Project 5

John Podias

2022-12-16

Objectives:

1. To fit and compare a traditional binary classification algorithm to a more modern binary classification model where they can both classify whether or not a given car insurance policy holder at a given point in time will file a claim within the next 6 months (model interpretation is not major concern).
    1. In doing this, we hope to find a model that can give reasonable precision and recall for an insurance company to use or at least provide a baseline for them to work from
2. In addition, we want to attempt to build these models using an imbalanced data set, which mirrors real world situations, and then test a solution to see if we can make any improvements to our base models.
    1. A priority of the business would be to catch more positives cases rather than missing out. They are willing to hire more resources to review the large number of positive cases that get flagged.
3. Lastly, I hope to use this analysis as a practice/learning opportunity and try to implement methods I've never used before to develop as a data scientist.

Data set Description: It is a collection of information about different car insurance policies at a given point in time. No further information, such as source or valuation date, was provided. Variables descriptions can be found below and it is sourced from the Kaggle link also provided below. We use the Kaggle "train" set as our main data set before the train/test split and rename it `policy_data`:

https://www.kaggle.com/datasets/ifteshanajnin/carinsuranceclaimprediction-classification?resource=download&select=train.csv

Dimensions:

- 58,592 rows (each row represents a policy)
- 43 predictors that describe the policy (mix of numeric/int and factor variables)
    - Includes policy information such a policy ID, policyholder age, city, population density of policyholder city, and many other data points on the policyholder's car
- 1 response variable (1 = policyholder will file a claim within 6 months, 0 = policyholder will not file a claim within 6 months)

Variable	Description
policy_id	Unique identifier of the policyholder
policy_tenure	Time period of the policy
age_of_car	Normalized age of the car in years
age_of_policyholder	Normalized age of policyholder in years
area_cluster	Area cluster of the policyholder
population density	Population density of the city (Policyholder City)
make	Encoded Manufacturer/company of the car
segment	Segment of the car (A/ B1/ B2/ C1/ C2)
model	Encoded name of the car
fuel_type	Type of fuel used by the car
max_torque	Maximum Torque generated by the car (Nm@rpm)
max_power	Maximum Power generated by the car (bhp@rpm)
engine_type	Type of engine used in the car
airbags	Number of airbags installed in the car
is_esc	Boolean flag indicating whether Electronic Stability Control (ESC) is present in the car or not.
is_adjustable_steering	Boolean flag indicating whether the steering wheel of the car is adjustable or not.
is_tpms	Boolean flag indicating whether Tyre Pressure Monitoring System (TPMS) is present in the car or not.
is_parking_sensors	Boolean flag indicating whether parking sensors are present in the car or not.
is_parking_camera	Boolean flag indicating whether the parking camera is present in the car or not.
rear_brakes_type	Type of brakes used in the rear of the car
displacement	Engine displacement of the car (cc)
cylinder	Number of cylinders present in the engine of the car
transmission_type	Transmission type of the car
gear_box	Number of gears in the car
steering_type	Type of the power steering present in the car
turning_radius	The space a vehicle needs to make a certain turn (Meters)
length	Length of the car (Millimetre)
width	Width of the car (Millimetre)
height	Height of the car (Millimetre)
gross_weight	The maximum allowable weight of the fully-loaded car, including passengers, cargo and equipment (Kg)
is_front_fog_lights	Boolean flag indicating whether front fog lights are available in the car or not.
is_rear_window_wiper	Boolean flag indicating whether the rear window wiper is available in the car or not.
is_rear_window_washer	Boolean flag indicating whether the rear window washer is available in the car or not.
is_rear_window_defogger	Boolean flag indicating whether rear window defogger is available in the car or not.
is_brake_assist	Boolean flag indicating whether the brake assistance feature is available in the car or not.
is_power_door_lock	Boolean flag indicating whether a power door lock is available in the car or not.
is_central_locking	Boolean flag indicating whether the central locking feature is available in the car or not.
is_power_steering	Boolean flag indicating whether power steering is available in the car or not.
is_driver_seat_height_adjustable	Boolean flag indicating whether the height of the driver seat is adjustable or not.
is_day_night_rear_view_mirror	Boolean flag indicating whether day & night rearview mirror is present in the car or not.
is_ecw	Boolean flag indicating whether Engine Check Warning (ECW) is available in the car or not.
is_speed_alert	Boolean flag indicating whether the speed alert system is available in the car or not.
ncap_rating	Safety rating given by NCAP (out of 5)
is_claim	Outcome: Boolean flag indicating whether the policyholder file a claim in the next 6 months or not.

Import packages and data:

EDA:

We start by investigating the data:

- We have 43 predictors and 1 target variable.
  - There is a mix of numeric/int and factor variables with 2 or more levels.
- We also see a highly skewed target variable distribution of 94% of negatives and 6% positives (54,844 vs. 3,748)
- No duplicate records or NA's are present
- We visualized distributions/relationships of each variable, but will not show that analysis here since the data is very high dimensional

```
str(policy_data)
```

```
## 'data.frame':    58592 obs. of  44 variables:
##  $ policy_id                         : Factor w/ 58592 levels "ID00001","ID00002",..: 1 2 3 4 5 6 7 8 9
##  $ policy_tenure                     : num  0.516 0.673 0.841 0.9 0.596 ...
##  $ age_of_car                        : num  0.05 0.02 0.02 0.11 0.11 0.07 0.16 0.14 0.07 0.04 ...
##  $ age_of_policyholder               : num  0.644 0.375 0.385 0.433 0.635 ...
##  $ area_cluster                      : Factor w/ 22 levels "C1","C10","C11",..: 1 12 16 17 18 19 20 21
##  $ population_density                : int  4990 27003 4076 21622 34738 13051 6112 8794 6112 17804 ...
##  $ make                              : int  1 1 1 1 2 3 4 1 3 1 ...
##  $ segment                           : Factor w/ 6 levels "A","B1","B2",..: 1 1 1 4 1 5 3 3 5 3 ...
##  $ model                             : Factor w/ 11 levels "M1","M10","M11",..: 1 1 1 4 5 6 7 8 6 9 ..
##  $ fuel_type                         : Factor w/ 3 levels "CNG","Diesel",..: 1 1 1 3 3 2 2 3 2 3 ...
##  $ max_torque                        : Factor w/ 9 levels "113Nm@4400rpm",..: 6 6 6 1 9 5 4 1 5 1 ...
##  $ max_power                         : Factor w/ 9 levels "113.45bhp@4000rpm",..: 3 3 3 7 6 1 8 7 1 7
##  $ engine_type                       : Factor w/ 11 levels "1.0 SCe","1.2 L K Series Engine",..: 7 7 7
##  $ airbags                           : int  2 2 2 2 2 6 2 2 6 6 ...
##  $ is_esc                            : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 1 2 2 ...
##  $ is_adjustable_steering            : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 2 2 2 2 ...
##  $ is_tpms                           : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 2 1 ...
##  $ is_parking_sensors                : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 2 2 2 2 ...
##  $ is_parking_camera                 : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 1 1 2 2 ...
##  $ rear_brakes_type                  : Factor w/ 2 levels "Disc","Drum": 2 2 2 2 2 1 2 2 1 2 ...
##  $ displacement                      : int  796 796 796 1197 999 1493 1497 1197 1493 1197 ...
##  $ cylinder                          : int  3 3 3 4 3 4 4 4 4 4 ...
##  $ transmission_type                 : Factor w/ 2 levels "Automatic","Manual": 2 2 2 1 1 1 2 2 1 1 ..
##  $ gear_box                          : int  5 5 5 5 5 6 5 5 6 5 ...
##  $ steering_type                     : Factor w/ 3 levels "Electric","Manual",..: 3 3 3 1 1 3 1 1 3 1
##  $ turning_radius                    : num  4.6 4.6 4.6 4.8 5 5.2 5 4.8 5.2 4.85 ...
##  $ length                            : int  3445 3445 3445 3995 3731 4300 3990 3845 4300 3990 ...
##  $ width                             : int  1515 1515 1515 1735 1579 1790 1755 1735 1790 1745 ...
##  $ height                            : int  1475 1475 1475 1515 1490 1635 1523 1530 1635 1500 ...
##  $ gross_weight                      : int  1185 1185 1185 1335 1155 1720 1490 1335 1720 1410 ...
##  $ is_front_fog_lights               : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 2 2 2 ...
##  $ is_rear_window_wiper              : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 2 2 ...
##  $ is_rear_window_washer             : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 2 2 ...
##  $ is_rear_window_defogger           : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 1 2 2 ...
##  $ is_brake_assist                   : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 2 2 2 ...
##  $ is_power_door_locks               : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 2 2 2 2 ...
##  $ is_central_locking                : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 2 2 2 2 ...
##  $ is_power_steering                 : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ is_driver_seat_height_adjustable: Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 2 2 2 ...
##  $ is_day_night_rear_view_mirror   : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 1 2 1 2 ...
```

```
##  $ is_ecw                          : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 2 2 2 2 ...
##  $ is_speed_alert                  : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ncap_rating                     : int  0 0 0 2 2 3 5 2 3 0 ...
##  $ is_claim                        : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
describe(policy_data)
```

```
##                                  vars     n     mean       sd   median
## policy_id*                          1 58592 29296.50 16914.20 29296.50
## policy_tenure                       2 58592     0.61     0.41     0.57
## age_of_car                          3 58592     0.07     0.06     0.06
## age_of_policyholder                 4 58592     0.47     0.12     0.45
## area_cluster*                       5 58592    14.04     6.80    16.00
## population_density                  6 58592 18826.86 17660.17  8794.00
## make                                7 58592     1.76     1.14     1.00
## segment*                            8 58592     2.94     1.57     3.00
## model*                              9 58592     5.66     3.20     6.00
## fuel_type*                         10 58592     2.00     0.84     2.00
## max_torque*                        11 58592     4.29     2.44     5.00
## max_power*                         12 58592     4.32     2.57     3.00
## engine_type*                       13 58592     6.74     2.94     7.00
## airbags                            14 58592     3.14     1.83     2.00
## is_esc*                            15 58592     1.31     0.46     1.00
## is_adjustable_steering*            16 58592     1.61     0.49     2.00
## is_tpms*                           17 58592     1.24     0.43     1.00
## is_parking_sensors*                18 58592     1.96     0.20     2.00
## is_parking_camera*                 19 58592     1.39     0.49     1.00
## rear_brakes_type*                  20 58592     1.76     0.43     2.00
## displacement                       21 58592  1162.36   266.30  1197.00
## cylinder                           22 58592     3.63     0.48     4.00
## transmission_type*                 23 58592     1.65     0.48     2.00
## gear_box                           24 58592     5.25     0.43     5.00
## steering_type*                     25 58592     2.16     0.98     3.00
## turning_radius                     26 58592     4.85     0.23     4.80
## length                             27 58592  3850.48   311.46  3845.00
## width                              28 58592  1672.23   112.09  1735.00
## height                             29 58592  1553.34    79.62  1530.00
## gross_weight                       30 58592  1385.28   212.42  1335.00
## is_front_fog_lights*               31 58592     1.58     0.49     2.00
## is_rear_window_wiper*              32 58592     1.29     0.45     1.00
## is_rear_window_washer*             33 58592     1.29     0.45     1.00
## is_rear_window_defogger*           34 58592     1.35     0.48     1.00
## is_brake_assist*                   35 58592     1.55     0.50     2.00
## is_power_door_locks*               36 58592     1.72     0.45     2.00
## is_central_locking*                37 58592     1.72     0.45     2.00
## is_power_steering*                 38 58592     1.98     0.14     2.00
## is_driver_seat_height_adjustable*  39 58592     1.59     0.49     2.00
## is_day_night_rear_view_mirror*     40 58592     1.38     0.49     1.00
## is_ecw*                            41 58592     1.72     0.45     2.00
## is_speed_alert*                    42 58592     1.99     0.08     2.00
## ncap_rating                        43 58592     1.76     1.39     2.00
## is_claim                           44 58592     0.06     0.24     0.00
##                                   trimmed      mad     min     max    range
## policy_id*                       29296.50 21717.12    1.00 58592.0 58591.00
## policy_tenure                       0.61     0.64    0.00     1.4     1.39
```

```
##                                    0.06     0.06    0.00     1.0     1.00
## age_of_policyholder                0.46     0.13    0.29     1.0     0.71
## area_cluster*                     14.62     7.41    1.00    22.0    21.00
## population_density             15819.46  6994.91  290.00 73430.0 73140.00
## make                               1.57     0.00    1.00     5.0     4.00
## segment*                           2.90     2.97    1.00     6.0     5.00
## model*                             5.65     2.97    1.00    11.0    10.00
## fuel_type*                         2.00     1.48    1.00     3.0     2.00
## max_torque*                        4.23     1.48    1.00     9.0     8.00
## max_power*                         4.27     2.97    1.00     9.0     8.00
## engine_type*                       6.87     4.45    1.00    11.0    10.00
## airbags                            2.95     0.00    1.00     6.0     5.00
## is_esc*                            1.27     0.00    1.00     2.0     1.00
## is_adjustable_steering*            1.63     0.00    1.00     2.0     1.00
## is_tpms*                           1.17     0.00    1.00     2.0     1.00
## is_parking_sensors*                2.00     0.00    1.00     2.0     1.00
## is_parking_camera*                 1.36     0.00    1.00     2.0     1.00
## rear_brakes_type*                  1.83     0.00    1.00     2.0     1.00
## displacement                    1166.46   438.85  796.00  1498.0   702.00
## cylinder                           3.66     0.00    3.00     4.0     1.00
## transmission_type*                 1.69     0.00    1.00     2.0     1.00
## gear_box                           5.18     0.00    5.00     6.0     1.00
## steering_type*                     2.21     0.00    1.00     3.0     2.00
## turning_radius                     4.84     0.30    4.50     5.2     0.70
## length                          3844.97   281.69 3445.00  4300.0   855.00
## width                           1678.04    81.54 1475.00  1811.0   336.00
## height                          1544.46    81.54 1475.00  1825.0   350.00
## gross_weight                    1376.03   222.39 1051.00  1720.0   669.00
## is_front_fog_lights*               1.60     0.00    1.00     2.0     1.00
## is_rear_window_wiper*              1.24     0.00    1.00     2.0     1.00
## is_rear_window_washer*             1.24     0.00    1.00     2.0     1.00
## is_rear_window_defogger*           1.31     0.00    1.00     2.0     1.00
## is_brake_assist*                   1.56     0.00    1.00     2.0     1.00
## is_power_door_locks*               1.78     0.00    1.00     2.0     1.00
## is_central_locking*                1.78     0.00    1.00     2.0     1.00
## is_power_steering*                 2.00     0.00    1.00     2.0     1.00
## is_driver_seat_height_adjustable*  1.61     0.00    1.00     2.0     1.00
## is_day_night_rear_view_mirror*     1.35     0.00    1.00     2.0     1.00
## is_ecw*                            1.78     0.00    1.00     2.0     1.00
## is_speed_alert*                    2.00     0.00    1.00     2.0     1.00
## ncap_rating                        1.70     1.48    0.00     5.0     5.00
## is_claim                           0.00     0.00    0.00     1.0     1.00
##                                    skew kurtosis    se
## policy_id*                         0.00    -1.20 69.88
## policy_tenure                      0.05    -1.50  0.00
## age_of_car                         1.09     5.30  0.00
## age_of_policyholder                0.64    -0.16  0.00
## area_cluster*                     -0.54    -1.15  0.03
## population_density                 1.67     2.59 72.96
## make                               1.20     0.34  0.00
## segment*                           0.14    -1.27  0.01
## model*                            -0.33    -1.16  0.01
## fuel_type*                        -0.01    -1.57  0.00
## max_torque*                       -0.20    -1.15  0.01
```

```
## max_power*                           0.10    -1.42  0.01
## engine_type*                        -0.19    -1.15  0.01
## airbags                              0.91    -1.14  0.01
## is_esc*                              0.80    -1.36  0.00
## is_adjustable_steering*             -0.44    -1.81  0.00
## is_tpms*                             1.22    -0.51  0.00
## is_parking_sensors*                 -4.66    19.73  0.00
## is_parking_camera*                   0.45    -1.80  0.00
## rear_brakes_type*                   -1.22    -0.51  0.00
## displacement                        -0.11    -1.34  1.10
## cylinder                            -0.53    -1.72  0.00
## transmission_type*                  -0.64    -1.59  0.00
## gear_box                             1.18    -0.60  0.00
## steering_type*                      -0.33    -1.87  0.00
## turning_radius                       0.42    -1.18  0.00
## length                               0.15    -1.21  1.29
## width                               -0.49    -1.44  0.46
## height                               1.04     0.73  0.33
## gross_weight                         0.55    -1.02  0.88
## is_front_fog_lights*                -0.32    -1.90  0.00
## is_rear_window_wiper*                0.93    -1.14  0.00
## is_rear_window_washer*               0.93    -1.14  0.00
## is_rear_window_defogger*             0.63    -1.61  0.00
## is_brake_assist*                    -0.20    -1.96  0.00
## is_power_door_locks*                -1.00    -0.99  0.00
## is_central_locking*                 -1.00    -0.99  0.00
## is_power_steering*                  -6.74    43.48  0.00
## is_driver_seat_height_adjustable*   -0.35    -1.88  0.00
## is_day_night_rear_view_mirror*       0.49    -1.76  0.00
## is_ecw*                             -1.00    -0.99  0.00
## is_speed_alert*                    -12.59   156.41  0.00
## ncap_rating                          0.09    -0.78  0.01
## is_claim                             3.56    10.70  0.00
```

```
summary(policy_data)
```

```
##    policy_id     policy_tenure        age_of_car      age_of_policyholder
##  ID00001:    1   Min.   :0.002735   Min.   :0.00000   Min.   :0.2885
##  ID00002:    1   1st Qu.:0.210250   1st Qu.:0.02000   1st Qu.:0.3654
##  ID00003:    1   Median :0.573792   Median :0.06000   Median :0.4519
##  ID00004:    1   Mean   :0.611246   Mean   :0.06942   Mean   :0.4694
##  ID00005:    1   3rd Qu.:1.039104   3rd Qu.:0.11000   3rd Qu.:0.5481
##  ID00006:    1   Max.   :1.396641   Max.   :1.00000   Max.   :1.0000
##  (Other):58586
##   area_cluster   population_density      make          segment
##  C8     :13654   Min.   :  290      Min.   :1.000   A      :17321
##  C2     : 7342   1st Qu.: 6112      1st Qu.:1.000   B1     : 4173
##  C5     : 6979   Median : 8794      Median :1.000   B2     :18314
##  C3     : 6101   Mean   :18827      Mean   :1.764   C1     : 3557
##  C14    : 3660   3rd Qu.:27003      3rd Qu.:3.000   C2     :14018
##  C13    : 3423   Max.   :73430      Max.   :5.000   Utility: 1209
##  (Other):17433
##      model        fuel_type             max_torque
##  M1     :14948   CNG   :20330   113Nm@4400rpm :17796
##  M4     :14018   Diesel:17730   60Nm@3500rpm  :14948
```

```
##  M6     :13776   Petrol:20532   250Nm@2750rpm :14018
##  M8     : 4173                  82.1Nm@3400rpm: 4173
##  M7     : 2940                  91Nm@4250rpm  : 2373
##  M3     : 2373                  200Nm@1750rpm : 2114
##  (Other): 6364                  (Other)       : 3170
##              max_power                    engine_type       airbags
##  88.50bhp@6000rpm :17796   F8D Petrol Engine    :14948   Min.   :1.000
##  40.36bhp@6000rpm :14948   1.5 L U2 CRDi        :14018   1st Qu.:2.000
##  113.45bhp@4000rpm:14018   K Series Dual jet    :13776   Median :2.000
##  55.92bhp@5300rpm : 4173   K10C                 : 4173   Mean   :3.137
##  67.06bhp@5500rpm : 2373   1.2 L K Series Engine: 2940   3rd Qu.:6.000
##  97.89bhp@3600rpm : 2114   1.0 SCe              : 2373   Max.   :6.000
##  (Other)          : 3170   (Other)              : 6364
##  is_esc      is_adjustable_steering is_tpms     is_parking_sensors
##  No :40191   No :23066              No :44574   No : 2373
##  Yes:18401   Yes:35526              Yes:14018   Yes:56219
##
##
##
##
##
##  is_parking_camera rear_brakes_type  displacement     cylinder
##  No :35704         Disc:14018       Min.   : 796   Min.   :3.000
##  Yes:22888         Drum:44574       1st Qu.: 796   1st Qu.:3.000
##                                     Median :1197   Median :4.000
##                                     Mean   :1162   Mean   :3.627
##                                     3rd Qu.:1493   3rd Qu.:4.000
##                                     Max.   :1498   Max.   :4.000
##
##  transmission_type    gear_box      steering_type   turning_radius
##  Automatic:20411   Min.   :5.000   Electric:23881   Min.   :4.500
##  Manual   :38181   1st Qu.:5.000   Manual  : 1209   1st Qu.:4.600
##                    Median :5.000   Power   :33502   Median :4.800
##                    Mean   :5.245                    Mean   :4.853
##                    3rd Qu.:5.000                    3rd Qu.:5.000
##                    Max.   :6.000                    Max.   :5.200
##
##      length         width          height      gross_weight
##  Min.   :3445   Min.   :1475   Min.   :1475   Min.   :1051
##  1st Qu.:3445   1st Qu.:1515   1st Qu.:1475   1st Qu.:1185
##  Median :3845   Median :1735   Median :1530   Median :1335
##  Mean   :3850   Mean   :1672   Mean   :1553   Mean   :1385
##  3rd Qu.:3995   3rd Qu.:1755   3rd Qu.:1635   3rd Qu.:1510
##  Max.   :4300   Max.   :1811   Max.   :1825   Max.   :1720
##
##  is_front_fog_lights is_rear_window_wiper is_rear_window_washer
##  No :24664           No :41634            No :41634
##  Yes:33928           Yes:16958            Yes:16958
##
##
##
##
##
##  is_rear_window_defogger is_brake_assist is_power_door_locks is_central_locking
```

```
##  No :38077               No :26415       No :16157           No :16157
##  Yes:20515               Yes:32177       Yes:42435           Yes:42435
##
##
##
##
##
##  is_power_steering is_driver_seat_height_adjustable
##  No : 1209         No :24301
##  Yes:57383         Yes:34291
##
##
##
##
##
##  is_day_night_rear_view_mirror is_ecw      is_speed_alert  ncap_rating
##  No :36309                     No :16157   No :  363      Min.   :0.00
##  Yes:22283                     Yes:42435   Yes:58229      1st Qu.:0.00
##                                                           Median :2.00
##                                                           Mean   :1.76
##                                                           3rd Qu.:3.00
##                                                           Max.   :5.00
##
##     is_claim
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.06397
##  3rd Qu.:0.00000
##  Max.   :1.00000
##
```

```
dim(policy_data)
```

```
## [1] 58592    44
```

