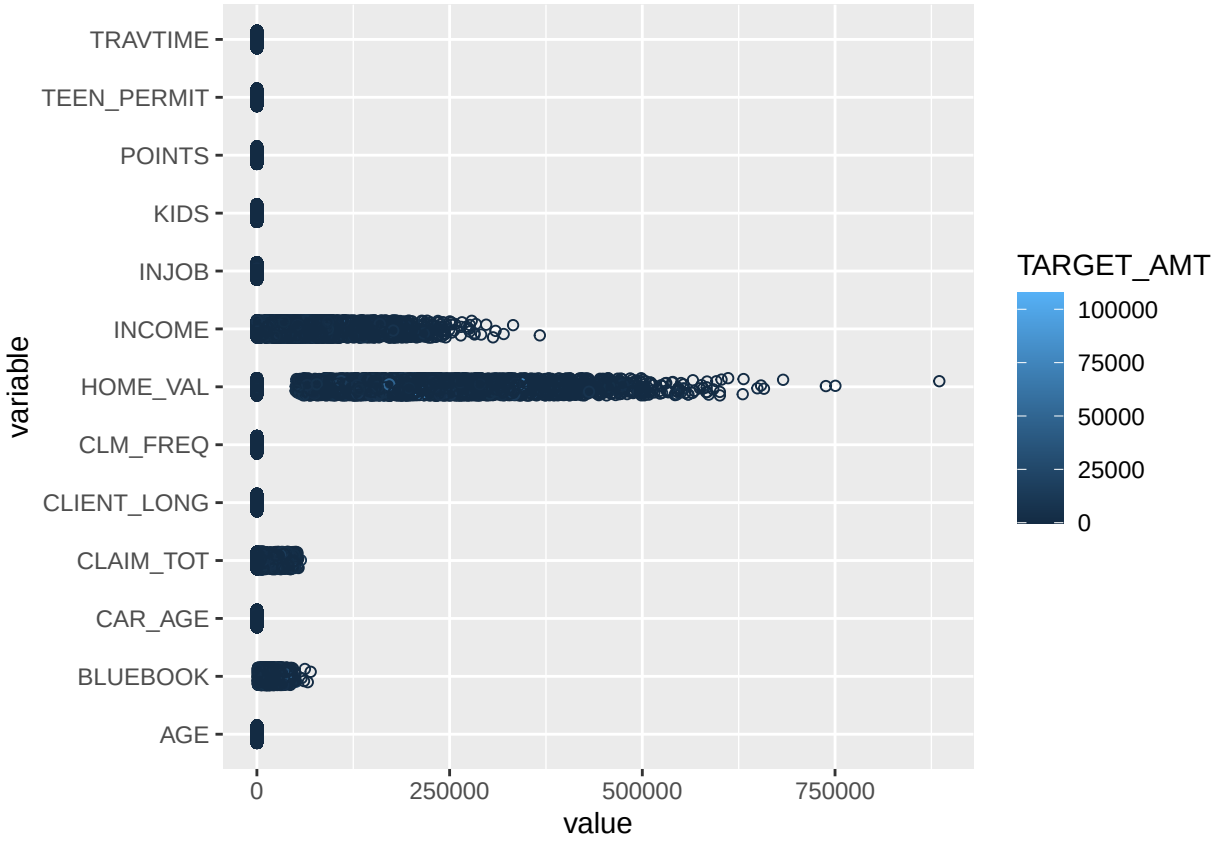
One of the first steps in any type of analysis is to take a closer look at the observations that have high leverage since they could have a large impact on the results of a given model.

Relationships

We want use scatter plots in each variable versus the target variable to get an idea of the relationship between them.

The plots indicate interesting relationship between the `target` variable however some of them start showing signs of relationship and groups.

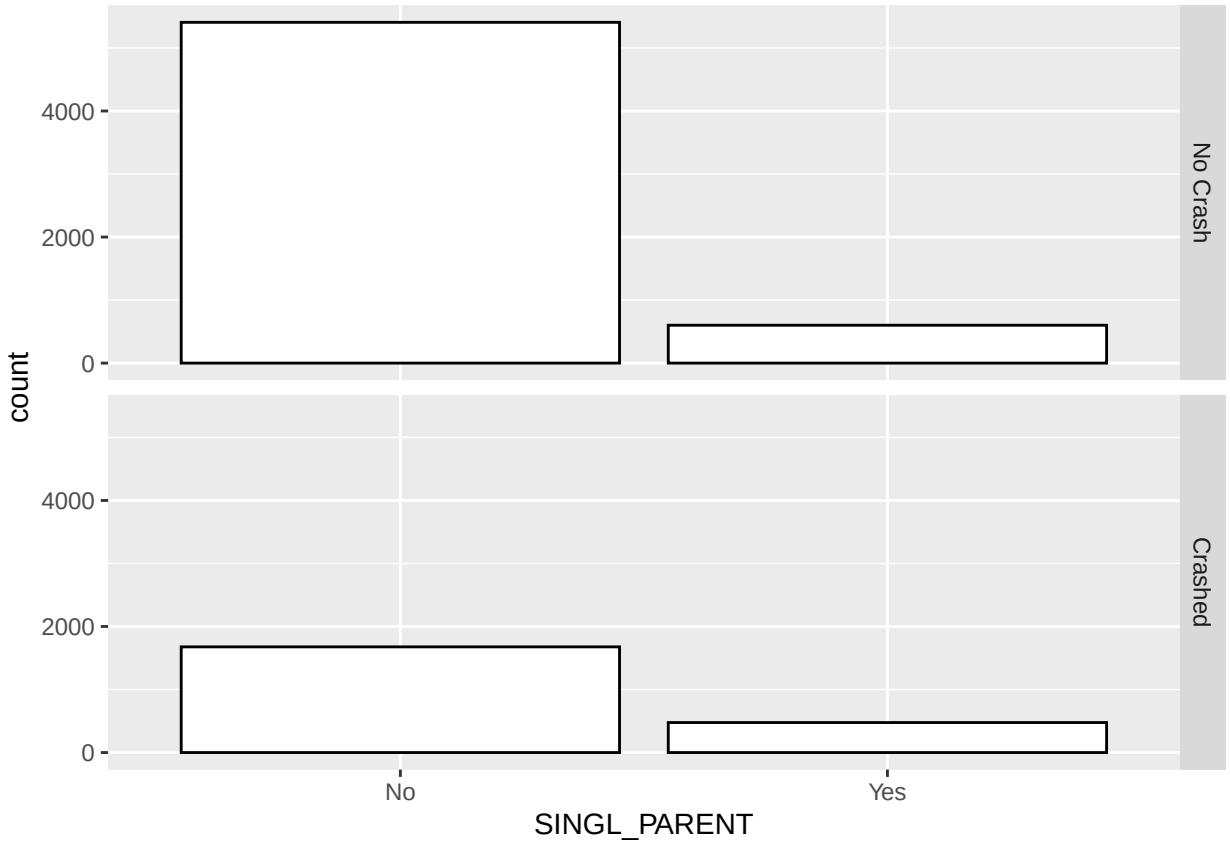
Some of the predictors variables are skewed and not normally distributed, in addition we have outliers and bimodality.

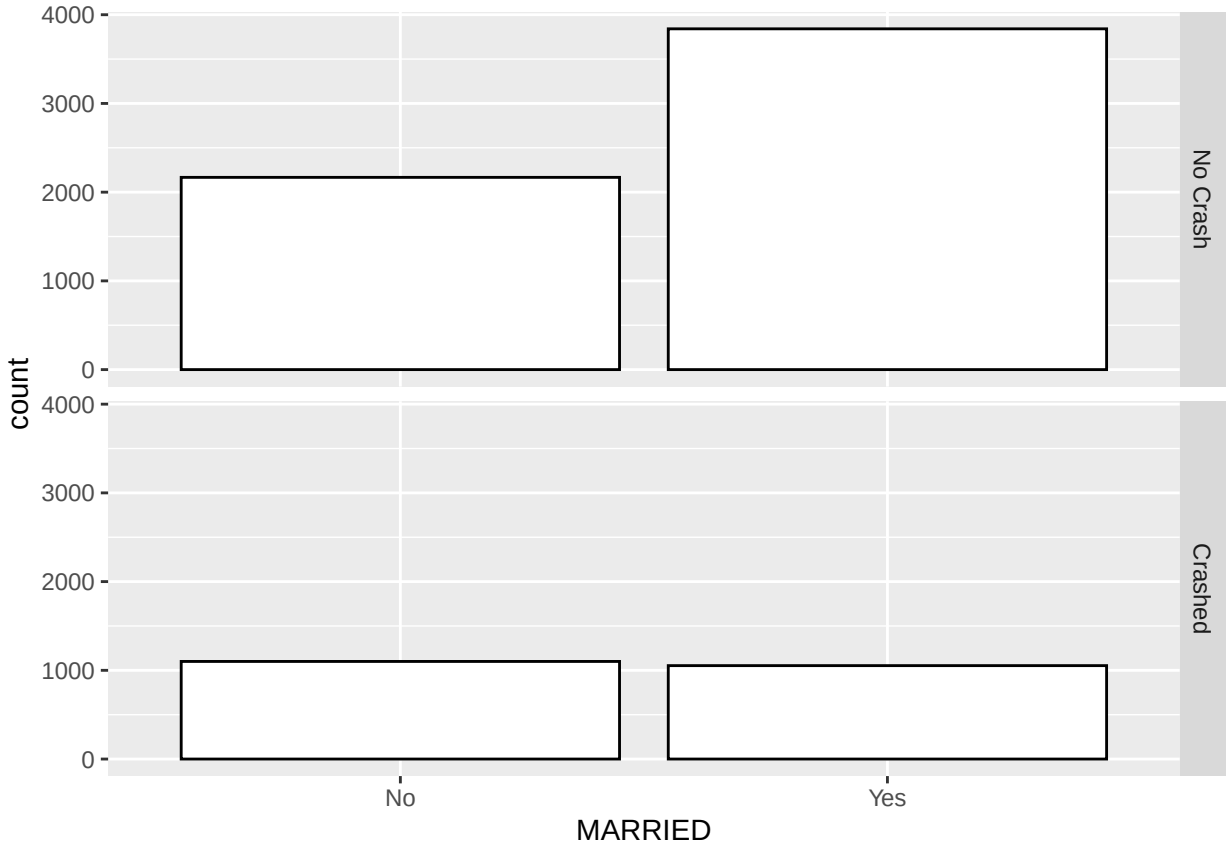


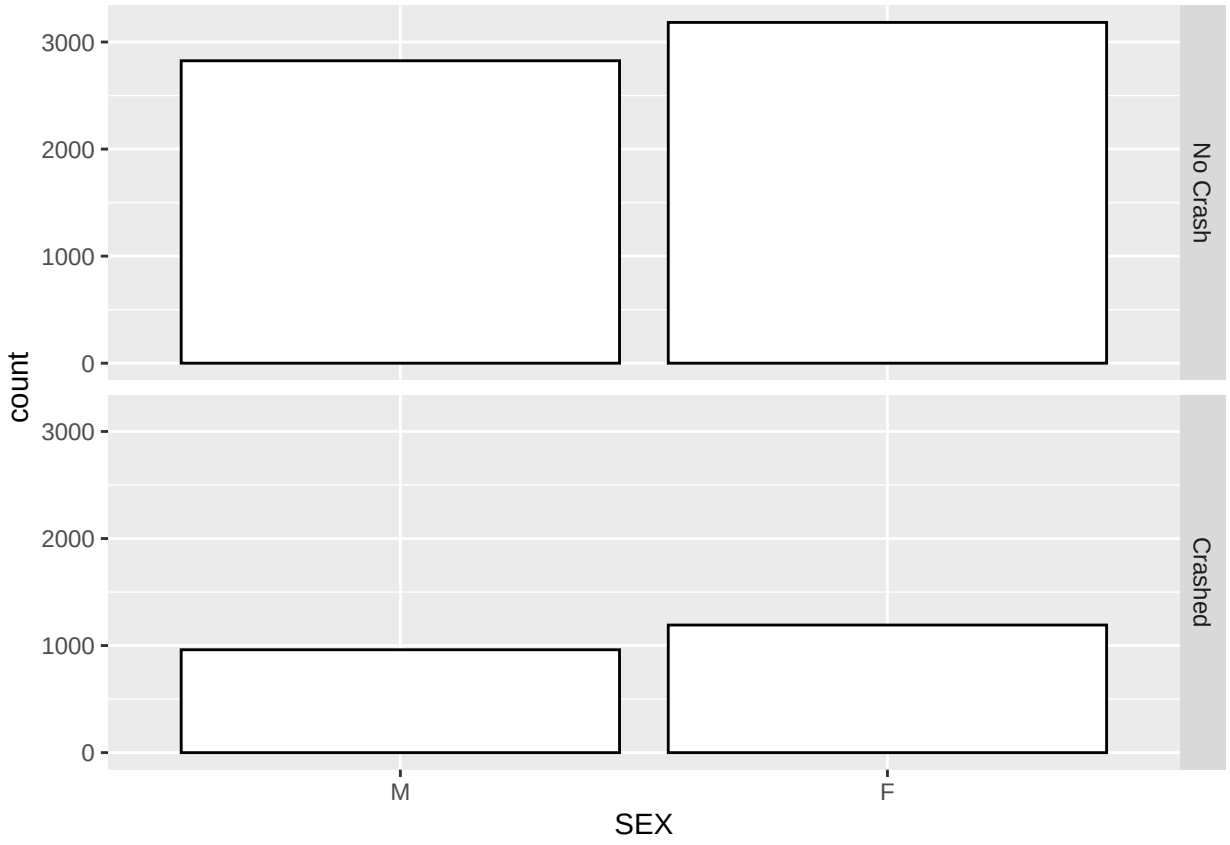
We take some of the variables to be analyzed separately against the target