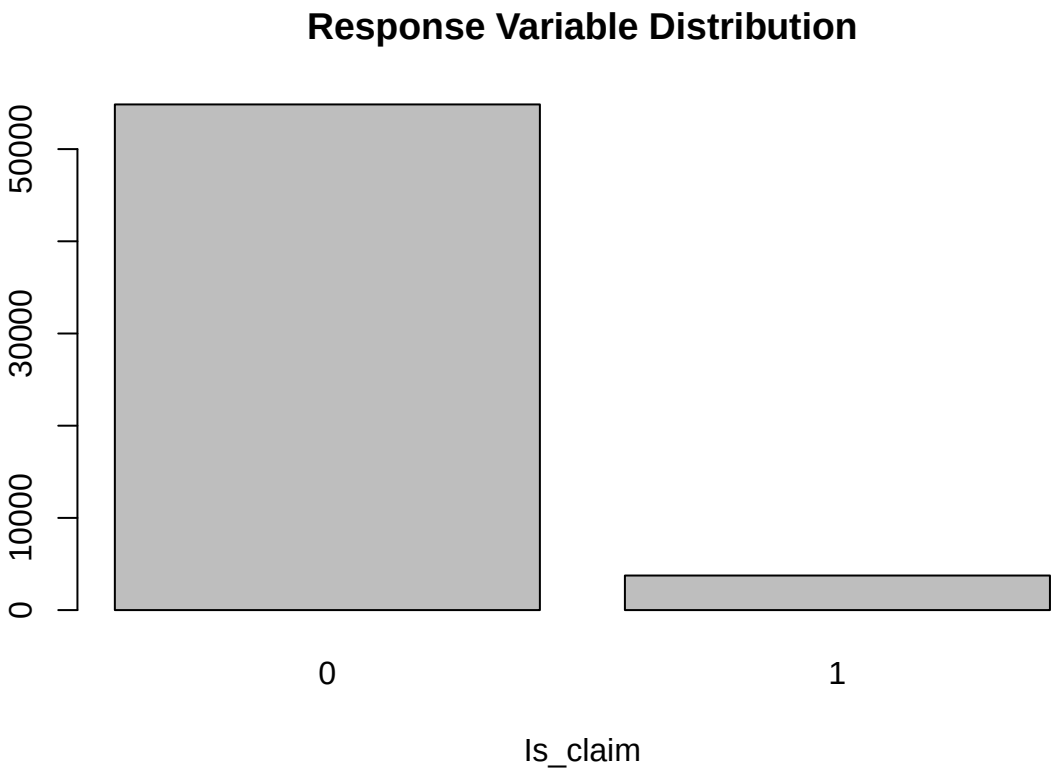
```
#Target variable count distribution
table(policy_data$is_claim)
```

```
##
##     0     1
## 54844  3748
```

```
#Target variable Percent distribution
prop.table(table(policy_data$is_claim))
```

```
##
##          0          1
## 0.93603222 0.06396778
```

```
barplot(table(policy_data$is_claim),main="Response Variable Distribution", xlab="Is_claim")
```




```
#We dont see any NA's
sum(is.na(policy_data))
```

```
## [1] 0
```

```
#The policy ID unique list is the same length as the number of rows so we do not have duplicates
length(unique(policy_data$policy_id))
```

```
## [1] 58592
```

Data Cleaning / Feature Engineering:

We create a new data set with 42 predictors and 1 target variable using the methods below:

- We start by converting `make` and `is_claim` into factors since the raw data set treated them as numbers. Because they contain multiple numbers in text format, we will use `max_torque` and `max_power` to create 4 new variables (`max_torque_Nm, max_torque_RPM, max_power_BHP, max_power_RPM`)
- In addition, we will create dummy variables only for factors with 2 or 3 levels, convert the new dummies to factors, and remove factor variables with more than 3 levels. This is an effort to reduce complexity/multicollinearity within our models. We do not create a dummy variable for the first categorical level for each one and will treat these as the baseline to avoid multicollinearity.
  - In our initial exploration of fitting our Logistic Regression and Random Forest models that are explained later on, we had created dummy variables for all columns regardless of the numbers of levels. This about doubled the number of variables and caused very low scores (precision/recall/F1). In the interest of an easy to follow presentation, we are not showing this early exploration step and decide to exclude the variables that contain high cardinality at the start of this analysis.

```
#The raw data has these two variables as numeric so we will convert them to factors
policy_data$is_claim<-as.factor(policy_data$is_claim)
policy_data$make<-as.factor(policy_data$make)

#Create 4 new numeric columns from max_power and max_torque
```

```
policy_data$max_torque_Nm<-sub("@.*","",policy_data$max_torque)
policy_data$max_torque_Nm<-str_replace(policy_data$max_torque_Nm,"Nm","")
policy_data$max_torque_Nm<-as.numeric(policy_data$max_torque_Nm)
policy_data$max_torque_RPM<-sub(".*@","",policy_data$max_torque)
policy_data$max_torque_RPM<-str_replace(policy_data$max_torque_RPM,"rpm","")
policy_data$max_torque_RPM<-as.numeric(policy_data$max_torque_RPM)
policy_data$max_power_BHP<-sub("@.*","",policy_data$max_power)
policy_data$max_power_BHP<-str_replace(policy_data$max_power_BHP,"bhp","")
policy_data$max_power_BHP<-as.numeric(policy_data$max_power_BHP)
policy_data$max_power_RPM<-sub(".*@","",policy_data$max_power)
policy_data$max_power_RPM<-str_replace(policy_data$max_power_RPM,"rpm","")
policy_data$max_power_RPM<-as.numeric(policy_data$max_power_RPM)

#We will only create dummy variables for factors with 3 levels:
policy_data<-dummy_cols(policy_data,remove_first_dummy=TRUE,select_columns = c("fuel_type","steering_ty

#Convert dummy variables to factors
policy_data[,c(49:52)] <- lapply(policy_data[,c(49:52)],factor)

#Get rid of Unnecessary ID columns and other factor variables with high cardinality that add complexity
policy_data<-policy_data %>% select(-c(policy_id,max_torque,max_power,area_cluster, make,segment,model,

#New Data set is doum to 42 variables
str(policy_data)
```

```
## 'data.frame':    58592 obs. of  42 variables:
##  $ policy_tenure                  : num  0.516 0.673 0.841 0.9 0.596 ...
##  $ age_of_car                     : num  0.05 0.02 0.02 0.11 0.11 0.07 0.16 0.14 0.07 0.04 ...
##  $ age_of_policyholder            : num  0.644 0.375 0.385 0.433 0.635 ...
##  $ population_density             : int  4990 27003 4076 21622 34738 13051 6112 8794 6112 17804 ...
##  $ airbags                        : int  2 2 2 2 2 6 2 2 6 6 ...
##  $ is_esc                         : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 1 2 2 ...
##  $ is_adjustable_steering         : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 2 2 2 2 ...
##  $ is_tpms                        : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 2 1 ...
##  $ is_parking_sensors             : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 2 2 2 2 ...
##  $ is_parking_camera              : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 1 1 2 2 ...
##  $ rear_brakes_type               : Factor w/ 2 levels "Disc","Drum": 2 2 2 2 2 1 2 2 1 2 ...
##  $ displacement                   : int  796 796 796 1197 999 1493 1497 1197 1493 1197 ...
##  $ cylinder                       : int  3 3 3 4 3 4 4 4 4 4 ...
##  $ transmission_type              : Factor w/ 2 levels "Automatic","Manual": 2 2 2 1 1 1 2 2 1 1 ..
##  $ gear_box                       : int  5 5 5 5 5 6 5 5 6 5 ...
##  $ turning_radius                 : num  4.6 4.6 4.6 4.8 5 5.2 5 4.8 5.2 4.85 ...
##  $ length                         : int  3445 3445 3445 3995 3731 4300 3990 3845 4300 3990 ...
##  $ width                          : int  1515 1515 1515 1735 1579 1790 1755 1735 1790 1745 ...
##  $ height                         : int  1475 1475 1475 1515 1490 1635 1523 1530 1635 1500 ...
##  $ gross_weight                   : int  1185 1185 1185 1335 1155 1720 1490 1335 1720 1410 ...
##  $ is_front_fog_lights            : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 2 2 2 ...
##  $ is_rear_window_wiper           : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 2 2 ...
##  $ is_rear_window_washer          : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 2 2 ...
##  $ is_rear_window_defogger        : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 1 2 2 ...
##  $ is_brake_assist                : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 2 2 2 ...
##  $ is_power_door_locks            : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 2 2 2 2 ...
##  $ is_central_locking             : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 2 2 2 2 ...
##  $ is_power_steering              : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
```

```
##  $ is_driver_seat_height_adjustable: Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 2 2 2 ...
##  $ is_day_night_rear_view_mirror   : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 1 2 1 2 ...
##  $ is_ecw                          : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 2 2 2 2 ...
##  $ is_speed_alert                  : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ncap_rating                     : int  0 0 0 2 2 3 5 2 3 0 ...
##  $ is_claim                        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_torque_Nm                   : num  60 60 60 113 91 250 200 113 250 113 ...
##  $ max_torque_RPM                  : num  3500 3500 3500 4400 4250 2750 3000 4400 2750 4400 ...
##  $ max_power_BHP                   : num  40.4 40.4 40.4 88.5 67.1 ...
##  $ max_power_RPM                   : num  6000 6000 6000 6000 5500 4000 4000 6000 4000 6000 ...
##  $ fuel_type_Diesel                : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ fuel_type_Petrol                : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 1 2 1 2 ...
##  $ steering_type_Manual            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ steering_type_Power             : Factor w/ 2 levels "0","1": 2 2 2 1 1 2 1 1 2 1 ...
```

Train/Test Split:

- We use the 90/10 split because of our imbalanced target variable distribution.
  - We experimented with other splits (80/20 and 85/15), but we would like as much training data as possible and we get better metrics with this split. For future work, we could try others.
- We also make sure our test set is representative with an imbalanced target variable distribution that is about the same as the training set

```
#make reproducible
set.seed(1)

#use 90% of original dataset as the training set and 10% as test set
sample <- sample(c(TRUE, FALSE), nrow(policy_data), replace=TRUE, prob=c(0.9,0.1))
train_final<- policy_data[sample, ]
test_final<- policy_data[!sample, ]

#Training target variable count distribution
table(train_final$is_claim)
```

```
##
##     0     1
## 49314  3362
```

```
#Training target variable percent distribution
prop.table(table(train_final$is_claim))
```

```
##
##          0          1
## 0.93617587 0.06382413
```

```
#Test target variable count distribution
table(test_final$is_claim)
```

```
##
##    0    1
## 5530  386
```

```
##Test target variable percent distribution
prop.table(table(test_final$is_claim))
```

```
##
##          0          1
```

```
## 0.93475321 0.06524679
```

Baseline Model Fitting and Comparison:

Next, we fit a Random Forest model (a more modern method) and a Logistic Regression model (a more traditional binary classification method) and compare precision, recall, and F1 score. We ignore the accuracy metric for now since we still have the target variable imbalance.

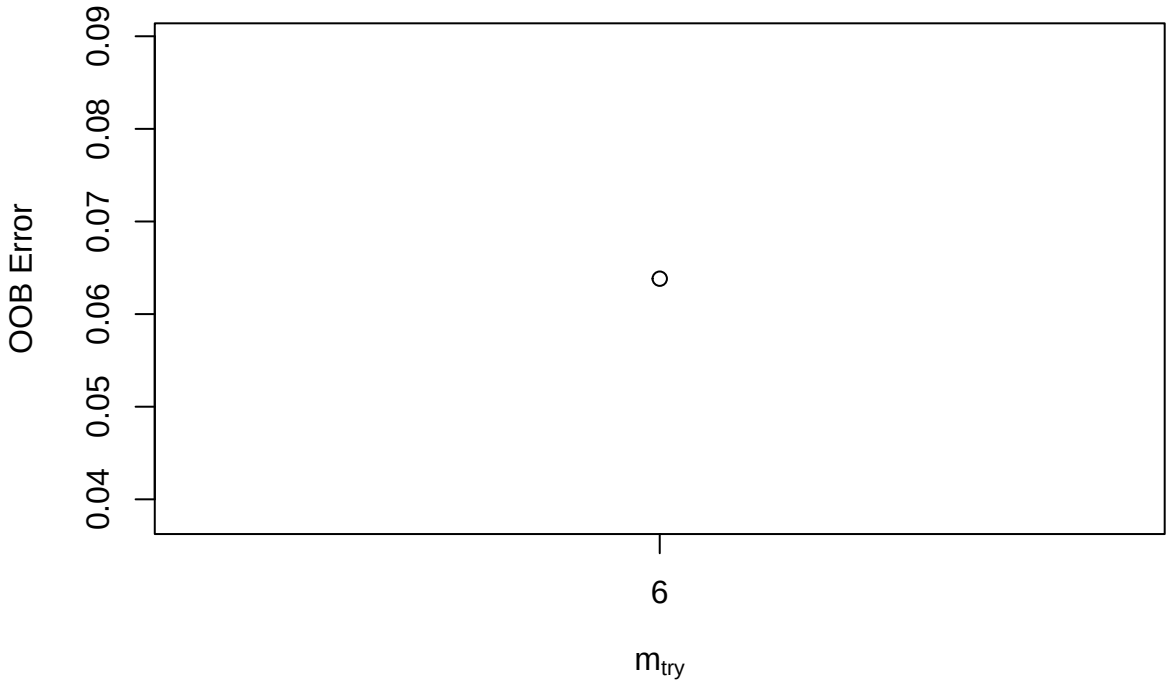
Random Forest Model:

We tune the mtry parameter to get 6 since it has the lowest OOB error and use 500 as the default number of trees even though we also tested with 1000 trees. We fit the model using all variables and we get a very poor model because of the target variable imbalance. We get NA for F1, 0 for recall, and NA for precision. We did look the variable importance and use that information to fit a final second model that is not shown here. It did not really provide any improvment in precision and recall. Further work could be done to tune the model.

```
#Random Forest Model

#Tune for mtry. We go with 500 tree default although we did test other numbers of trees and see that mt
tuneRF(x = train_final[,-34],y = train_final$is_claim,ntreeTry = 500,trace=T, plot= T,improve = .01,step
```

```
## mtry = 6  OOB error = 6.38%
## Searching left ...
## Searching right ...
```




```
##       mtry   OOBError
## 6.OOB    6 0.06382413
```

```
#Fit with tuned mtry parameter and default 500 trees
RF_Model_1<-randomForest(is_claim ~ .,mtry=6,data = train_final)

RF_Model_1
```

```
##
## Call:
##  randomForest(formula = is_claim ~ ., data = train_final, mtry = 6)
```

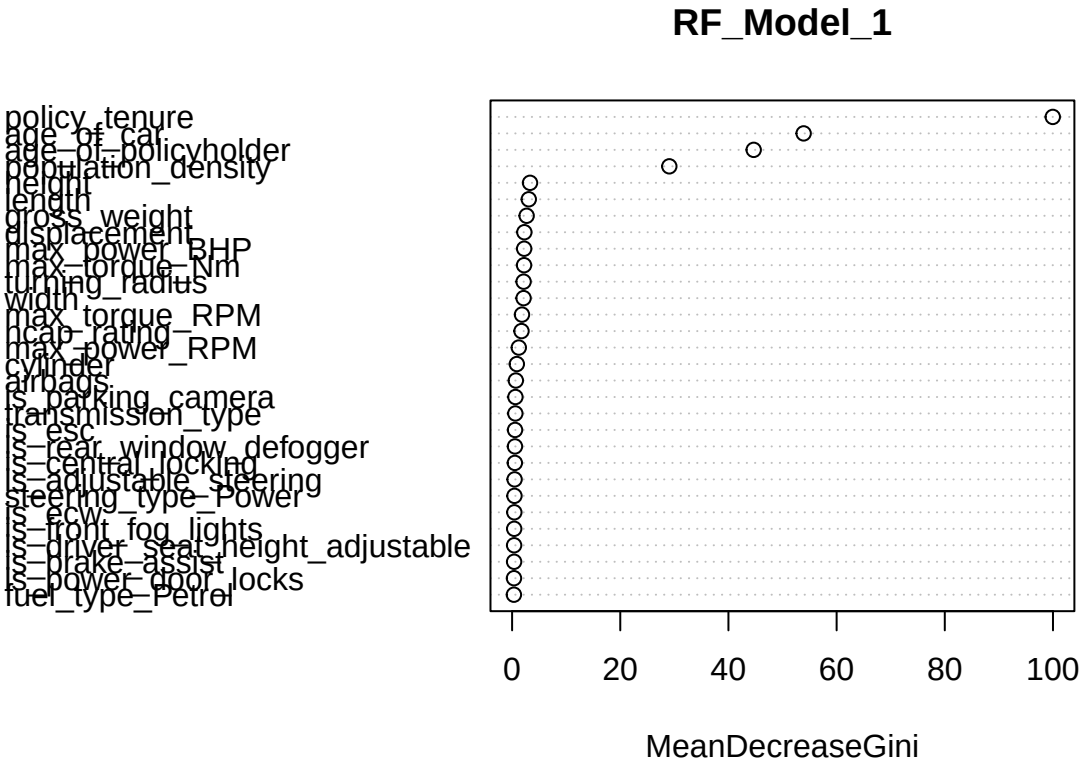
```
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 6.38%
## Confusion matrix:
##       0 1 class.error
## 0 49314 0           0
## 1  3362 0           1
```

```
#Create vector of predictions
RF_predicted_1<-predict(RF_Model_1,test_final[,-34])

#We see this model does not provide good predictive ability looking at precision, recall and F1 score.
confusionMatrix(RF_predicted_1,test_final$is_claim,mode = "everything",positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5530  386
##          1    0    0
##
##                Accuracy : 0.9348
##                  95% CI : (0.9282, 0.9409)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 0.5135
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.93475
##               Precision :      NA
##                  Recall : 0.00000
##                      F1 :      NA
##              Prevalence : 0.06525
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```
#Variable Importance
varImpPlot(RF_Model_1)
```



Logistic Regression Model:

We tune the logistic regression model to get the optimal fit. First, we fit the model with all variables and this gives us a F1 of NA, Recall of 0, and precision of NA To do a simple fitting, we only take significant predictors from the full model and confirm no multicollinearity is present. We see that this still does not provide any improvement. We could tune further if more time is allowed.

- Note we classify positive cases in Logistic Regression as anything with probability greater than 0.5.

```
#First model:

#We fit first model using all variables
mylogit_1 <- glm(is_claim ~ ., data = train_final, family = "binomial")

#VIF indicates error that there is high multicollinearity
#vif(mylogit_1)

#We see lots of issues with NA coefficients
summary(mylogit_1)
```

```
##
## Call:
## glm(formula = is_claim ~ ., family = "binomial", data = train_final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5606  -0.4036  -0.3411  -0.2959   2.8849
##
## Coefficients: (27 not defined because of singularities)
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -2.875e+01  1.706e+01  -1.685   0.0920 .
```

```
## policy_tenure                           8.398e-01  4.716e-02  17.809   <2e-16 ***
## age_of_car                             -3.550e+00  3.808e-01  -9.323   <2e-16 ***
## age_of_policyholder                     2.356e-01  1.450e-01   1.625   0.1041
## population_density                     -2.176e-06  1.080e-06  -2.015   0.0439 *
## airbags                                -6.625e-02  5.414e-02  -1.224   0.2211
## is_escYes                              -1.963e+00  1.139e+00  -1.724   0.0846 .
## is_adjustable_steeringYes              -1.312e+00  9.212e-01  -1.424   0.1545
## is_tpmsYes                             -9.392e-01  5.628e-01  -1.669   0.0952 .
## is_parking_sensorsYes                   3.456e+00  2.032e+00   1.700   0.0891 .
## is_parking_cameraYes                    2.499e-01  2.822e-01   0.886   0.3758
## rear_brakes_typeDrum                           NA         NA      NA       NA
## displacement                           -2.668e-03  1.878e-03  -1.421   0.1554
## cylinder                                1.552e+00  1.017e+00   1.526   0.1270
## transmission_typeManual                -1.759e+00  9.144e-01  -1.924   0.0544 .
## gear_box                                       NA         NA      NA       NA
## turning_radius                          4.680e+00  3.145e+00   1.488   0.1367
## length                                         NA         NA      NA       NA
## width                                          NA         NA      NA       NA
## height                                         NA         NA      NA       NA
## gross_weight                                   NA         NA      NA       NA
## is_front_fog_lightsYes                         NA         NA      NA       NA
## is_rear_window_wiperYes                        NA         NA      NA       NA
## is_rear_window_washerYes                       NA         NA      NA       NA
## is_rear_window_defoggerYes                     NA         NA      NA       NA
## is_brake_assistYes                             NA         NA      NA       NA
## is_power_door_locksYes                         NA         NA      NA       NA
## is_central_lockingYes                          NA         NA      NA       NA
## is_power_steeringYes                           NA         NA      NA       NA
## is_driver_seat_height_adjustableYes            NA         NA      NA       NA
## is_day_night_rear_view_mirrorYes               NA         NA      NA       NA
## is_ecwYes                                      NA         NA      NA       NA
## is_speed_alertYes                              NA         NA      NA       NA
## ncap_rating                                    NA         NA      NA       NA
## max_torque_Nm                                  NA         NA      NA       NA
## max_torque_RPM                                 NA         NA      NA       NA
## max_power_BHP                                  NA         NA      NA       NA
## max_power_RPM                                  NA         NA      NA       NA
## fuel_type_Diesel1                              NA         NA      NA       NA
## fuel_type_Petrol1                              NA         NA      NA       NA
## steering_type_Manual1                          NA         NA      NA       NA
## steering_type_Power1                           NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25007  on 52675  degrees of freedom
## Residual deviance: 24555  on 52661  degrees of freedom
## AIC: 24585
##
## Number of Fisher Scoring iterations: 5
```

```
#Make predictions using first model
Log_predictions_1 <- predict(mylogit_1, test_final[,-34], type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
#If prob exceeds threshold of 0.5, 1 else 0
Log_predictions_1 <- ifelse(Log_predictions_1 > 0.5, 1, 0)
#Convert to factor
Log_predictions_1 <- as.factor(Log_predictions_1)
#Create confusion matrix
confusionMatrix(Log_predictions_1, test_final$is_claim,mode = "everything",positive = "1")
```

```
## Warning in confusionMatrix.default(Log_predictions_1, test_final$is_claim, :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5530  386
##          1    0    0
##
##                Accuracy : 0.9348
##                  95% CI : (0.9282, 0.9409)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 0.5135
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.93475
##               Precision :      NA
##                  Recall : 0.00000
##                      F1 :      NA
##              Prevalence : 0.06525
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```
#Second Model:

#We only take significant predictors right now and create a second model:
mylogit_2 <- glm(is_claim ~ population_density + age_of_policyholder + age_of_car + policy_tenure, data

#All variables are significant
summary(mylogit_2)
```

```
##
## Call:
## glm(formula = is_claim ~ population_density + age_of_policyholder +
```

```
##     age_of_car + policy_tenure, family = "binomial", data = train_final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5373  -0.4052  -0.3417  -0.2966   2.8902
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -3.114e+00  7.974e-02 -39.051   <2e-16 ***
## population_density  -2.319e-06  1.076e-06  -2.154   0.0312 *
## age_of_policyholder  2.598e-01  1.441e-01   1.803   0.0714 .
## age_of_car          -3.341e+00  3.482e-01  -9.594   <2e-16 ***
## policy_tenure        8.504e-01  4.599e-02  18.492   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25007  on 52675  degrees of freedom
## Residual deviance: 24574  on 52671  degrees of freedom
## AIC: 24584
##
## Number of Fisher Scoring iterations: 5
```

```
#VIF indicates no multicollinearity; no values 10 or above
vif(mylogit_2)
```

```
##  population_density age_of_policyholder          age_of_car       policy_tenure
##            1.013157            1.023249            1.045540            1.072836
```

```
#Make predictions using second model
Log_predictions_2 <- predict(mylogit_2, test_final[,-34], type = "response")
#If prob exceeds threshold of 0.5, 1 else 0
Log_predictions_2 <- ifelse(Log_predictions_2 > 0.5, 1, 0)
#Convert to factor
Log_predictions_2 <- as.factor(Log_predictions_2)
#Create confusion matrix
confusionMatrix(Log_predictions_2, test_final$is_claim,mode = "everything",positive = "1")
```

```
## Warning in confusionMatrix.default(Log_predictions_2, test_final$is_claim, :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5530  386
##          1    0    0
##
##                Accuracy : 0.9348
##                  95% CI : (0.9282, 0.9409)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 0.5135
##
##                   Kappa : 0
```

```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.93475
##               Precision :      NA
##                  Recall : 0.00000
##                      F1 :      NA
##              Prevalence : 0.06525
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

Both Random Forest and Logistic Regression provide us with very bad predictive ability and so we need to find a solution to implement.

SMOTE:

We see we are consistently getting 0 or NA metric scores for our initial models. This leads us to believe that it could be a major problem with the data set. Therefore, we will test a method that we have never tried before called SMOTE and see how it could improve our models. Future work could include exploring other sampling methods.

SMOTE (Synthetic Minority Oversampling Technique):

- This is an oversampling technique done on our training set to create and add more synthetic observations (in our case policyholders) of the minority class to create a more balanced distribution of the target variable and hopefully help us train models with better data. Also note:
  - We try SMOTE because of our small amount of data and would rather over sample the minority than lose data by under sampling the majority. It is also a way to prevent over fitting because its generates similar versions of the minority as opposed to just adding duplicates like random oversampling
  - We try different parameters to get a 50/50 split as mentioned in the comments below. This leads us to perc.over=40 and perc.under=1. Future work could include further tuning of this sampling.
    - We do not include our analysis of different SMOTE data sets for simplicity and to prevent long run time when reproducing the file. We only include our final SMOTE sampled data set and added some other lines of code commented out.
  - SMOTE is known to increase recall and reduce precision. This is a good starting point for our model as we want to identify all cases since the business does not want to miss out and cost themselves money. They are willing to hire more resources to review the large number of positive cases that get flagged.

```
#Try different parameter combinations to get us close to an ideal 50/50 split
#train_final_SMOTE <- smote(is_claim ~ ., train_final, perc.over = 10)
#train_final_SMOTE <- smote(is_claim ~ ., train_final, perc.over =10,perc.under = 1)
#train_final_SMOTE <- smote(is_claim ~ ., train_final, perc.over =20,perc.under = 1)
#train_final_SMOTE <- smote(is_claim ~ ., train_final, perc.over =30,perc.under = 1)

#perc.over=40 and perc.under=1 gets us pretty close to 50/50 split
train_final_SMOTE <- smote(is_claim ~ ., train_final, perc.over =40,perc.under = 1)
```

```
#New target variable count distribution
table(train_final_SMOTE$is_claim)
```

```
##
##      0      1
## 134480 137842
```

```
#New target variable percent distribution
prop.table(table(train_final_SMOTE$is_claim))
```

```
##
##         0         1
## 0.4938272 0.5061728
```

Now we refit and tune with the SMOTE sampled train set, run against the test set again, and then compare.

Random Forest with SMOTE:

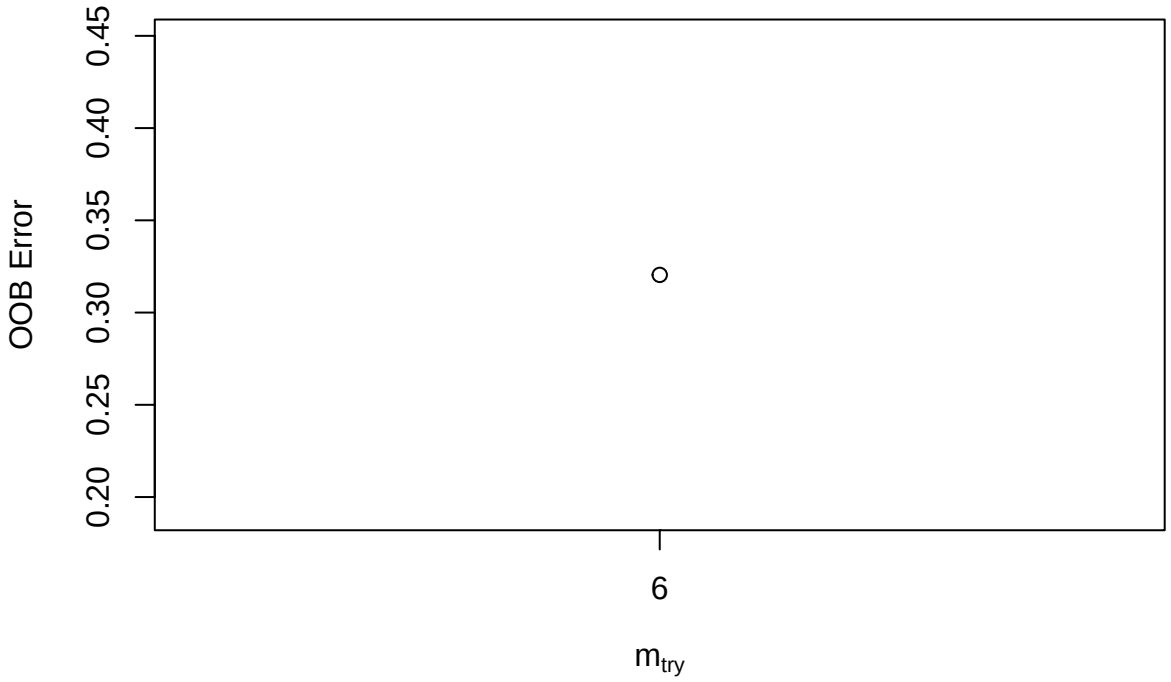
We fit the Random Forest model with SMOTE using all variables:

- We tried to fit a smaller model with less variables using the important plot, but it also gave worse performance. This analysis is not included here. Further tuning could be explored with more time.