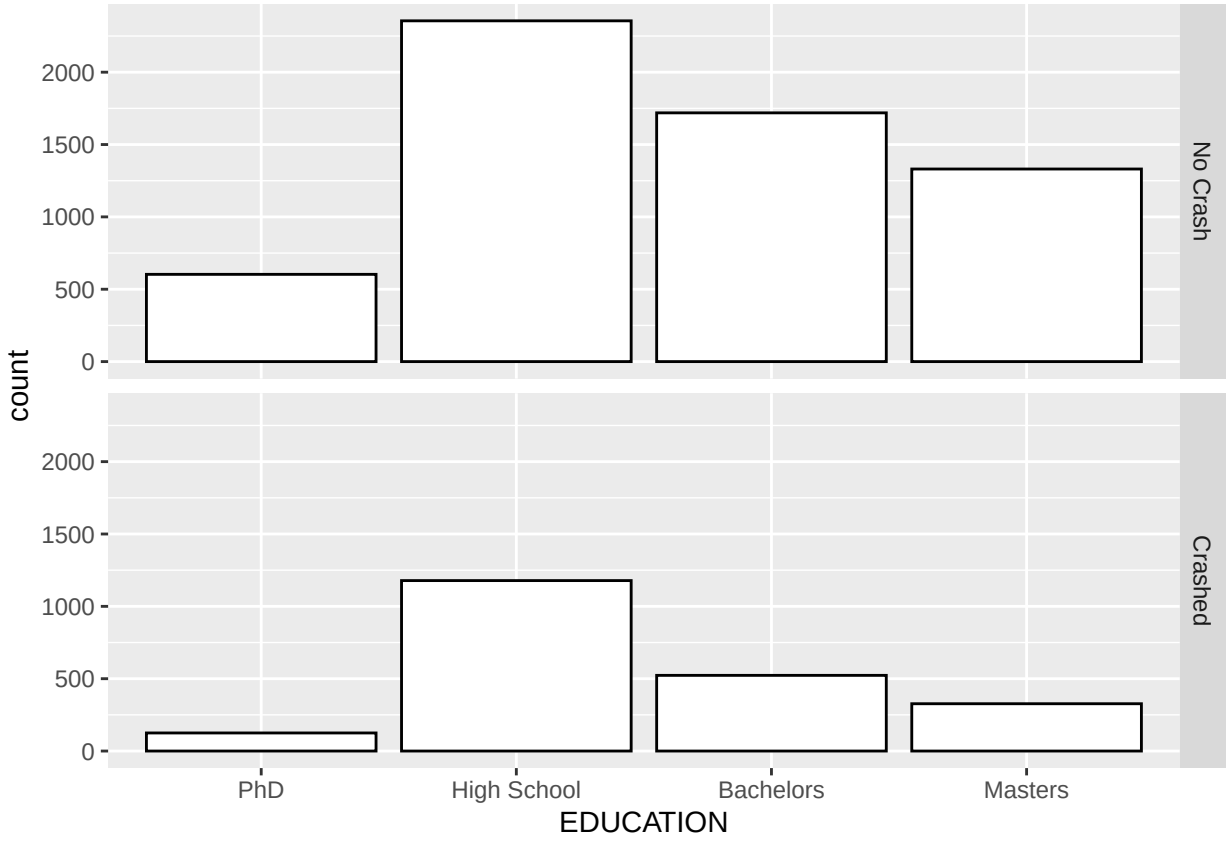
Skeweness

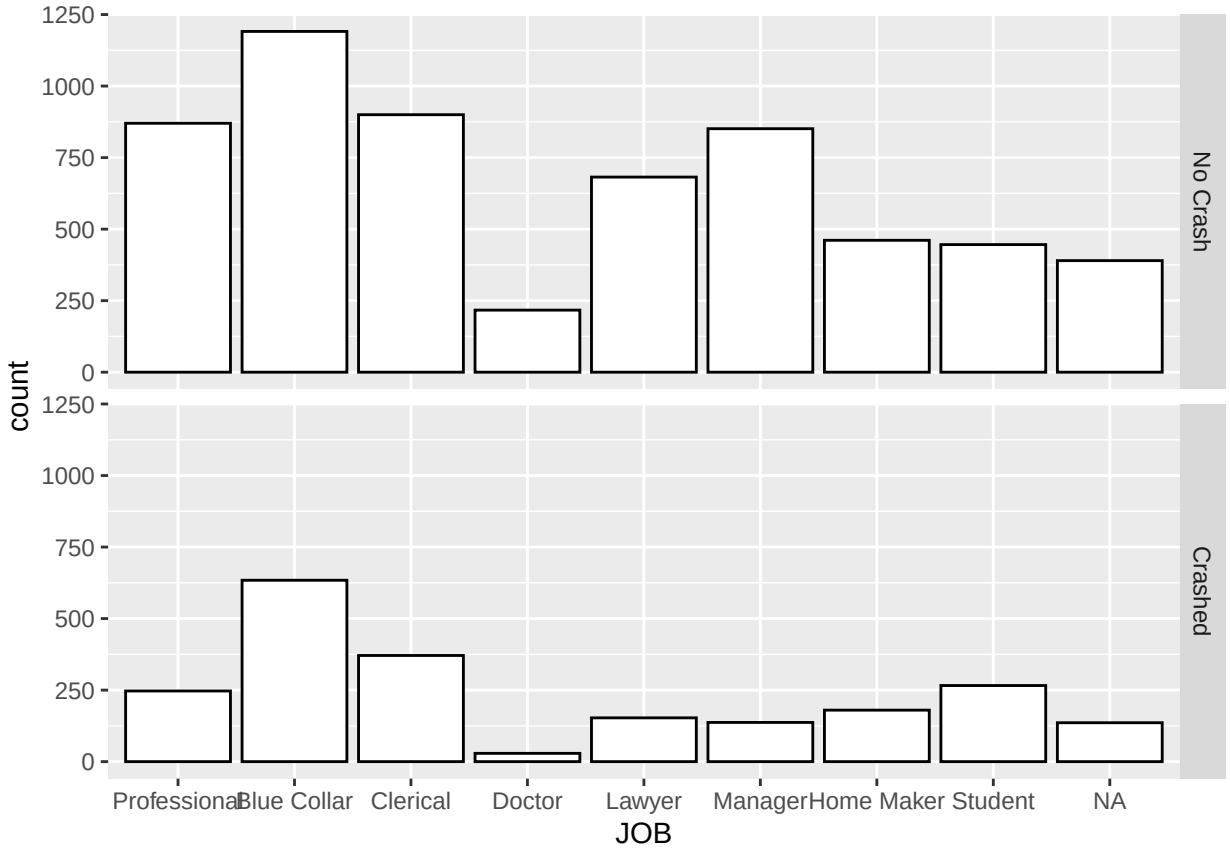
Histogram of predictors by factors of target

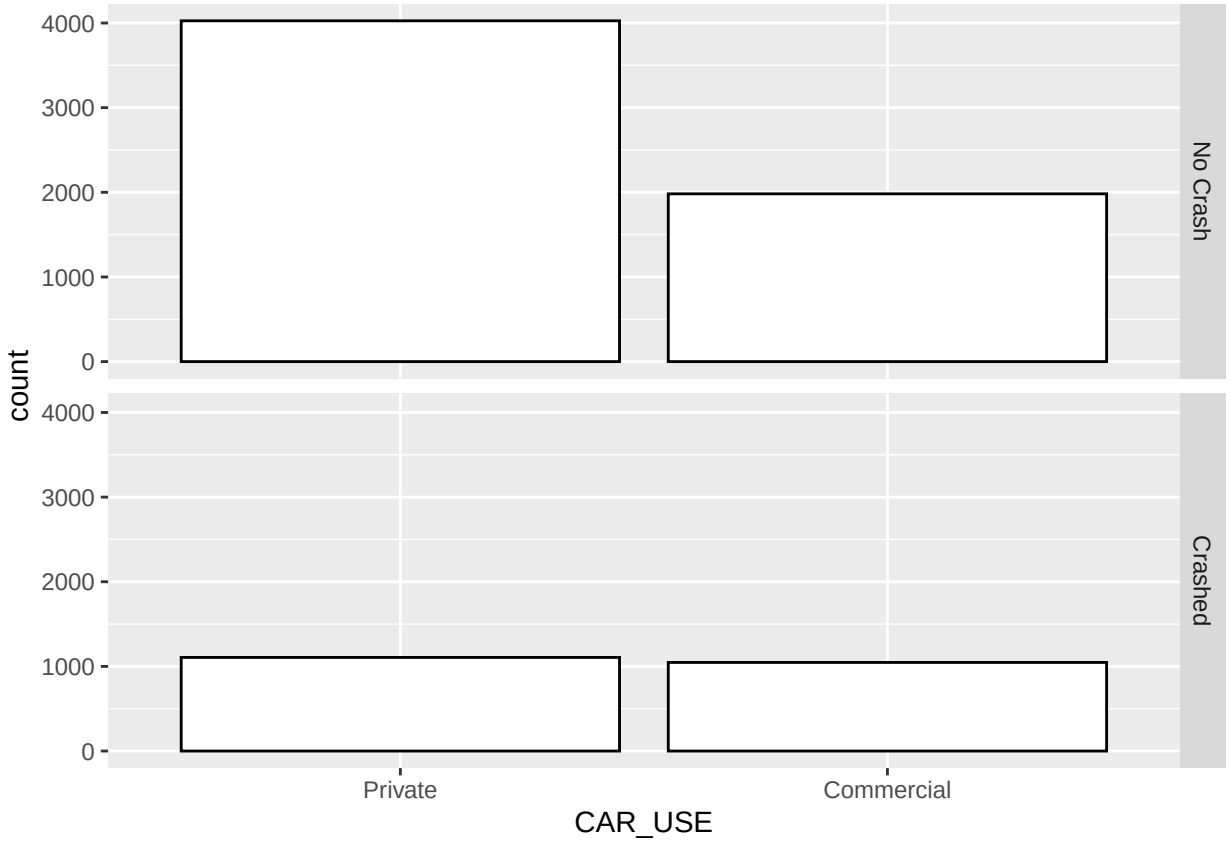


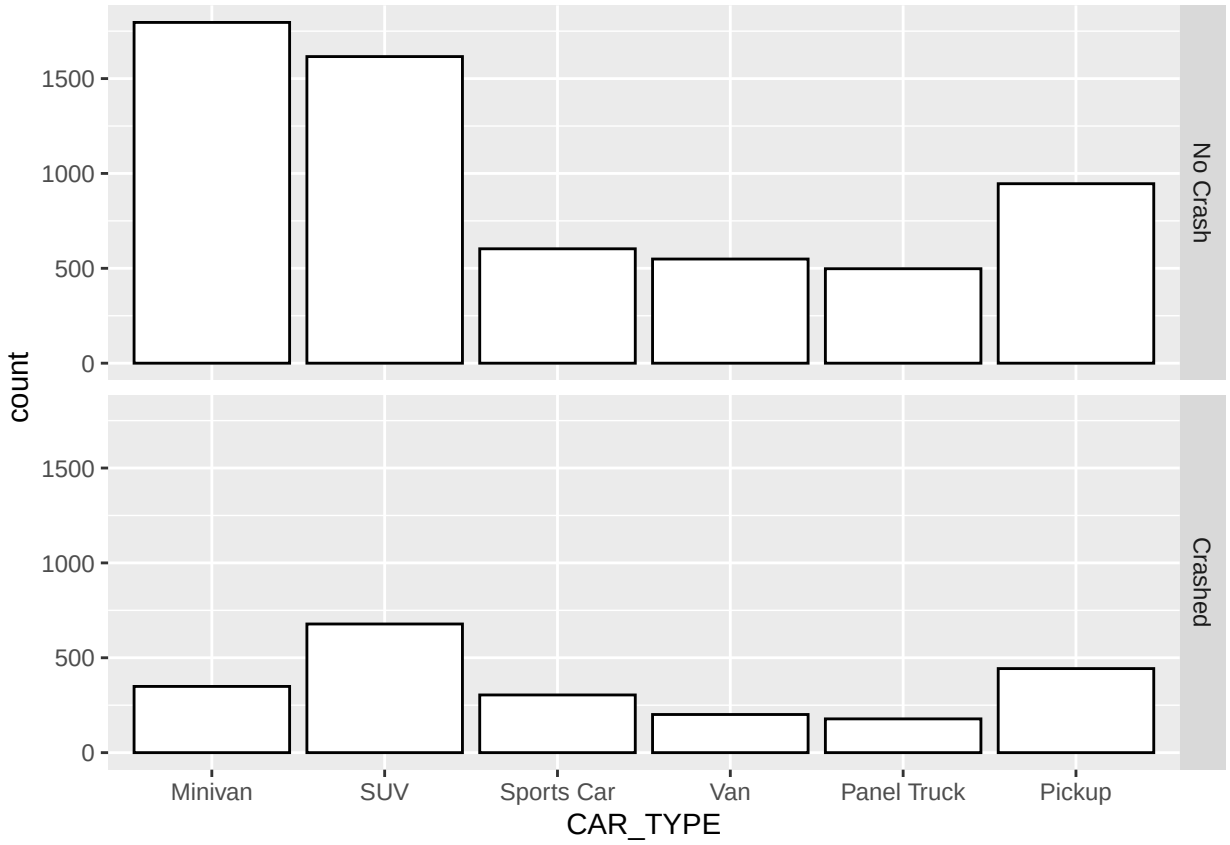

4000
3000
2000
1000
0
count
No Crash
Crashed
No
Yes
MARRIED

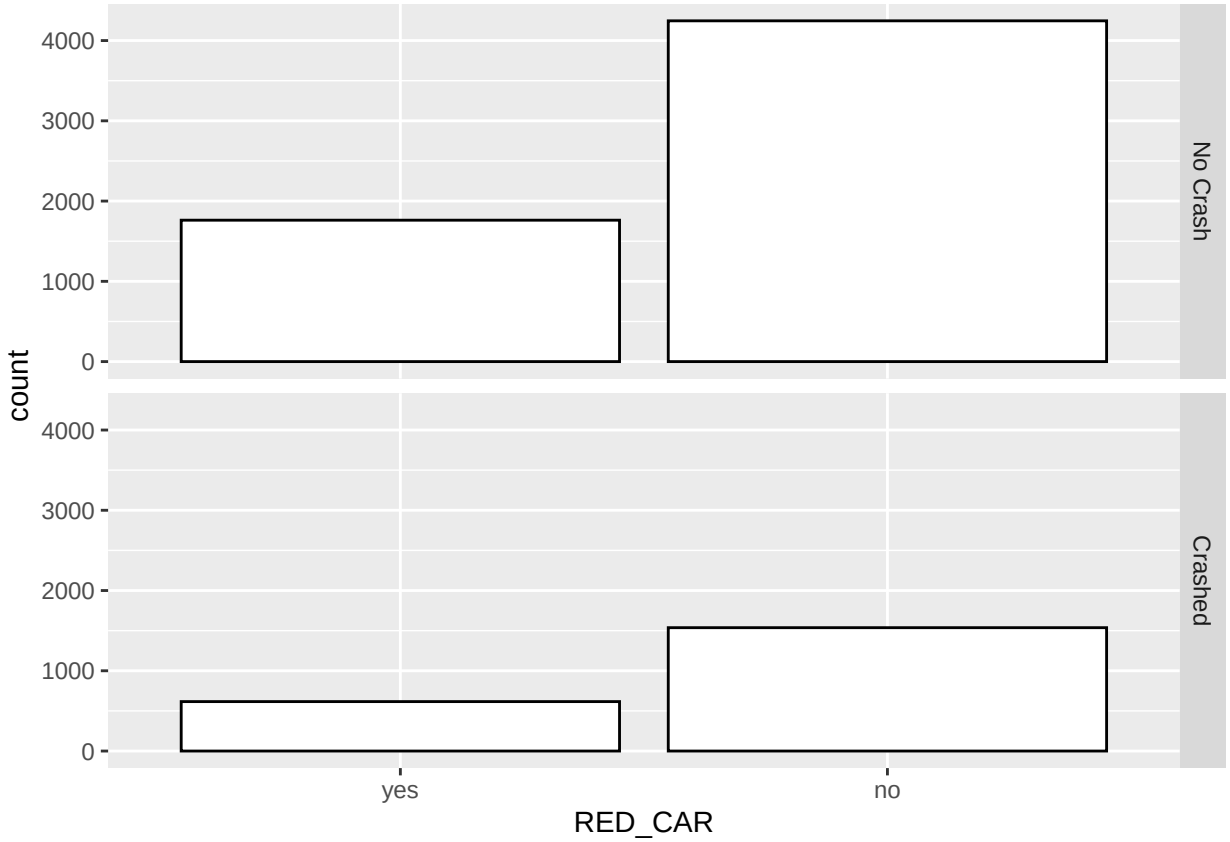

3000
2000
1000
0
count
No Crash
Crashed
M
F
SEX

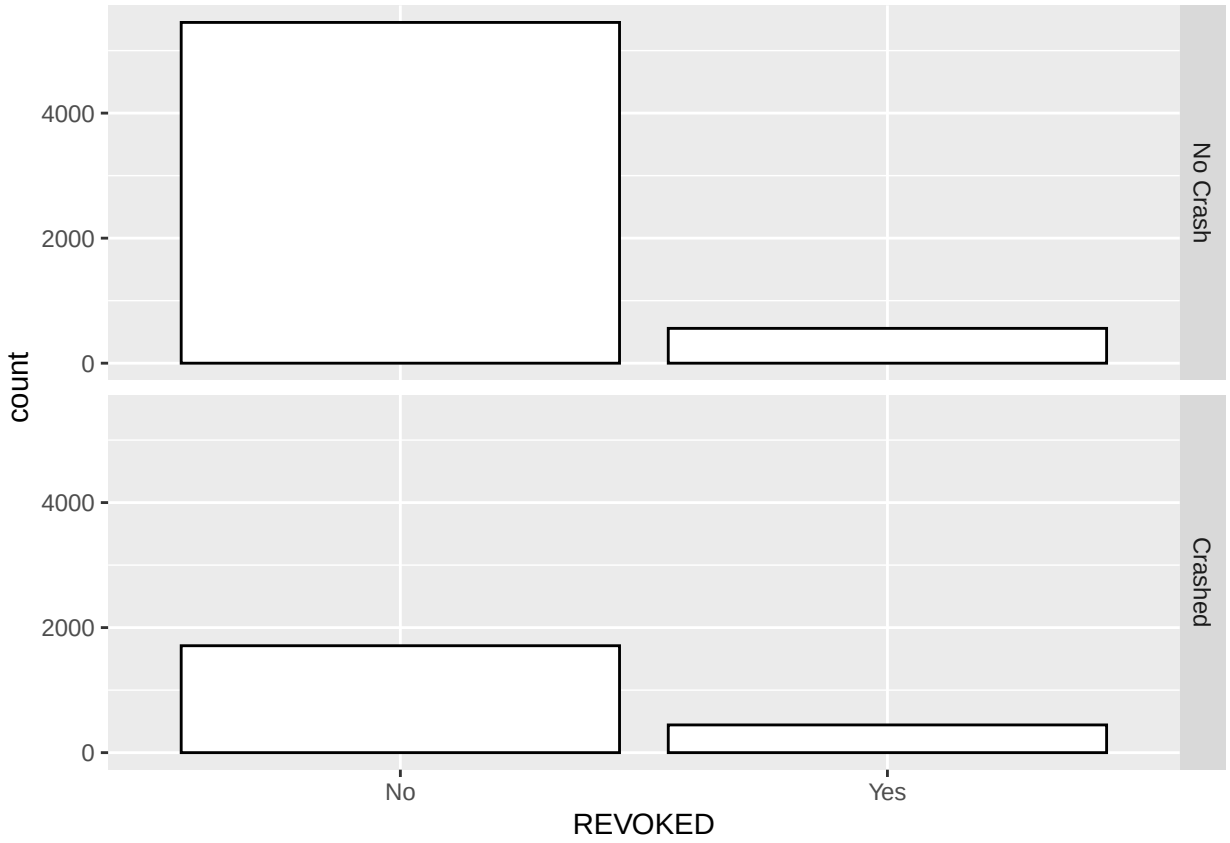

count
2000
1500
1000
500
0
No Crash
Crashed
PhD
High School
Bachelors
Masters
EDUCATION

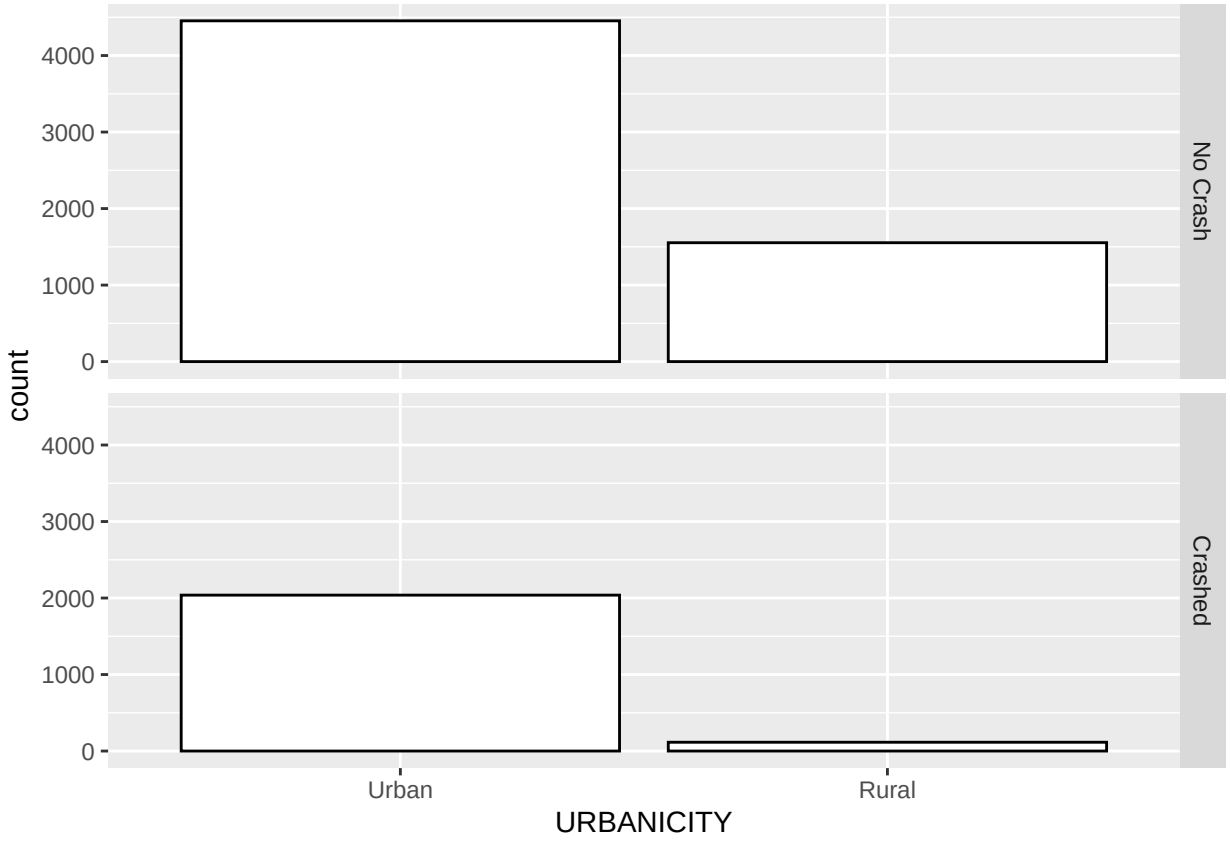

count
1250
1000
750
500
250
0
No Crash
Crashed
Professional
Blue Collar
Clerical
Doctor
Lawyer
Manager
Home Maker
Student
NA
JOB


4000
3000
2000
1000
0
count
No Crash
Crashed
Private
Commercial
CAR_USE


1500
1000
500
0
count
No Crash
Crashed
Minivan
SUV
Sports Car
Van
Panel Truck
Pickup
CAR_TYPE


4000
3000
2000
1000
0
count
No Crash
Crashed
yes
no
RED_CAR


4000
2000
0
count
No Crash
Crashed
No
Yes
REVOKED

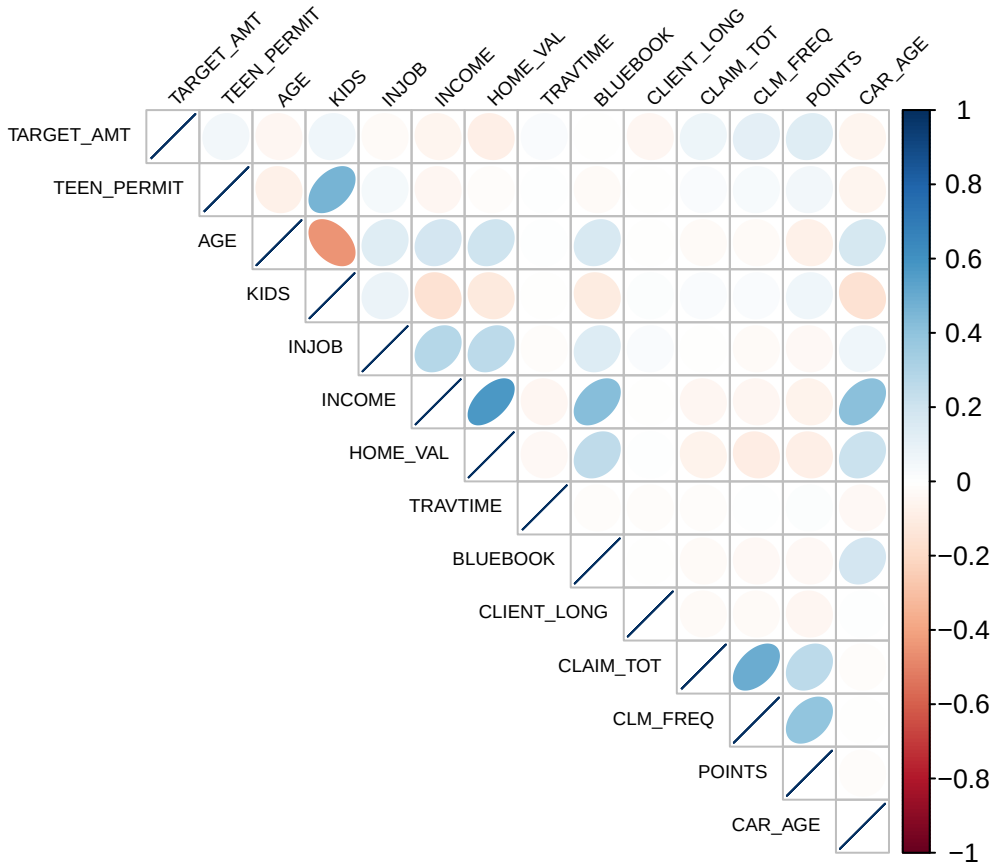


We can observe very interesting histograms from the predictors by factors of Target (crashed or not)

Variable	Target (crashed or not)
SINGLE_PARENT	There are less single parents that have crashed than single parents who have not crashed.
MARRIED	There is more Married people who DO NOT crashed rather than Married people who crash
SEX	Females are slightly higher on crashed than males. However, the competition is almost equal for no crashes.
EDUCATION	The Highest rate for crashing is in High school followed by Bachelors degree.
JOB	Blue Collar, clerical and students tend to have the higher rates for crashes.
CAR_USE	Private and Commercial are almost identical. fewer additional cases in Private
CAR_TYPE	The highest rate for crashing is in SUV, Pickup and Minivan. Surprisingly, sport cars are not in the top.
RED_CAR	It seems like the color of the car has little influence in crashes as there rate of crashes of red cars is low.
REVOKED	People with NO revoked licenses have more crashes than the ones with revoked licences
URBANICITY	There is definitively more crashes in Urban areas rather than rural areas.