```
#Random Forest Model With SMOTE:

#Tune for mtry. We go with 500 tree default although we did test other numbers of trees and see that mt
tuneRF(x = train_final_SMOTE[,-34],y = train_final_SMOTE$is_claim,ntreeTry = 500,trace=T, plot= T,impro
```

```
## mtry = 6  OOB error = 32.04%
## Searching left ...
## Searching right ...
```




```
##       mtry  OOBError
## 6.OOB    6 0.3204258
```

```r
#Fit with tuned mtry parameter and default 500 trees
RF_Model_1_SMOTE<-randomForest(is_claim ~ .,mtry=6,data = train_final_SMOTE)
#summary(RF_Model_1)
RF_Model_1_SMOTE
```

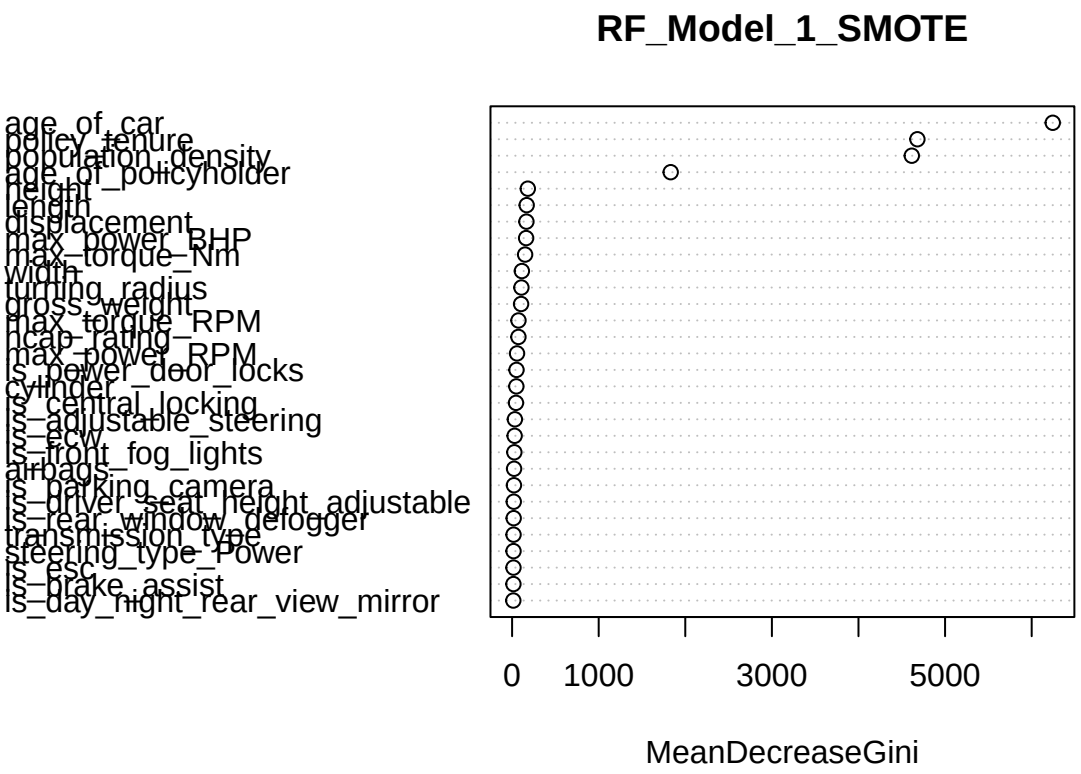
```
##
## Call:
##  randomForest(formula = is_claim ~ ., data = train_final_SMOTE,      mtry = 6)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 31.64%
## Confusion matrix:
##       0      1 class.error
## 0 73855  60625   0.4508105
## 1 25525 112317   0.1851758
```

```r
#Create vector of predictions
RF_predicted_1_SMOTE<-predict(RF_Model_1_SMOTE,test_final[,-34])

#Looking at precision, recall and F1 score:
confusionMatrix(RF_predicted_1_SMOTE,test_final$is_claim,mode = "everything",positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3010  125
##          1 2520  261
##
##                Accuracy : 0.5529
##                  95% CI : (0.5401, 0.5656)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0567
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.67617
##             Specificity : 0.54430
##          Pos Pred Value : 0.09385
##          Neg Pred Value : 0.96013
##               Precision : 0.09385
##                  Recall : 0.67617
##                      F1 : 0.16482
##              Prevalence : 0.06525
##          Detection Rate : 0.04412
##    Detection Prevalence : 0.47008
##       Balanced Accuracy : 0.61023
##
##        'Positive' Class : 1
##
```

```
#Variable Importance Plot; We see only a few variables are really important
varImpPlot(RF_Model_1_SMOTE)
```



Logistic Regression with SMOTE:

We train on the new SMOTE training data using the same two models we created previously (our full model and our reduced model). We already notice a nice improvement. We also see that the second Logistic Regression model does not provide really any improvement over the first logistic regression model.

```
#First model:

#We fit the first model using all variables:
mylogit_1_SMOTE <- glm(is_claim ~ ., data = train_final_SMOTE, family = "binomial")

#VIF indicates error that there is high multicollinearity
#vif(mylogit_1_SMOTE)

#We see NA coefficients indicating issues with fit
summary(mylogit_1_SMOTE)
```

```
##
## Call:
## glm(formula = is_claim ~ ., family = "binomial", data = train_final_SMOTE)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6577  -1.1442   0.8704   1.1225   1.9345
##
## Coefficients: (27 not defined because of singularities)
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -3.161e+01  3.750e+00  -8.431  < 2e-16 ***
```

```
## policy_tenure                          9.172e-01  1.038e-02  88.398  < 2e-16 ***
## age_of_car                            -4.400e+00  8.510e-02 -51.706  < 2e-16 ***
## age_of_policyholder                    6.360e-02  3.283e-02   1.937   0.0527 .
## population_density                    -2.890e-06  2.333e-07 -12.391  < 2e-16 ***
## airbags                               -7.969e-02  1.197e-02  -6.659 2.76e-11 ***
## is_escYes                             -2.276e+00  2.490e-01  -9.140  < 2e-16 ***
## is_adjustable_steeringYes             -1.598e+00  2.025e-01  -7.892 2.97e-15 ***
## is_tpmsYes                            -1.053e+00  1.247e-01  -8.445  < 2e-16 ***
## is_parking_sensorsYes                  3.975e+00  4.470e-01   8.892  < 2e-16 ***
## is_parking_cameraYes                   3.321e-01  6.134e-02   5.415 6.14e-08 ***
## rear_brakes_typeDrum                          NA         NA      NA       NA
## displacement                          -3.424e-03  4.098e-04  -8.354  < 2e-16 ***
## cylinder                               1.947e+00  2.222e-01   8.762  < 2e-16 ***
## transmission_typeManual               -1.964e+00  2.002e-01  -9.807  < 2e-16 ***
## gear_box                                      NA         NA      NA       NA
## turning_radius                         5.723e+00  6.914e-01   8.278  < 2e-16 ***
## length                                        NA         NA      NA       NA
## width                                         NA         NA      NA       NA
## height                                        NA         NA      NA       NA
## gross_weight                                  NA         NA      NA       NA
## is_front_fog_lightsYes                        NA         NA      NA       NA
## is_rear_window_wiperYes                       NA         NA      NA       NA
## is_rear_window_washerYes                      NA         NA      NA       NA
## is_rear_window_defoggerYes                    NA         NA      NA       NA
## is_brake_assistYes                            NA         NA      NA       NA
## is_power_door_locksYes                        NA         NA      NA       NA
## is_central_lockingYes                         NA         NA      NA       NA
## is_power_steeringYes                          NA         NA      NA       NA
## is_driver_seat_height_adjustableYes           NA         NA      NA       NA
## is_day_night_rear_view_mirrorYes              NA         NA      NA       NA
## is_ecwYes                                     NA         NA      NA       NA
## is_speed_alertYes                             NA         NA      NA       NA
## ncap_rating                                   NA         NA      NA       NA
## max_torque_Nm                                 NA         NA      NA       NA
## max_torque_RPM                                NA         NA      NA       NA
## max_power_BHP                                 NA         NA      NA       NA
## max_power_RPM                                 NA         NA      NA       NA
## fuel_type_Diesel1                             NA         NA      NA       NA
## fuel_type_Petrol1                             NA         NA      NA       NA
## steering_type_Manual1                         NA         NA      NA       NA
## steering_type_Power1                          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 377477  on 272321  degrees of freedom
## Residual deviance: 366154  on 272307  degrees of freedom
## AIC: 366184
##
## Number of Fisher Scoring iterations: 4
```

```
#Make predictions using first model
Log_predictions_1_SMOTE <- predict(mylogit_1_SMOTE, test_final[,-34], type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
#If prob exceeds threshold of 0.5, 1 else 0
Log_predictions_1_SMOTE <- ifelse(Log_predictions_1_SMOTE > 0.5, 1, 0)
#Convert to factor
Log_predictions_1_SMOTE <- as.factor(Log_predictions_1_SMOTE)
#Create confusion matrix
confusionMatrix(Log_predictions_1_SMOTE, test_final$is_claim,mode = "everything",positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3030  144
##          1 2500  242
##
##                Accuracy : 0.5531
##                  95% CI : (0.5403, 0.5658)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0456
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62694
##             Specificity : 0.54792
##          Pos Pred Value : 0.08826
##          Neg Pred Value : 0.95463
##               Precision : 0.08826
##                  Recall : 0.62694
##                      F1 : 0.15473
##              Prevalence : 0.06525
##          Detection Rate : 0.04091
##    Detection Prevalence : 0.46349
##       Balanced Accuracy : 0.58743
##
##        'Positive' Class : 1
##
```