Multicolinearity

TARGET_AMOUNT



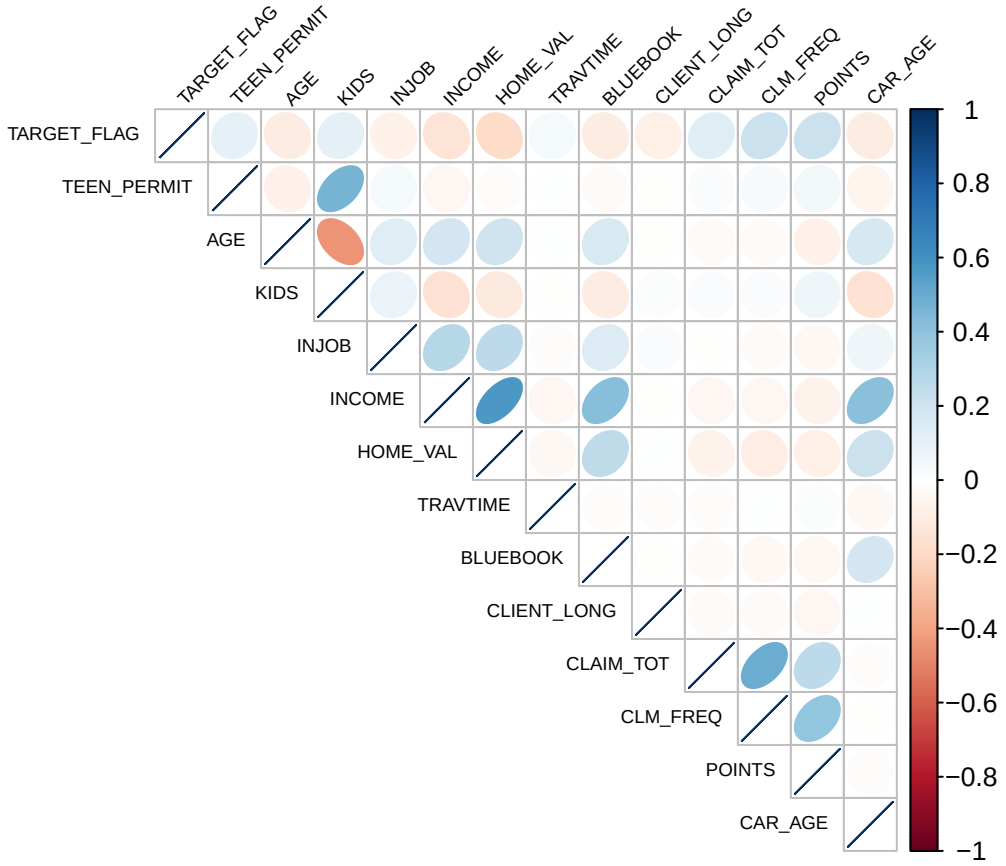
We can see that some variables are correlated with one another, such as `kids` & `teen_permit`.

When we start considering features for our models, we'll need to account for the correlations between features and avoid including pairs with strong correlations.

Many features are also inherently associated, for example, as the `income` increases to `home_value` increase, In addition, we can see that when `kids` in the household increase age in the policy holder decreases

Both target variables have no direct correlation with any of the predictors.

TARGET_FLAG



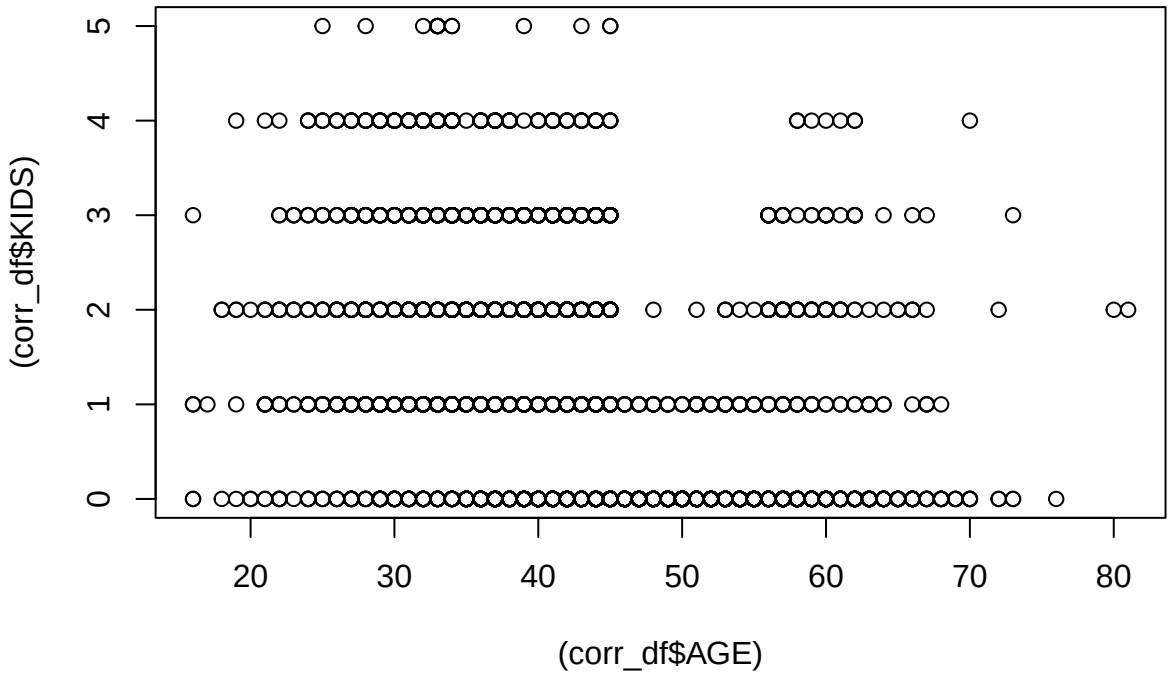
In this correlation plot we can see more relationships with the target_flag variable.

We can see that as `income` increases the value of `Bluebook` also increases, this is also correlated to the `car_age`. Which makes sense because most people with higher incomes usually drive newer pricier cars.

In addition, we find `Claim_tot` and `Claim_freq` to have a relationship with `Points`.

Correlation

Earlier we discovered the correlation between `AGE` and `KIDS`. We want to understand this relationship better by plotting them.




```
##
##  Pearson's product-moment correlation
##
## data:  corr_df$AGE and corr_df$KIDS
## t = -44.924, df = 8153, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4626723 -0.4278733
## sample estimates:
##       cor
## -0.445441
```

Regarding the strength of the relationship: The **more extreme** the correlation coefficient (the closer to -1 or 1), the **stronger the relationship**. This also means that a **correlation close to 0** indicates that the two variables are **independent**, that is, as one variable increases, there is no tendency in the other variable to either decrease or increase.

The p-value of the correlation test between these 2 variables is 2.2e-16. At the -4.6% significance level, we do not reject the null hypothesis of no correlation. We therefore conclude that we do not reject the hypothesis that there is no linear relationship between the 2 variables.

This test proves that even if the correlation coefficient is different from 0 (the correlation is -0.4 in the sample), it is actually not significantly different from 0 in the population.

2. Data Preparation

Missing Data

To prepare our data we have already determined that we have missing data in our dataset.

Numerical Variables

We will see our numerical values first:

variable	total	isna	num.isna	pct
AGE	8161	FALSE	8155	99.9264796
AGE	8161	TRUE	6	0.0735204
BLUEBOOK	8161	FALSE	8161	100.0000000
CAR_AGE	8161	FALSE	7651	93.7507658
CAR_AGE	8161	TRUE	510	6.2492342
CLAIM_TOT	8161	FALSE	8161	100.0000000
CLIENT_LONG	8161	FALSE	8161	100.0000000
CLM_FREQ	8161	FALSE	8161	100.0000000
HOME_VAL	8161	FALSE	7697	94.3144223
HOME_VAL	8161	TRUE	464	5.6855777
INCOME	8161	FALSE	7716	94.5472369
INCOME	8161	TRUE	445	5.4527631
INJOB	8161	FALSE	7707	94.4369563
INJOB	8161	TRUE	454	5.5630437
KIDS	8161	FALSE	8161	100.0000000
POINTS	8161	FALSE	8161	100.0000000
TARGET_FLAG	8161	FALSE	8161	100.0000000
TEEN_PERMIT	8161	FALSE	8161	100.0000000
TRAVTIME	8161	FALSE	8161	100.0000000

in the case of our numerical variables we will proceed to use a median imputation to complete the data that is missing.

Factor Variables

variable	total	isna	num.isna	pct
CAR_TYPE	8161	FALSE	8161	100.000000
CAR_USE	8161	FALSE	8161	100.000000
EDUCATION	8161	FALSE	8161	100.000000
JOB	8161	FALSE	7635	93.554711
JOB	8161	TRUE	526	6.445289
MARRIED	8161	FALSE	8161	100.000000
RED_CAR	8161	FALSE	8161	100.000000
REVOKED	8161	FALSE	8161	100.000000
SEX	8161	FALSE	8161	100.000000
SINGL_PARENT	8161	FALSE	8161	100.000000
TARGET_FLAG	8161	FALSE	8161	100.000000
URBANICITY	8161	FALSE	8161	100.000000

Variable	Median Imputation
AGE	TRUE
CAR_AGE	TRUE
HOME_VAL	TRUE

Variable	Median Imputation
INCOME	TRUE
INJOB	TRUE

Correlation

In order to determine the best predictor for our model we need to detect which are the predictor variables with low correlation value. We use the corrr package to determine all the variables with values <0.10. This will allow us to only manipulate the variables that have significance to our model.

TARGET_AMT

```
##           term TARGET_AMT
## 1  TEEN_PERMIT        .10
## 2          AGE       -.10
## 3         KIDS        .13
## 4        INJOB       -.05
## 5       INCOME       -.15
## 6     HOME_VAL       -.19
## 7     TRAVTIME        .06
## 8     BLUEBOOK       -.11
## 9  CLIENT_LONG       -.08
## 10   CLAIM_TOT        .23
## 11    CLM_FREQ        .23
## 12      POINTS        .19
## 13     CAR_AGE       -.10
```

TARGET_FLAG

```
##
## Correlation method: 'spearman'
## Missing treated using: 'pairwise.complete.obs'
```

```
##           term TARGET_FLAG
## 1  TEEN_PERMIT         .10
## 2          AGE        -.10
## 3         KIDS         .13
## 4        INJOB        -.06
## 5       INCOME        -.15
## 6     HOME_VAL        -.19
## 7     TRAVTIME         .06
## 8     BLUEBOOK        -.11
## 9  CLIENT_LONG        -.08
## 10   CLAIM_TOT         .23
## 11    CLM_FREQ         .24
## 12      POINTS         .19
## 13     CAR_AGE        -.10
```

Preprocess

Recommended pre processing outline

While every individual project's needs are different, here is a suggested order of *potential* steps that should work for most problems:

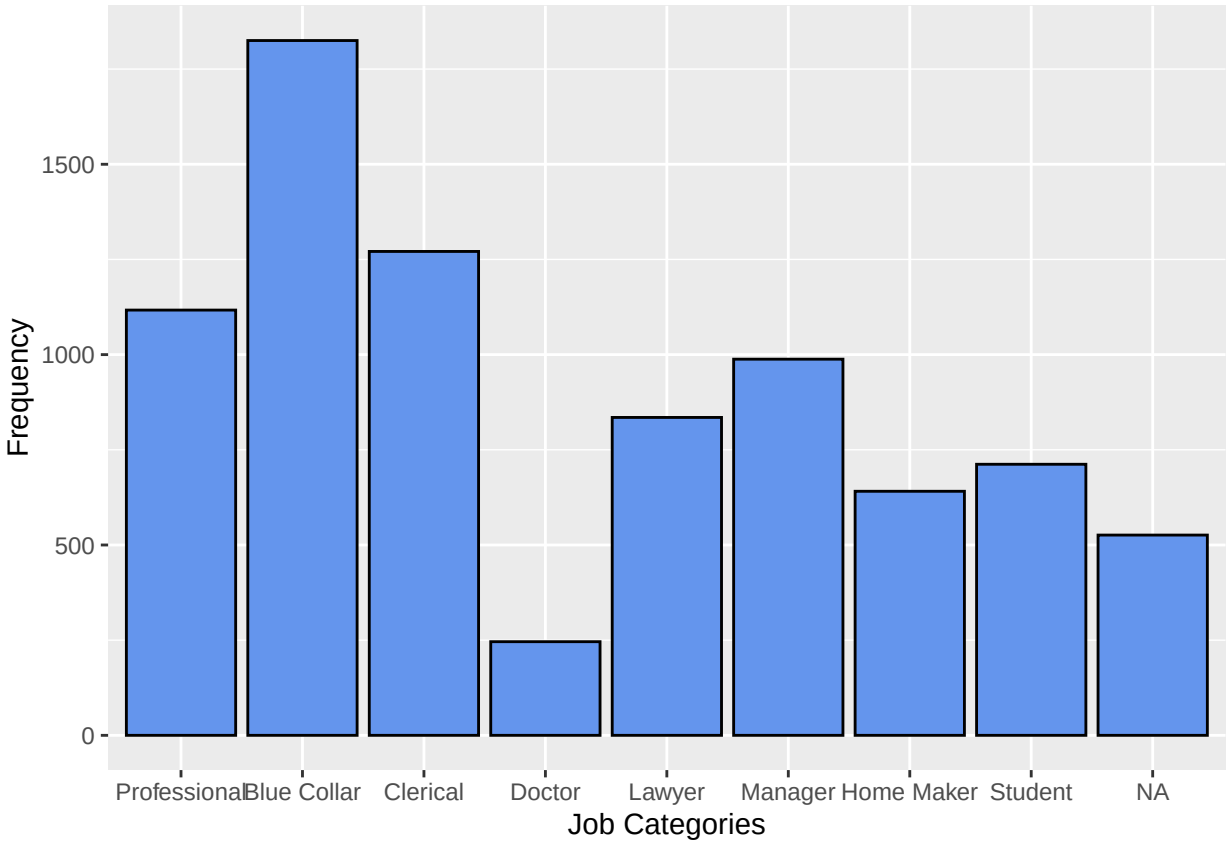
1. Impute
2. Handle factor levels
3. Individual transformations for skewness and other issues
4. Discretize (if needed and if you have no other choice)
5. Create dummy variables
6. Create interactions
7. Normalization steps (center, scale, range, etc)
8. Multivariate transformation (e.g. PCA, spatial sign, etc)

Lets take a look at our dataset now.

```
## # A tibble: 2 x 3
##   TARGET_FLAG count  prop
##   <fct>       <int> <dbl>
## 1 0            6008 0.736
## 2 1            2153 0.264
```

Was car in a crash?	Target var codes	Percent Frequency
Yes	1	26%
No	0	74%

We can see that the only variable with missing values is JOBS. The percentage of missing values in JOBS is 6.44%. When we build our recipe we will take care of this factor variable.


Frequency
1500
1000
500
0
Professional
Blue Collar
Clerical
Doctor
Lawyer
Manager
Home Maker
Student
NA
Job Categories

Partition

The tidymodels `rsample` library handles data splitting. Training and testing split is done as shown,.

To get started, let's split this single dataset into two: a *training* set and a *testing* set. We'll keep most of the rows in the original dataset (subset chosen randomly) in the *training* set. The training data will be used to *fit* the model, and the *testing* set will be used to measure model performance.

To do this, we can use the `rsample` package to create an object that contains the information on *how* to split the data, and then two more rsample functions to create data frames for the training and testing sets:

Remember that we already partitioned our data set into a *training set* and *test set*. This lets us judge whether a given model will generalize well to new data. However, using only two partitions may be insufficient when doing many rounds of hyperparameter tuning (which we don't perform in this tutorial but it is always recommended to use a validation set).

Therefore, it is usually a good idea to create a so called `validation set`. Watch this short video from Google's Machine Learning crash course to learn more about the value of a validation set.

We use k-fold crossvalidation to build a set of 5 validation folds with the function `vfold_cv`. We also use stratified sampling.

We will come back to the *validation set* after we specified our models.

Recipes & Roles

The recipes package is our go-to for defining the steps that will be used to pre-process the data. We will see examples for imputing missing data, one hot encoding the dummy variables (transforming the variables to where each level has a column), removing near zero variance predictors, and normalizing the predictors.

To get started, let's create a recipe for a simple **logistic regression** model. Before training the model, we can use a recipe to create a few new predictors and conduct some preprocessing required by the model.

The type of data preprocessing is dependent on the data and the type of model being fit. The excellent book "Tidy Modeling with R" provides an appendix with recommendations for baseline levels of preprocessing that are needed for various model functions.

Let's create a base `recipe` for all of our classification models. Note that the sequence of steps matter:

- The `recipe()` function has two arguments:
    1. A formula. Any variable on the left-hand side of the tilde (`~`) is considered the model outcome (here, `price_category`). On the right-hand side of the tilde are the predictors. Variables may be listed by name (separated by a `+`), or you can use the dot (`.`) to indicate all other variables as predictors.
    2. The data. A recipe is associated with the data set used to create the model. This will typically be the training set, so `data = train_data` here.

Let's initiate a new recipe:

Recipe 1 Recipe 1 includes BOXCOX transformations

```
## # A tibble: 25 x 4
##    variable     type    role      source
##    <chr>        <chr>   <chr>     <chr>
##  1 TARGET_AMT   numeric ID        original
##  2 TEEN_PERMIT  numeric predictor original
##  3 AGE          numeric predictor original
##  4 KIDS         numeric predictor original
##  5 INJOB        numeric predictor original
##  6 INCOME       numeric predictor original
##  7 SINGL_PARENT nominal predictor original
##  8 HOME_VAL     numeric predictor original
##  9 MARRIED      nominal predictor original
## 10 SEX          nominal predictor original
## # ... with 15 more rows
```

```
## Rows: 6,120
## Columns: 36
## $ TARGET_AMT           <dbl> 1254.00, 0.00, 1493.00, 1373.00, 0.00, 7400.00, ~
## $ TEEN_PERMIT          <dbl> -0.337013, -0.337013, 1.627291, -0.337013, -0.33~
## $ AGE                  <dbl> -1.36390608, -1.02264795, -0.10203666, 0.5973115~
## $ KIDS                 <dbl> 1.1472574, -0.6397661, 0.2537457, -0.6397661, -0~
## $ INJOB                <dbl> 0.1270420, -0.1236018, -2.6300396, -2.6300396, -~
## $ INCOME               <dbl> -0.902695019, 0.085828954, -1.337248016, -1.3372~
## $ HOME_VAL             <dbl> 0.01976078, -1.24318801, -1.24318801, -0.7979511~
## $ TRAVTIME             <dbl> 1.141505939, 0.702215199, -0.803180766, -0.32475~
## $ BLUEBOOK             <dbl> -1.1108248, 0.4704351, -0.7910831, -2.0783840, 1~
## $ CLIENT_LONG          <dbl> 1.104152375, -1.303010593, 0.643570590, -1.30301~
## $ CLM_FREQ             <dbl> 0.1723238, -0.6885919, -0.6885919, 0.1723238, -0~
## $ POINTS               <dbl> -0.7842938, -0.7842938, -0.3201338, -0.3201338, ~
## $ CAR_AGE              <dbl> -0.2341261, 0.1298652, -1.3261002, 0.1298652, -0~
```

```
## $ TARGET_FLAG           <fct> 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, ~
## $ SINGL_PARENT_Yes      <dbl> -0.3885876, -0.3885876, -0.3885876, -0.3885876, ~
## $ MARRIED_Yes           <dbl> 0.8153186, -1.2263139, 0.8153186, 0.8153186, 0.8~
## $ SEX_F                 <dbl> 0.9240369, 0.9240369, -1.0820311, 0.9240369, 0.9~
## $ EDUCATION_High.School <dbl> 1.1388965, 1.1388965, 1.1388965, -0.8778994, 1.1~
## $ EDUCATION_Bachelors   <dbl> -0.6145698, -0.6145698, -0.6145698, -0.6145698, ~
## $ EDUCATION_Masters     <dbl> -0.5065882, -0.5065882, -0.5065882, -0.5065882, ~
## $ JOB_Blue.Collar       <dbl> 1.5756917, -0.6345382, -0.6345382, -0.6345382, -~
## $ JOB_Clerical          <dbl> -0.4288955, -0.4288955, -0.4288955, -0.4288955, ~
## $ JOB_Doctor            <dbl> -0.1784972, -0.1784972, -0.1784972, -0.1784972, ~
## $ JOB_Lawyer            <dbl> -0.335420, -0.335420, -0.335420, -0.335420, -0.3~
## $ JOB_Manager           <dbl> -0.3699867, -0.3699867, -0.3699867, -0.3699867, ~
## $ JOB_Home.Maker        <dbl> -0.2926942, -0.2926942, -0.2926942, 3.4159772, -~
## $ JOB_Student           <dbl> -0.3129517, -0.3129517, 3.1948589, -0.3129517, -~
## $ CAR_USE_Commercial    <dbl> 1.2999360, -0.7691429, 1.2999360, -0.7691429, -0~
## $ CAR_TYPE_SUV          <dbl> -0.6312467, 1.5839079, -0.6312467, -0.6312467, -~
## $ CAR_TYPE_Sports.Car   <dbl> 2.8494541, -0.3508871, -0.3508871, 2.8494541, -0~
## $ CAR_TYPE_Van          <dbl> -0.309161, -0.309161, -0.309161, -0.309161, -0.3~
## $ CAR_TYPE_Panel.Truck  <dbl> -0.3005183, -0.3005183, -0.3005183, -0.3005183, ~
## $ CAR_TYPE_Pickup       <dbl> -0.4589641, -0.4589641, 2.1784635, -0.4589641, 2~
## $ RED_CAR_no            <dbl> 0.6398608, 0.6398608, -1.5625846, 0.6398608, 0.6~
## $ REVOKED_Yes           <dbl> 2.6899247, -0.3716969, -0.3716969, -0.3716969, 2~
## $ URBANICITY_Rural      <dbl> -0.5076066, 1.9697078, -0.5076066, -0.5076066, -~
```

Recipe 2 For this classification model we will change the recipe removing BOXCOX and correlation variables

```
## # A tibble: 25 x 4
##    variable     type    role      source
##    <chr>        <chr>   <chr>     <chr>
##  1 TARGET_AMT   numeric ID        original
##  2 TEEN_PERMIT  numeric predictor original
##  3 AGE          numeric predictor original
##  4 KIDS         numeric predictor original
##  5 INJOB        numeric predictor original
##  6 INCOME       numeric predictor original
##  7 SINGL_PARENT nominal predictor original
##  8 HOME_VAL     numeric predictor original
##  9 MARRIED      nominal predictor original
## 10 SEX          nominal predictor original
## # ... with 15 more rows
```

```
## Rows: 6,120
## Columns: 37
## $ TARGET_AMT            <dbl> 1254.00, 0.00, 1493.00, 1373.00, 0.00, 7400.00, ~
## $ TEEN_PERMIT           <dbl> -0.337013, -0.337013, 1.627291, -0.337013, -0.33~
## $ AGE                   <dbl> -1.36933553, -1.02156537, -0.09417828, 0.6013620~
## $ KIDS                  <dbl> 1.1472574, -0.6397661, 0.2537457, -0.6397661, -0~
## $ INJOB                 <dbl> 0.1270420, -0.1236018, -2.6300396, -2.6300396, -~
## $ INCOME                <dbl> -0.902695019, 0.085828954, -1.337248016, -1.3372~
## $ HOME_VAL              <dbl> 0.01976078, -1.24318801, -1.24318801, -0.7979511~
## $ TRAVTIME              <dbl> 1.17218258, 0.66869476, -0.84176870, -0.40121686~
## $ BLUEBOOK              <dbl> -1.051826968, 0.342518249, -0.829160396, -1.5519~
## $ CLIENT_LONG           <dbl> 1.1255936, -1.0463348, 0.4016175, -1.0463348, -0~
## $ CLAIM_TOT             <dbl> 1.4710866, -0.4610917, -0.4610917, 0.1776440, -0~
## $ CLM_FREQ              <dbl> 0.1723238, -0.6885919, -0.6885919, 0.1723238, -0~
## $ POINTS                <dbl> -0.7842938, -0.7842938, -0.3201338, -0.3201338, ~
```

```
## $ CAR_AGE               <dbl> -0.2341261, 0.1298652, -1.3261002, 0.1298652, -0~
## $ TARGET_FLAG           <fct> 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, ~
## $ SINGL_PARENT_Yes      <dbl> -0.3885876, -0.3885876, -0.3885876, -0.3885876, ~
## $ MARRIED_Yes           <dbl> 0.8153186, -1.2263139, 0.8153186, 0.8153186, 0.8~
## $ SEX_F                 <dbl> 0.9240369, 0.9240369, -1.0820311, 0.9240369, 0.9~
## $ EDUCATION_High.School <dbl> 1.1388965, 1.1388965, 1.1388965, -0.8778994, 1.1~
## $ EDUCATION_Bachelors   <dbl> -0.6145698, -0.6145698, -0.6145698, -0.6145698, ~
## $ EDUCATION_Masters     <dbl> -0.5065882, -0.5065882, -0.5065882, -0.5065882, ~
## $ JOB_Blue.Collar       <dbl> 1.5756917, -0.6345382, -0.6345382, -0.6345382, -~
## $ JOB_Clerical          <dbl> -0.4288955, -0.4288955, -0.4288955, -0.4288955, ~
## $ JOB_Doctor            <dbl> -0.1784972, -0.1784972, -0.1784972, -0.1784972, ~
## $ JOB_Lawyer            <dbl> -0.335420, -0.335420, -0.335420, -0.335420, -0.3~
## $ JOB_Manager           <dbl> -0.3699867, -0.3699867, -0.3699867, -0.3699867, ~
## $ JOB_Home.Maker        <dbl> -0.2926942, -0.2926942, -0.2926942, 3.4159772, -~
## $ JOB_Student           <dbl> -0.3129517, -0.3129517, 3.1948589, -0.3129517, -~
## $ CAR_USE_Commercial    <dbl> 1.2999360, -0.7691429, 1.2999360, -0.7691429, -0~
## $ CAR_TYPE_SUV          <dbl> -0.6312467, 1.5839079, -0.6312467, -0.6312467, -~
## $ CAR_TYPE_Sports.Car   <dbl> 2.8494541, -0.3508871, -0.3508871, 2.8494541, -0~
## $ CAR_TYPE_Van          <dbl> -0.309161, -0.309161, -0.309161, -0.309161, -0.3~
## $ CAR_TYPE_Panel.Truck  <dbl> -0.3005183, -0.3005183, -0.3005183, -0.3005183, ~
## $ CAR_TYPE_Pickup       <dbl> -0.4589641, -0.4589641, 2.1784635, -0.4589641, 2~
## $ RED_CAR_no            <dbl> 0.6398608, 0.6398608, -1.5625846, 0.6398608, 0.6~
## $ REVOKED_Yes           <dbl> 2.6899247, -0.3716969, -0.3716969, -0.3716969, 2~
## $ URBANICITY_Rural      <dbl> -0.5076066, 1.9697078, -0.5076066, -0.5076066, -~
```

Recipe 3 For this Multiple Linear regression model we will change the recipe removing TARGET_FLAG for TARGET_AMT

```
## Rows: 6,120
## Columns: 37
## $ TARGET_FLAG           <fct> 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, ~
## $ TEEN_PERMIT           <dbl> -0.337013, -0.337013, 1.627291, -0.337013, -0.33~
## $ AGE                   <dbl> -1.36933553, -1.02156537, -0.09417828, 0.6013620~
## $ KIDS                  <dbl> 1.1472574, -0.6397661, 0.2537457, -0.6397661, -0~
## $ INJOB                 <dbl> 0.1270420, -0.1236018, -2.6300396, -2.6300396, -~
## $ INCOME                <dbl> -0.902695019, 0.085828954, -1.337248016, -1.3372~
## $ HOME_VAL              <dbl> 0.01976078, -1.24318801, -1.24318801, -0.7979511~
## $ TRAVTIME              <dbl> 1.17218258, 0.66869476, -0.84176870, -0.40121686~
## $ BLUEBOOK              <dbl> -1.051826968, 0.342518249, -0.829160396, -1.5519~
## $ CLIENT_LONG           <dbl> 1.1255936, -1.0463348, 0.4016175, -1.0463348, -0~
## $ CLAIM_TOT             <dbl> 1.4710866, -0.4610917, -0.4610917, 0.1776440, -0~
## $ CLM_FREQ              <dbl> 0.1723238, -0.6885919, -0.6885919, 0.1723238, -0~
## $ POINTS                <dbl> -0.7842938, -0.7842938, -0.3201338, -0.3201338, ~
## $ CAR_AGE               <dbl> -0.2341261, 0.1298652, -1.3261002, 0.1298652, -0~
## $ TARGET_AMT            <dbl> 1254.00, 0.00, 1493.00, 1373.00, 0.00, 7400.00, ~
## $ SINGL_PARENT_Yes      <dbl> -0.3885876, -0.3885876, -0.3885876, -0.3885876, ~
## $ MARRIED_Yes           <dbl> 0.8153186, -1.2263139, 0.8153186, 0.8153186, 0.8~
## $ SEX_F                 <dbl> 0.9240369, 0.9240369, -1.0820311, 0.9240369, 0.9~
## $ EDUCATION_High.School <dbl> 1.1388965, 1.1388965, 1.1388965, -0.8778994, 1.1~
## $ EDUCATION_Bachelors   <dbl> -0.6145698, -0.6145698, -0.6145698, -0.6145698, ~
## $ EDUCATION_Masters     <dbl> -0.5065882, -0.5065882, -0.5065882, -0.5065882, ~
## $ JOB_Blue.Collar       <dbl> 1.5756917, -0.6345382, -0.6345382, -0.6345382, -~
## $ JOB_Clerical          <dbl> -0.4288955, -0.4288955, -0.4288955, -0.4288955, ~
## $ JOB_Doctor            <dbl> -0.1784972, -0.1784972, -0.1784972, -0.1784972, ~
## $ JOB_Lawyer            <dbl> -0.335420, -0.335420, -0.335420, -0.335420, -0.3~
## $ JOB_Manager           <dbl> -0.3699867, -0.3699867, -0.3699867, -0.3699867, ~
```

```
## $ JOB_Home.Maker        <dbl> -0.2926942, -0.2926942, -0.2926942, 3.4159772, -~
## $ JOB_Student           <dbl> -0.3129517, -0.3129517, 3.1948589, -0.3129517, -~
## $ CAR_USE_Commercial    <dbl> 1.2999360, -0.7691429, 1.2999360, -0.7691429, -0~
## $ CAR_TYPE_SUV          <dbl> -0.6312467, 1.5839079, -0.6312467, -0.6312467, -~
## $ CAR_TYPE_Sports.Car   <dbl> 2.8494541, -0.3508871, -0.3508871, 2.8494541, -0~
## $ CAR_TYPE_Van          <dbl> -0.309161, -0.309161, -0.309161, -0.309161, -0.3~
## $ CAR_TYPE_Panel.Truck  <dbl> -0.3005183, -0.3005183, -0.3005183, -0.3005183, ~
## $ CAR_TYPE_Pickup       <dbl> -0.4589641, -0.4589641, 2.1784635, -0.4589641, 2~
## $ RED_CAR_no            <dbl> 0.6398608, 0.6398608, -1.5625846, 0.6398608, 0.6~
## $ REVOKED_Yes           <dbl> 2.6899247, -0.3716969, -0.3716969, -0.3716969, 2~
## $ URBANICITY_Rural      <dbl> -0.5076066, 1.9697078, -0.5076066, -0.5076066, -~
```