```
#Second Model:

#We only take significant predictors right now, eliminate ones with high VIF values, and create a secon
mylogit_2_SMOTE <- glm(is_claim ~ population_density + age_of_policyholder + age_of_car + policy_tenure

#All variables are significant
summary(mylogit_2_SMOTE)
```

```
##
## Call:
## glm(formula = is_claim ~ population_density + age_of_policyholder +
##     age_of_car + policy_tenure, family = "binomial", data = train_final_SMOTE)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -1.6173  -1.1466   0.8923   1.1203   1.9381
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -3.137e-01  1.764e-02 -17.784  < 2e-16 ***
## population_density  -3.067e-06  2.324e-07 -13.195  < 2e-16 ***
## age_of_policyholder  9.235e-02  3.266e-02   2.828  0.00469 **
## age_of_car          -4.120e+00  7.569e-02 -54.429  < 2e-16 ***
## policy_tenure        9.289e-01  1.009e-02  92.033  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 377477  on 272321  degrees of freedom
## Residual deviance: 366601  on 272317  degrees of freedom
## AIC: 366611
##
## Number of Fisher Scoring iterations: 4
```

```
#VIF indicates no multicollinearity (no values 10 or above)
vif(mylogit_2_SMOTE)
```

```
##  population_density age_of_policyholder          age_of_car       policy_tenure
##            1.015964            1.024433            1.056859            1.082886
```

```
#Make predictions using second model
Log_predictions_2_SMOTE <- predict(mylogit_2_SMOTE, test_final[,-34], type = "response")
#If prob exceeds threshold of 0.5, 1 else 0
Log_predictions_2_SMOTE <- ifelse(Log_predictions_2_SMOTE > 0.5, 1, 0)
#Convert to factor
Log_predictions_2_SMOTE <- as.factor(Log_predictions_2_SMOTE)
#Create confusion matrix
confusionMatrix(Log_predictions_2_SMOTE, test_final$is_claim,mode = "everything",positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2959  143
##          1 2571  243
##
##                Accuracy : 0.5412
##                  95% CI : (0.5284, 0.554)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0419
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62953
##             Specificity : 0.53508
##          Pos Pred Value : 0.08635
##          Neg Pred Value : 0.95390
```

```
##               Precision : 0.08635
##                  Recall : 0.62953
##                      F1 : 0.15187
##              Prevalence : 0.06525
##          Detection Rate : 0.04108
##    Detection Prevalence : 0.47566
##       Balanced Accuracy : 0.58231
##
##        'Positive' Class : 1
##
```

Final Model Evaluation:

We see that SMOTE provides a good improvement for both models.

For random forest, precision goes from NA to 0.09385, which is not very high, but a good improvement. Its recall goes from 0 to 0.67617. Its F1 score goes from NA to 0.16482, which is not very high, but a good improvement.

For logistic regression for both models, precision goes from NA to 0.08826, recall goes from 0 to 0.62694, and F1 score goes from NA to 0.15473.

We have seen that SMOTE has done a good job of improving our models, especially in recall where we are able to identify more positive cases. This is good because the business is willing to take on the extra work to review these extra flagged cases. The model performance is ultimately not that great due to its low precision (people won't have great confidence in the model) and it's probably not ready for production use, but we would choose the random forest model since all three metrics are higher. We can also look at accuracy after SMOTE since the data is balanced, but even then, all models provide a very similiar accuracy metric (0.55).

```
#Show improvement from SMOTE for the Random Forest model:
confusionMatrix(RF_predicted_1,test_final$is_claim,mode = "everything",positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5530  386
##          1    0    0
##
##                Accuracy : 0.9348
##                  95% CI : (0.9282, 0.9409)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 0.5135
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.93475
##               Precision :      NA
##                  Recall : 0.00000
##                      F1 :      NA
##              Prevalence : 0.06525
##          Detection Rate : 0.00000
```

```
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```
confusionMatrix(RF_predicted_1_SMOTE,test_final$is_claim,mode = "everything",positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3010  125
##          1 2520  261
##
##                Accuracy : 0.5529
##                  95% CI : (0.5401, 0.5656)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0567
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.67617
##             Specificity : 0.54430
##          Pos Pred Value : 0.09385
##          Neg Pred Value : 0.96013
##               Precision : 0.09385
##                  Recall : 0.67617
##                      F1 : 0.16482
##              Prevalence : 0.06525
##          Detection Rate : 0.04412
##    Detection Prevalence : 0.47008
##       Balanced Accuracy : 0.61023
##
##        'Positive' Class : 1
##
```

```
#Show improvement from SMOTE for first Logistic Regression model:
confusionMatrix(Log_predictions_1, test_final$is_claim,mode = "everything",positive = "1")
```

```
## Warning in confusionMatrix.default(Log_predictions_1, test_final$is_claim, :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5530  386
##          1    0    0
##
##                Accuracy : 0.9348
##                  95% CI : (0.9282, 0.9409)
##     No Information Rate : 0.9348
```

```
##     P-Value [Acc > NIR] : 0.5135
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.93475
##               Precision :      NA
##                  Recall : 0.00000
##                      F1 :      NA
##              Prevalence : 0.06525
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```
confusionMatrix(Log_predictions_1_SMOTE, test_final$is_claim,mode = "everything",positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3030  144
##          1 2500  242
##
##                Accuracy : 0.5531
##                  95% CI : (0.5403, 0.5658)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0456
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62694
##             Specificity : 0.54792
##          Pos Pred Value : 0.08826
##          Neg Pred Value : 0.95463
##               Precision : 0.08826
##                  Recall : 0.62694
##                      F1 : 0.15473
##              Prevalence : 0.06525
##          Detection Rate : 0.04091
##    Detection Prevalence : 0.46349
##       Balanced Accuracy : 0.58743
##
##        'Positive' Class : 1
##
```

```r
#Show improvement from SMOTE for second Logistic Regression model:
confusionMatrix(Log_predictions_2, test_final$is_claim,mode = "everything",positive = "1")
```

```
## Warning in confusionMatrix.default(Log_predictions_2, test_final$is_claim, :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5530  386
##          1    0    0
##
##                Accuracy : 0.9348
##                  95% CI : (0.9282, 0.9409)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 0.5135
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.93475
##               Precision :      NA
##                  Recall : 0.00000
##                      F1 :      NA
##              Prevalence : 0.06525
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```r
confusionMatrix(Log_predictions_2_SMOTE, test_final$is_claim,mode = "everything",positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2959  143
##          1 2571  243
##
##                Accuracy : 0.5412
##                  95% CI : (0.5284, 0.554)
##     No Information Rate : 0.9348
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0419
##
##  Mcnemar's Test P-Value : <2e-16
```

```
##
##             Sensitivity : 0.62953
##             Specificity : 0.53508
##          Pos Pred Value : 0.08635
##          Neg Pred Value : 0.95390
##               Precision : 0.08635
##                  Recall : 0.62953
##                      F1 : 0.15187
##              Prevalence : 0.06525
##          Detection Rate : 0.04108
##    Detection Prevalence : 0.47566
##       Balanced Accuracy : 0.58231
##
##        'Positive' Class : 1
##
```

Limitations of Analysis:

- Data set is not continually updated so we would need to find a better source to put this model into production
  - Since the data set is small, this model will need to be retrained over time when we can receive more observations
  - Reminder that there are not many useful free insurance data sets available to the public
- Since this project was done by one person with limited time, future work could include more creative tuning of the data set and models
  - The final model seems to have pretty low precision even though it has decent recall. So it can only really be used depending on the business case. It is not a great model is we want to be sure of ourselves.

Value/Significance:

- We have created a model with a reasonable recall metric, which satisfies an insurance company's desire to identity more positive cases and it would be valuable to them in multiple ways:
  - Allows companies to anticipate future claim handling workload, manage staffing, and allocate resources properly
  - Allows companies to anticipate possible claim reserve amounts that need to be set aside for future claims
  - It gives companies the early notice to prepare and hopefully settle claims quicker and at lower amounts
  - It gives underwriting an idea of what policy holders might be risky and lets them make decisions on renewals/premiums
  - It gives loss control divisions an opportunity to intervene if they know a policyholder is at risk for a claim
- Even though the final model's precision is low, we have provided a baseline with suggestions for improvement and even concluded that Random Forest might be a better option over Logistic Rewgression for this type of data/prediction
- We produced this analysis on a imbalanced data set, which mirrors a common real world problem, and implemented a new solution known as SMOTE. This acted as a great learning opportunity to develop knoweldge and experience as a data scientist.

Future Work:

Since this entire project was done by one person with limited time, there is room to do more. If there was more time, we would explore areas to improve our models, our data, and find more interesting conclusions:

- Explore more advanced models to the ones used in this analysis
- Experiment with other sampling methods to balance the data
- Explore more creative feature engineering
- Collect more data
- Experiment with other tuning methods:
  - Weight one class more than the other
  - More extreme train/test splits (only tried 80/20 and 85/15)
- Other interesting methods to model insurance claims:
  - Survival Analysis
  - Poisson Regression
  - Claim count simulations