3. Building Models

Logistic Regression

Next, we specify our linear regression model with `parsnip`.

```
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

Random Forest

For this classification model we will change the recipe including log numerical data, removing high correlation variables and removing any numeric variables that have zero variance.

```
## Random Forest Model Specification (classification)
##
## Engine-Specific Arguments:
##   importance = impurity
##
## Computational engine: ranger
```

Boosted Tree

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice

## Boosted Tree Model Specification (classification)
##
## Computational engine: xgboost
```

LM LINEAR Regression

Next, we specify our linear regression model with `parsnip`.

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

GLM LINEAR Regression

```
## Linear Regression Model Specification (regression)
##
## Computational engine: glm
```

3.2 Workflow Creation

To combine the data preparation recipe with the model building, we use the package workflows. A workflow is an object that can bundle together your pre-processing recipe, modeling, and even post-processing requests (like calculating the RMSE).

Logistic Regression

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 8 Recipe Steps
##
## * step_impute_median()
## * step_impute_mode()
## * step_BoxCox()
## * step_novel()
## * step_dummy()
## * step_zv()
## * step_corr()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

Random Forest

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 6 Recipe Steps
##
## * step_impute_median()
## * step_impute_mode()
## * step_novel()
## * step_dummy()
## * step_zv()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Engine-Specific Arguments:
##   importance = impurity
##
## Computational engine: ranger
```

Boosted Tree

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: boost_tree()
##
## -- Preprocessor ----------------------------------------------------------------
## 8 Recipe Steps
##
## * step_impute_median()
## * step_impute_mode()
## * step_BoxCox()
## * step_novel()
## * step_dummy()
## * step_zv()
## * step_corr()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Boosted Tree Model Specification (classification)
##
## Computational engine: xgboost
```

LM LINEAR Regression

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 6 Recipe Steps
##
## * step_impute_median()
## * step_impute_mode()
## * step_novel()
## * step_dummy()
## * step_zv()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

GLM LINEAR Regression

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 6 Recipe Steps
##
## * step_impute_median()
## * step_impute_mode()
```

```
## * step_novel()
## * step_dummy()
## * step_zv()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------
## Linear Regression Model Specification (regression)
##
## Computational engine: glm
```

4. Evaluate Models

Now we can use our validation set (`cv_folds`) to estimate the performance of our models using the `fit_resamples()` function to fit the models on each of the folds and store the results.

Note that `fit_resamples()` will fit our model to each resample and evaluate on the heldout set from each resample. The function is usually only used for computing performance metrics across some set of resamples to evaluate our models (like accuracy) - the models are not even stored. However, in our example we save the predictions in order to visualize the model fit and residuals with `control_resamples(save_pred = TRUE)`.

Finally, we collect the performance metrics with `collect_metrics()` and pick the model that does best on the validation set.

Logistic Regression

We use our workflow object to perform resampling. Furthermore, we use `metric_set()`to choose some common classification performance metrics provided by the `yardstick` package. Visit `yardsticks` reference to see the complete list of all possible metrics.

Show average performance over all folds

```
## # A tibble: 8 x 6
##   .metric   .estimator  mean     n std_err .config
##   <chr>     <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary     0.791     5 0.00300 Preprocessor1_Model1
## 2 f_meas    binary     0.867     5 0.00160 Preprocessor1_Model1
## 3 kap       binary     0.390     5 0.0121  Preprocessor1_Model1
## 4 precision binary     0.816     5 0.00333 Preprocessor1_Model1
## 5 recall    binary     0.925     5 0.00163 Preprocessor1_Model1
## 6 roc_auc   binary     0.808     5 0.00364 Preprocessor1_Model1
## 7 sens      binary     0.925     5 0.00163 Preprocessor1_Model1
## 8 spec      binary     0.419     5 0.0135  Preprocessor1_Model1
```

Show performance for every single fold:

```
## # A tibble: 40 x 5
##    id    .metric   .estimator .estimate .config
##    <chr> <chr>     <chr>          <dbl> <chr>
##  1 Fold1 recall    binary         0.920 Preprocessor1_Model1
##  2 Fold1 precision binary         0.827 Preprocessor1_Model1
##  3 Fold1 f_meas    binary         0.871 Preprocessor1_Model1
##  4 Fold1 accuracy  binary         0.799 Preprocessor1_Model1
##  5 Fold1 kap       binary         0.424 Preprocessor1_Model1
##  6 Fold1 sens      binary         0.920 Preprocessor1_Model1
##  7 Fold1 spec      binary         0.461 Preprocessor1_Model1
##  8 Fold1 roc_auc   binary         0.814 Preprocessor1_Model1
##  9 Fold2 recall    binary         0.927 Preprocessor1_Model1
## 10 Fold2 precision binary         0.806 Preprocessor1_Model1
## # ... with 30 more rows
```

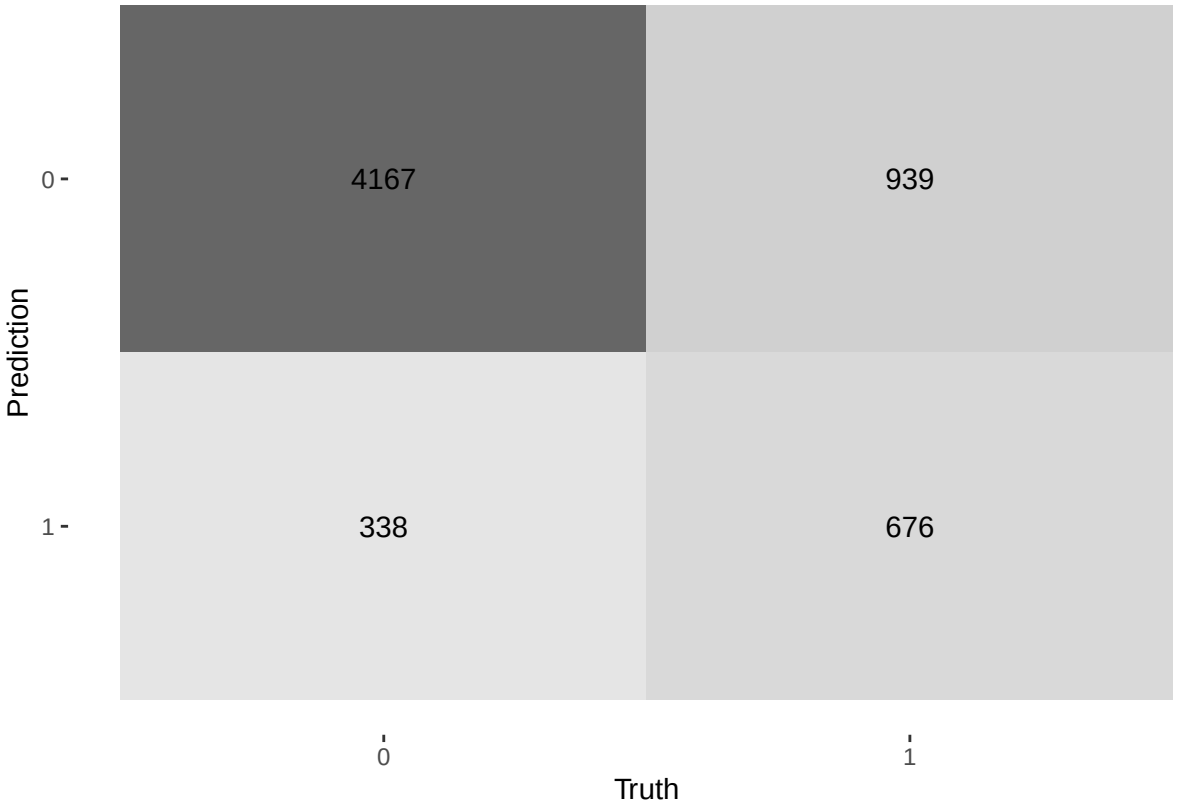
Collect Predictions

To obtain the actual model predictions, we use the function `collect_predictions` and save the result as `log_pred`:

Confusion Matrix

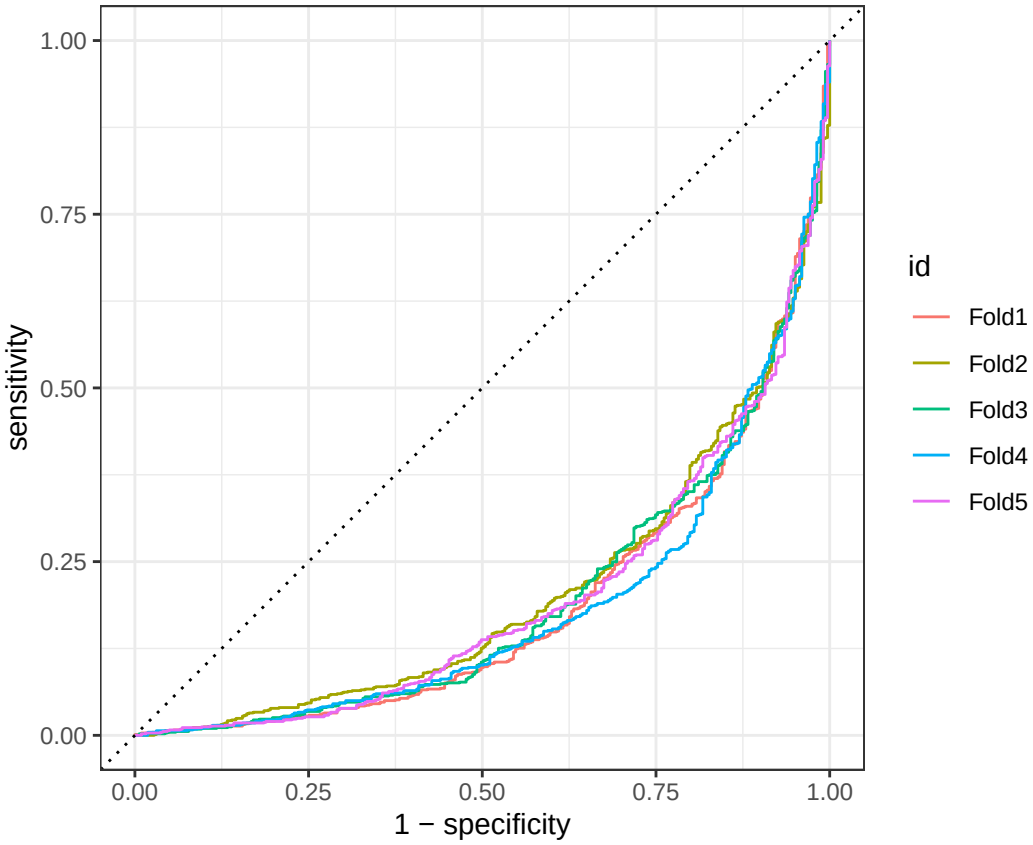
Now we can use the predictions to create a *confusion matrix* with `conf_mat()`:

```
##            Truth
## Prediction    0    1
##          0 4167  939
##          1  338  676
```

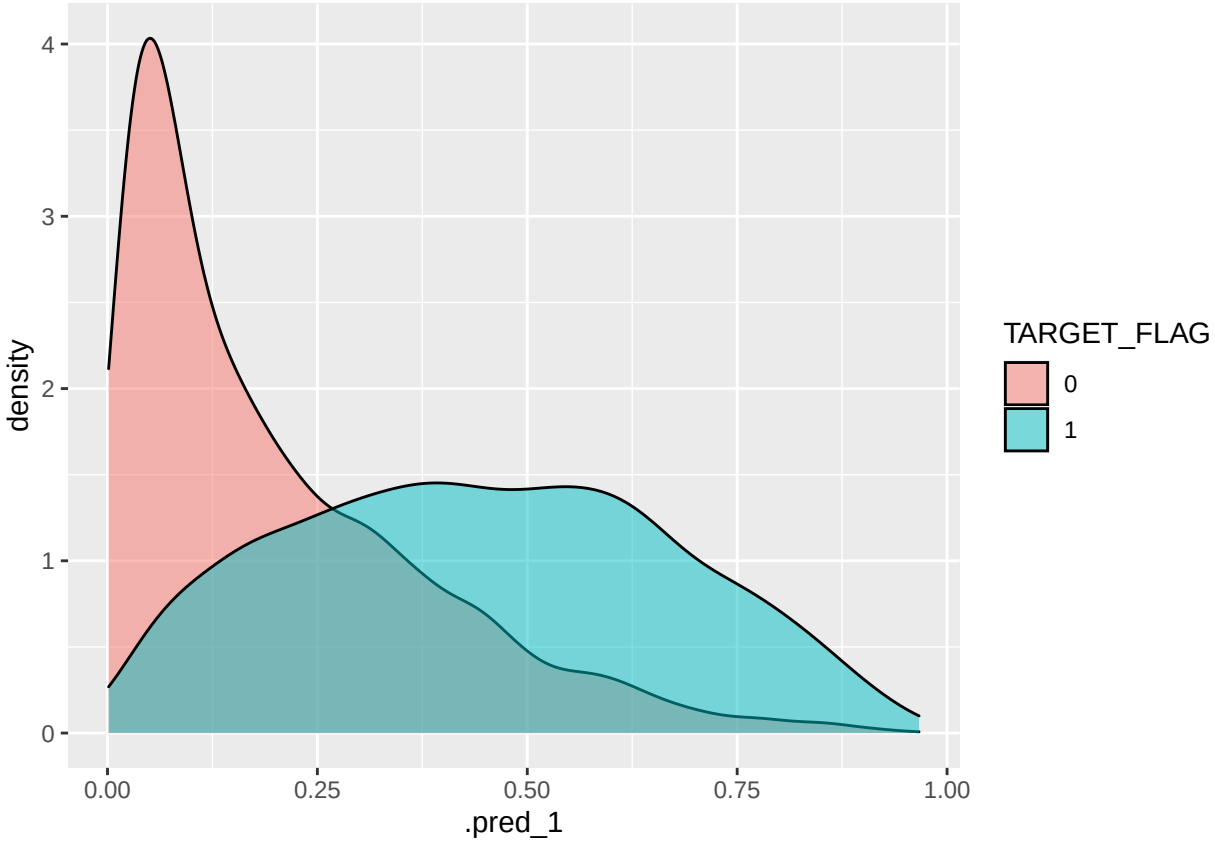


ROC-Curve

We can also make an ROC curve for our 5 folds. Since the category we are predicting is the second level in the TARGET_FLAG factor ("1"), we provide `roc_curve()` with the relevant class probability `.pred_1`:



Probability Distributions



Random Forests

We don't repeat all of the steps shown in logistic regression and just focus on the performance metrics

```
## # A tibble: 8 x 6
##   .metric   .estimator  mean     n std_err .config
##   <chr>     <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary     0.783     5 0.00480 Preprocessor1_Model1
## 2 f_meas    binary     0.866     5 0.00268 Preprocessor1_Model1
## 3 kap       binary     0.324     5 0.0188  Preprocessor1_Model1
## 4 precision binary     0.795     5 0.00400 Preprocessor1_Model1
## 5 recall    binary     0.950     5 0.00200 Preprocessor1_Model1
## 6 roc_auc   binary     0.804     5 0.00668 Preprocessor1_Model1
## 7 sens      binary     0.950     5 0.00200 Preprocessor1_Model1
## 8 spec      binary     0.318     5 0.0166  Preprocessor1_Model1
```

Xboost

```
## ! Fold1: preprocessor 1/1: Non-positive values in selected variable., No Box-Cox ...

## ! Fold2: preprocessor 1/1: Non-positive values in selected variable., No Box-Cox ...

## ! Fold3: preprocessor 1/1: Non-positive values in selected variable., No Box-Cox ...

## ! Fold4: preprocessor 1/1: Non-positive values in selected variable., No Box-Cox ...

## ! Fold5: preprocessor 1/1: Non-positive values in selected variable., No Box-Cox ...

## # A tibble: 8 x 6
##   .metric   .estimator  mean     n std_err .config
##   <chr>     <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary     0.782     5 0.00384 Preprocessor1_Model1
## 2 f_meas    binary     0.861     5 0.00216 Preprocessor1_Model1
## 3 kap       binary     0.362     5 0.0152  Preprocessor1_Model1
## 4 precision binary     0.811     5 0.00393 Preprocessor1_Model1
## 5 recall    binary     0.918     5 0.00301 Preprocessor1_Model1
## 6 roc_auc   binary     0.802     5 0.00682 Preprocessor1_Model1
## 7 sens      binary     0.918     5 0.00301 Preprocessor1_Model1
## 8 spec      binary     0.401     5 0.0164  Preprocessor1_Model1
```

LINEAR REGRESSION

First we build a validation set with K-fold crossvalidation for linear regression:

```
## #  5-fold cross-validation using stratification
## # A tibble: 5 x 2
##   splits              id
##   <list>              <chr>
## 1 <split [4895/1225]> Fold1
## 2 <split [4896/1224]> Fold2
## 3 <split [4896/1224]> Fold3
## 4 <split [4896/1224]> Fold4
## 5 <split [4897/1223]> Fold5
```

LM LINEAR REGRESSION

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

```
## # A tibble: 2 x 6
##   .metric .estimator      mean     n   std_err .config
##   <chr>   <chr>          <dbl> <int>     <dbl> <chr>
## 1 rmse    standard   4471.         5 220.      Preprocessor1_Model1
## 2 rsq     standard      0.0588     5   0.00645 Preprocessor1_Model1
```

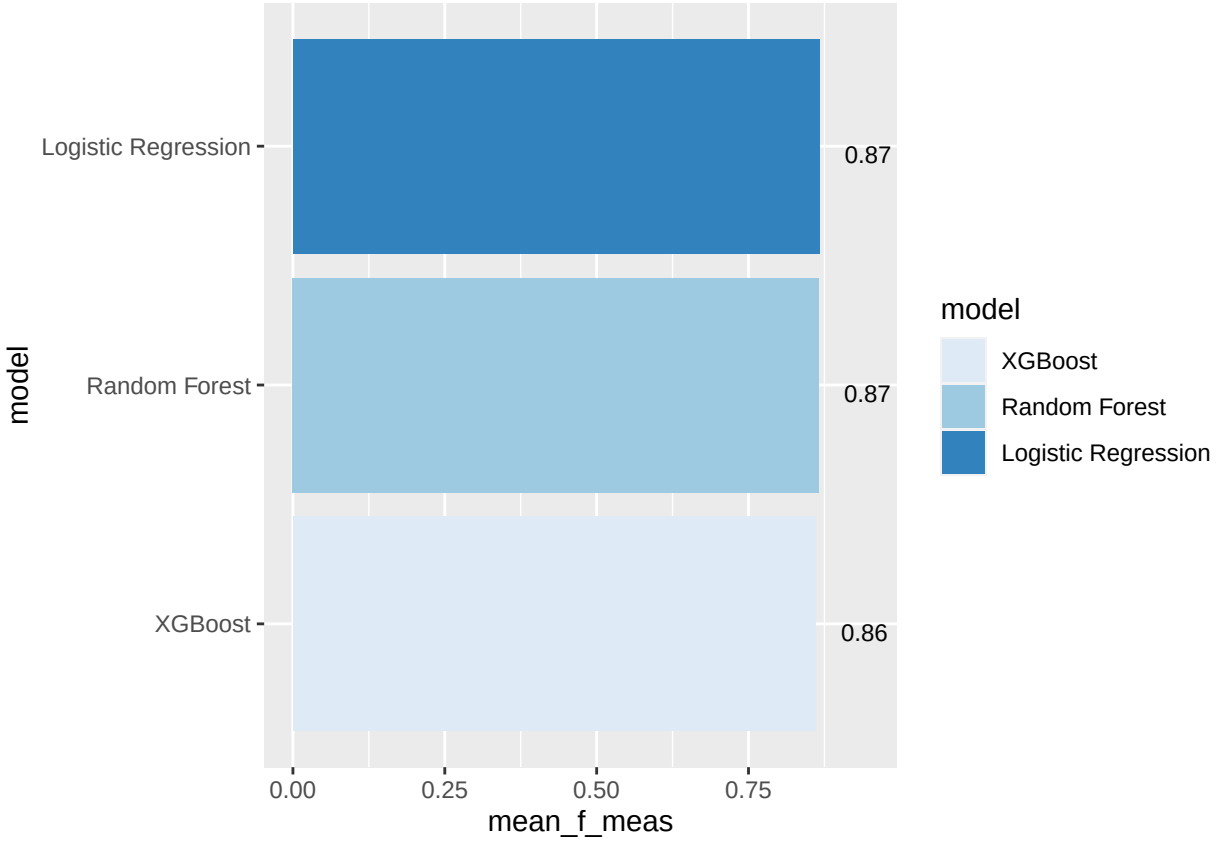
GLM LINEAR REGRESSION

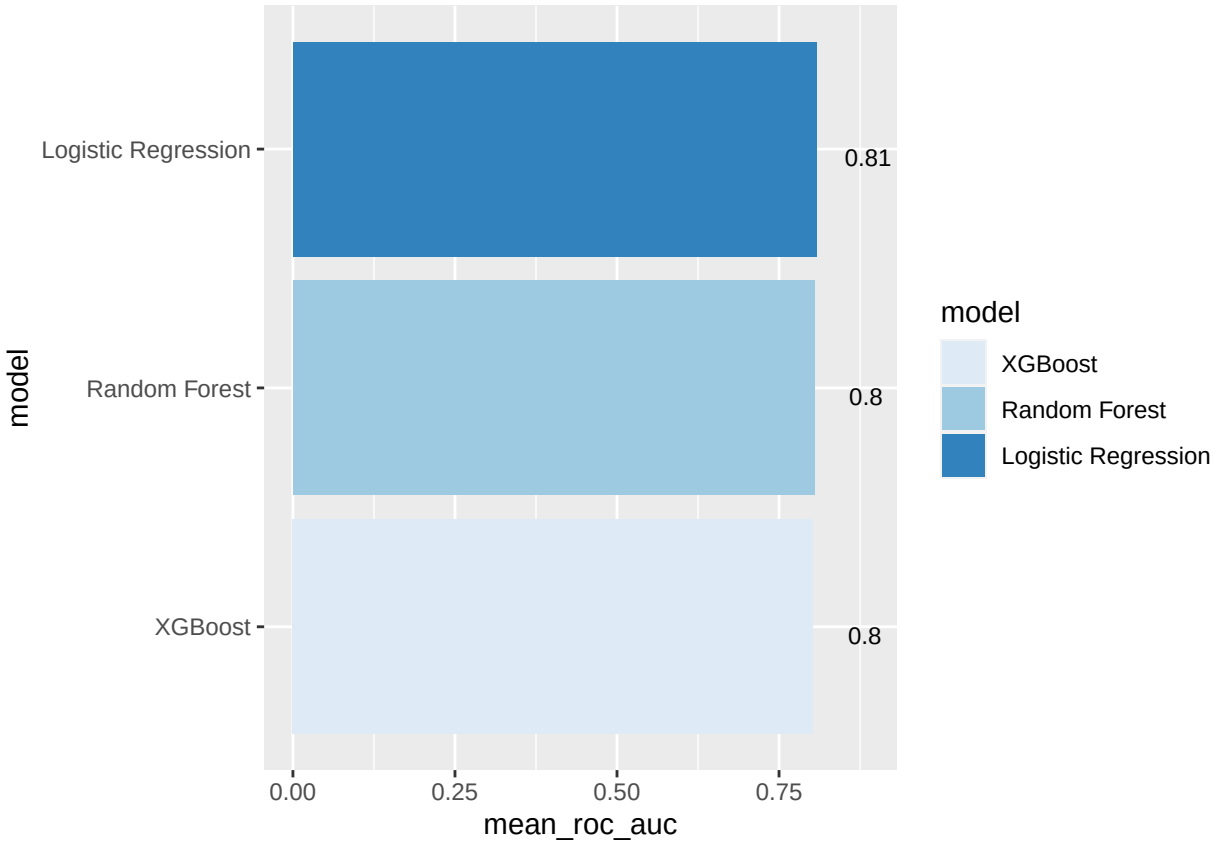
```
## Warning: The `...` are not used in this function but one or more objects were
## passed: ''
```

```
## # A tibble: 2 x 6
##   .metric .estimator      mean     n   std_err .config
##   <chr>   <chr>          <dbl> <int>     <dbl> <chr>
## 1 rmse    standard   4471.         5 220.      Preprocessor1_Model1
## 2 rsq     standard      0.0588     5   0.00645 Preprocessor1_Model1
```

Final Selection

The predictions show that there are still underlying problems with hte models that need to be resolved. All 3 models produced predictions that are not on par with the distributions in the original training dataset indicating that they are not a good fit for the data. This is evidenced inthe histogrema below as well as in the summary statistics above.





Note that the model results are all quite similar. In our example we choose the F1-Score as performance measure to select the best model. Let's find the maximum mean F1-Score:

```
## # A tibble: 1 x 17
##   model            mean_accuracy mean_f_meas mean_kap mean_precision mean_recall
##   <chr>                    <dbl>       <dbl>    <dbl>          <dbl>       <dbl>
## 1 Logistic Regres~         0.791       0.867    0.390          0.816       0.925
## # ... with 11 more variables: mean_roc_auc <dbl>, mean_sens <dbl>,
## #   mean_spec <dbl>, std_err_accuracy <dbl>, std_err_f_meas <dbl>,
## #   std_err_kap <dbl>, std_err_precision <dbl>, std_err_recall <dbl>,
## #   std_err_roc_auc <dbl>, std_err_sens <dbl>, std_err_spec <dbl>
```

References

Ordering of steps • recipes (tidymodels.org)

Linear regression — linear_reg • parsnip (tidymodels.org)

Linear regression via lm — details_linear_reg_lm • parsnip (tidymodels.org)

(5) Evaluating ML Performance, Resampling, and Workflows in "tidymodels" | R Tutorial (2021) - YouTube

Apendix

```
knitr::opts_chunk$set(echo = FALSE)

library(tidyverse)
```

```
library(tidymodels)
library(skimr)
library(tinytex)
library(e1071)
library(ggthemes)
#library(caret)
df <- read_csv("D:/PORTFOLIO_DS/Portfolio_R/Business_Analytics/Business_Analytics/Projects/PROJECT_HW4/Dat

# insu_eval <- read_csv("D:/PORTFOLIO_DS/Portfolio_R/Business_Analytics/Business_Analytics/Projects/PROJEC

skim_without_charts(df)

df1 <- df %>%
  rename(KIDS = HOMEKIDS,
         TEEN_PERMIT = KIDSDRIV,
         MARRIED = MSTATUS,
         POINTS = MVR_PTS,
         CLAIM_TOT = OLDCLAIM,
         SINGL_PARENT = PARENT1,
         CLIENT_LONG = TIF,
         INJOB = YOJ)

df2 <- df1 %>%
  mutate(BLUEBOOK = str_remove(BLUEBOOK, "[\\$]"),
         BLUEBOOK = as.numeric(str_remove(BLUEBOOK, "[,]"))) %>%
  mutate(HOME_VAL = str_remove(HOME_VAL, "[\\$]"),
         HOME_VAL = as.numeric(str_remove(HOME_VAL, "[,]"))) %>%
  mutate(INCOME = str_remove(INCOME, "[\\$]"),
         INCOME = as.numeric(str_remove(INCOME, "[,]"))) %>%
  mutate(CLAIM_TOT = str_remove(CLAIM_TOT, "[\\$]"),
         CLAIM_TOT = as.numeric(str_remove(CLAIM_TOT, "[,]")))


df3 <- df2 %>%
  mutate(EDUCATION = str_remove(EDUCATION, "z"),
         EDUCATION = str_remove(EDUCATION, "_"),
         EDUCATION = str_remove(EDUCATION, "<")) %>%
  mutate(MARRIED = str_remove(MARRIED, "z"),
         MARRIED = str_remove(MARRIED, "_")) %>%
  mutate(SEX = str_remove(SEX, "z"),
         SEX = str_remove(SEX, "_")) %>%
  mutate(JOB = str_remove(JOB, "z"),
         JOB = str_remove(JOB, "_")) %>%
  mutate(CAR_TYPE = str_remove(CAR_TYPE, "z"),
         CAR_TYPE = str_remove(CAR_TYPE, "_")) %>%
  mutate(URBANICITY = str_remove(URBANICITY, "z"),
         URBANICITY = str_remove(URBANICITY, "_"),
         URBANICITY = str_sub(URBANICITY, 15,19))


df3F <- df3 %>%
  mutate( across(where(is_character), as_factor) ) %>%
  mutate( TARGET_FLAG = factor(TARGET_FLAG) ) %>%
  select(-INDEX)
```

```
str(df3F)

df2_num <- df3F %>%
  select_if(., is.numeric)

df3_fct <- df3F %>%
  select_if(., is.factor)


longnum_df <- df2_num %>% pivot_longer(
    everything(),
    names_to = c("variable"),
    values_to = "value"
  )

longfct_df <- df3_fct %>% pivot_longer(
    everything(),
    names_to = c("variable"),
    values_to = "value")

ggplot(longnum_df, aes(value)) +
  geom_histogram(aes(x=value, y = ..density..),
                 colour = 4, bins = 30) +
  geom_density(aes(x=value), color = "red") +
  facet_wrap(~variable, scales = "free")
ggplot(longfct_df, aes(value)) +
  stat_count() +
  guides(x = guide_axis(angle = 90)) +
  facet_wrap(~variable, scales = "free")
ggplot(longnum_df, aes(value, variable)) +
  geom_boxplot(outlier.color = "red") +
  facet_wrap(~variable, scales = "free", drop = FALSE)+
  coord_flip()
num_scatter <- df2_num %>%
  pivot_longer(
    cols = -TARGET_AMT,
    names_to = c("variable"),
    values_to = "value")
ggplot(num_scatter, aes(x = value,
                        y = variable,
                        color = TARGET_AMT)) +
  geom_jitter(
    position = position_jitterdodge(dodge.width = 0.8,
                                    jitter.width = 0.3),
    shape=21  )
fct_skew <- df3_fct
  #convert target to factor and new names
  fct_skew$TARGET_FLAG <- recode_factor(fct_skew$TARGET_FLAG,
                                        '0' = 'No Crash',
                                        '1' = 'Crashed' )

ggplot(fct_skew, aes(x=SINGL_PARENT)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=MARRIED)) +
  stat_count(fill = 'white', colour = 'black') +
```

```
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=SEX)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=EDUCATION)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=JOB)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=CAR_USE)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=CAR_TYPE)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=RED_CAR)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=REVOKED)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
ggplot(fct_skew, aes(x=URBANICITY)) +
  stat_count(fill = 'white', colour = 'black') +
  facet_grid(TARGET_FLAG ~ .)
corr_df <- df3 %>%
  mutate( across(where(is_character), as_factor) ) %>%
  select(-INDEX, -TARGET_AMT) %>%
  select_if(., is.numeric)

library(corrplot)

corrplot(corr = cor(df2_num,
                    use = 'pairwise.complete.obs'),
         method = "ellipse",
         type = "upper",
         order = "original",
         tl.col = "black",
         tl.srt = 45,
         tl.cex = 0.55)
corrplot(corr = cor(corr_df,
                    use = 'pairwise.complete.obs'),
         method = "ellipse",
         type = "upper",
         order = "original",
         tl.col = "black",
         tl.srt = 45,
         tl.cex = 0.55)
plot((corr_df$AGE),(corr_df$KIDS))
cor.test(corr_df$AGE,corr_df$KIDS, method = "pearson")
na_df <- corr_df %>%
  pivot_longer(
    everything(),
    names_to = c("variable"),
    values_to = "value" ) %>%
  mutate(isna = is.na(value)) %>%
```

```
  group_by(variable) %>%
  mutate(total = n()) %>%
  group_by(variable,total,isna) %>%
  summarise(num.isna = n()) %>%
  mutate(pct = num.isna / total * 100)
knitr::kable(na_df)

na_df2 <- df3_fct %>%
  pivot_longer(
    everything(),
    names_to = c("variable"),
    values_to = "value" ) %>%
  mutate(isna = is.na(value)) %>%
  group_by(variable) %>%
  mutate(total = n()) %>%
  group_by(variable,total,isna) %>%
  summarise(num.isna = n()) %>%
  mutate(pct = num.isna / total * 100)
knitr::kable(na_df2)

library(corrr)
corr_tgamt <- correlate(df2_num,use = "pairwise.complete.obs",
          method = "spearman")

corr_tgamt %>%
  focus(TARGET_AMT) %>%
  fashion()
corr_tgflag <- correlate(corr_df,use = "pairwise.complete.obs",
          method = "spearman")
corr_tgflag %>%
  focus(TARGET_FLAG) %>%
  fashion()
df3F %>%
  group_by(TARGET_FLAG) %>%
  summarise(count = n() ) %>%
  mutate( prop = count / sum(count) )
# plot dist of jobs

ggplot(df3F, aes(x = JOB)) +
  geom_bar(fill = "cornflowerblue",
           color = "black") +
  labs(x = "Job Categories",
       y = "Frequency")

library(tidymodels)

set.seed(1188)
data_split <- initial_split(df3F, prop = 3/4)

train_data <- training(data_split)
test_data <- testing(data_split)

cv_folds <-
  vfold_cv(train_data,
           v = 5,
           strata = TARGET_FLAG)
```

```
auto_recipe <-
  recipe(TARGET_FLAG ~ ., data = train_data) %>%
  update_role(TARGET_AMT, new_role = "ID") %>%
  step_impute_median(AGE, CAR_AGE, HOME_VAL, INCOME, INJOB) %>%
  step_impute_mode(JOB) %>%
  step_BoxCox(all_numeric_predictors()) %>%
  step_novel(all_nominal_predictors(), -all_outcomes()) %>%
  step_dummy(all_nominal_predictors(), -all_outcomes()) %>%
  step_zv(all_numeric_predictors(), -all_outcomes()) %>%
  step_corr(all_predictors(), threshold = 0.7, method = "spearman") %>%
  step_normalize(all_numeric_predictors())



summary(auto_recipe)
prepped_data<-
  auto_recipe %>%
  prep() %>%
  juice()
glimpse(prepped_data)
auto_recipe2 <-
  recipe(TARGET_FLAG ~ ., data = train_data) %>%
  update_role(TARGET_AMT, new_role = "ID") %>%
  step_impute_median(AGE, CAR_AGE, HOME_VAL, INCOME, INJOB) %>%
  step_impute_mode(JOB) %>%
  step_novel(all_nominal_predictors(), -all_outcomes()) %>%
  step_dummy(all_nominal_predictors(), -all_outcomes()) %>%
  step_zv(all_numeric_predictors(), -all_outcomes()) %>%
  step_normalize(all_numeric_predictors())
summary(auto_recipe2)
prepped_data2<-
  auto_recipe2 %>%
  prep() %>%
  juice()
glimpse(prepped_data2)
auto_recipe3 <-
  recipe(TARGET_AMT ~ ., data = train_data) %>%
  update_role(TARGET_FLAG, new_role = "ID") %>%
  step_impute_median(AGE, CAR_AGE, HOME_VAL, INCOME, INJOB, TRAVTIME,
                     CLAIM_TOT) %>%
  step_impute_mode(JOB) %>%
  step_novel(all_nominal_predictors(), -all_outcomes()) %>%
  step_dummy(all_nominal_predictors(), -all_outcomes()) %>%
  step_zv(all_numeric_predictors(), -all_outcomes()) %>%
  step_normalize(all_numeric_predictors())
prepped_data3<-
  auto_recipe3 %>%
  prep() %>%
  juice()
glimpse(prepped_data3)
logreg_model <- logistic_reg() %>%
                set_engine('glm') %>%
                set_mode("classification")
```

```
logreg_model
library(ranger)

rf_model <-
  rand_forest() %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification")

rf_model

library(xgboost)

xgb_model <-
  boost_tree() %>%
  set_engine("xgboost") %>%
  set_mode("classification")

xgb_model
lm_model <- # your model specification
  linear_reg() %>%  # model type
  set_engine(engine = "lm") %>%  # model engine
  set_mode("regression") # model mode

# Show your model specification
lm_model
glm_model <-
  linear_reg() %>%
  set_engine(engine = "glm") %>%
  set_mode("regression")

glm_model
log_wflow <-
  workflow() %>%
  add_model(logreg_model) %>%
  add_recipe(auto_recipe)

log_wflow


rf_wflow <-
 workflow() %>%
 add_recipe(auto_recipe2) %>%
 add_model(rf_model)

rf_wflow

xgb_wflow <-
 workflow() %>%
 add_recipe(auto_recipe) %>%
 add_model(xgb_model)

xgb_wflow
lm_wflow <-
 workflow() %>%
 add_recipe(auto_recipe3) %>%
 add_model(lm_model)
```

```
lm_wflow

glm_wflow <-
 workflow() %>%
 add_recipe(auto_recipe3) %>%
 add_model(glm_model)

glm_wflow

log_res <-
  log_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(
      recall, precision, f_meas,
      accuracy, kap,
      roc_auc, sens, spec),
    control = control_resamples(
      save_pred = TRUE)
    )
log_res %>% collect_metrics(summarize = T)
log_res %>% collect_metrics(summarize = F)
log_pred <-
  log_res %>%
  collect_predictions()
log_pred %>%
  conf_mat(TARGET_FLAG, .pred_class)
log_pred %>%
  conf_mat(TARGET_FLAG, .pred_class) %>%
  autoplot(type = "heatmap")

log_pred %>%
  group_by(id) %>% # id contains our folds
  roc_curve(TARGET_FLAG, .pred_1) %>%
  autoplot()

log_pred %>%
  ggplot() +
  geom_density(aes(x = .pred_1,
                   fill = TARGET_FLAG),
               alpha = 0.5)

rf_res <-
  rf_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(
      recall, precision, f_meas,
      accuracy, kap,
      roc_auc, sens, spec),
    control = control_resamples(save_pred = TRUE)
    )

rf_res %>%  collect_metrics(summarize = TRUE)
```

```r
xgb_res <-
  xgb_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(
      recall, precision, f_meas,
      accuracy, kap,
      roc_auc, sens, spec),
    control = control_resamples(save_pred = TRUE)
    )

xgb_res %>% collect_metrics(summarize = TRUE)
cv_folds2 <-
  vfold_cv(train_data,
           v = 5,
           strata = TARGET_AMT,
           breaks = 5)

cv_folds2
lm_wflow_eval <-
  lm_wflow %>%
  fit_resamples(
    TARGET_AMT ~ .,
    resamples = cv_folds
    )

lm_wflow_eval%>%
    collect_metrics()
glm_wflow_eval <-
  glm_wflow %>%
  fit_resamples(
    TARGET_AMT ~ .,
    resamples = cv_folds
    )

glm_wflow_eval%>%
    collect_metrics()
log_metrics <-
  log_res %>%
  collect_metrics(summarise = TRUE) %>%
  mutate(model = "Logistic Regression") # add the name of the model to every row

rf_metrics <-
  rf_res %>%
  collect_metrics(summarise = TRUE) %>%
  mutate(model = "Random Forest")

xgb_metrics <-
  xgb_res %>%
  collect_metrics(summarise = TRUE) %>%
  mutate(model = "XGBoost")

# create dataframe with all models
model_compare <- bind_rows(
                          log_metrics,
                           rf_metrics,
```

```
                          xgb_metrics
                          )

# change data structure
model_comp <-
  model_compare %>%
  select(model, .metric, mean, std_err) %>%
  pivot_wider(names_from = .metric, values_from = c(mean, std_err))

# show mean F1-Score for every model
model_comp %>%
  arrange(mean_f_meas) %>%
  mutate(model = fct_reorder(model, mean_f_meas)) %>% # order results
  ggplot(aes(model, mean_f_meas, fill=model)) +
  geom_col() +
  coord_flip() +
  scale_fill_brewer(palette = "Blues") +
   geom_text(
     size = 3,
     aes(label = round(mean_f_meas, 2), y = mean_f_meas + 0.08),
     vjust = 1
  )
# show mean area under the curve (auc) per model
model_comp %>%
  arrange(mean_roc_auc) %>%
  mutate(model = fct_reorder(model, mean_roc_auc)) %>%
  ggplot(aes(model, mean_roc_auc, fill=model)) +
  geom_col() +
  coord_flip() +
  scale_fill_brewer(palette = "Blues") +
     geom_text(
     size = 3,
     aes(label = round(mean_roc_auc, 2), y = mean_roc_auc + 0.08),
     vjust = 1
  )
model_comp %>% slice_max(mean_f_meas)
```