Project Insurance

July 8, 2024

1 Introduction:

As the automobile gradually becomes the necessity of every family, the car insurance also becomes more and more prosperous for auto insurance companies, to maximize revenue, they must sell corresponding insurance plans to different customers. However, because the types of customers are so diverse and the correlation between the characteristics is not obvious, the use of simple statistics cannot enable insurance companies to make accurate judgments about customers. With the advent of machine learning, more models are available to learn data in depth. Thus, more accurate predictions can be achieved.

2 Problem Statement:

Develop a predictive model that assesses the claim probability for car insurance policies. The objective would be to understand the factors that influence claim frequency and severity in the period of six months and enable insurance companies to better assess risk and determine appropriate premiums for policyholders.

3 Objective:

3.0.1 We aim to solve the problem statement by creating a plan of action, laid down are some of the necessary steps:

1. Data Ingestion
2. Exploratory Data Analysis (EDA)
3. Data Pre-processing
4. Feature Selection/Extraction
5. Predictive Modelling
6. Project Outcomes & Conclusion

3.1 *Dataset Attributes*:

1. **policy_id**: The unique identifier for each insurance policy.
2. **policy_tenure**: The length of time (in years) that the policy has been active.
3. **age_of_car**: The age of the insured car (in years) at the time the policy was taken.
4. **age_of_policyholder**: The age of the policyholder (in years) at the time the policy was taken.

5. **area_cluster**: A categorical variable representing the cluster or category to which the area of residence belongs.
6. **population_density**: A measure of the population density of the area where the policyholder resides.
7. **Make**: The make or manufacturer of the insured car.
8. **segment**: The segment or category to which the insured car belongs (e.g., compact, sedan, SUV).
9. **model**: The specific model or variant of the insured car.
10. **fuel_type**: The type of fuel used by the insured car (e.g., petrol, diesel, electric).
11. **max_torque**: The maximum torque output of the car's engine.
12. **max_power**: The maximum power output of the car's engine.
13. **engine_type**: The type of engine used in the insured car (e.g., inline, V-type).
14. **airbags**: The number of airbags installed in the car.
15. **is_esc**: A binary variable indicating whether the car has an electronic stability control (ESC) system.
16. **is_adjustable_steering**: A binary variable indicating whether the car has adjustable steering.
17. **is_tpms**: A binary variable indicating whether the car has a tire pressure monitoring system (TPMS).
18. **is_parking_sensors**: A binary variable indicating whether the car has parking sensors.
19. **is_parking_camera**: A binary variable indicating whether the car has a parking camera.
20. **rear_brakes_type**: The type of rear brakes used in the car.
21. **displacement**: The engine displacement of the car (typically measured in liters or cubic centimeters).
22. **cylinder**: The number of cylinders in the car's engine.
23. **transmission_type**: The type of transmission used in the car (e.g., manual, automatic).
24. **gear_box**: The number of gears in the car's gearbox.
25. **steering_type**: The type of steering system used in the car.
26. **turning_radius**: The minimum radius of the circular path that the car can make.
27. **length**: The length of the car.
28. **width**: The width of the car.
29. **height**: The height of the car.
30. **gross_weight**: The gross weight or total weight of the car.
31. **is_front_fog_lights**: A binary variable indicating whether the car has front fog lights.

32. **is_rear_window_wiper**: A binary variable indicating whether the car has a rear window wiper.
33. **is_rear_window_washer**: A binary variable indicating whether the car has a rear window washer.
34. **is_rear_window_defogger**: A binary variable indicating whether the car has a rear window defogger.
35. **is_brake_assist**: A binary variable indicating whether the car has a brake assist system.
36. **is_power_door_locks**: A binary variable indicating whether the car has power door locks.
37. **is_central_locking**: A binary variable indicating whether the car has central locking.
38. **is_power_steering**: A binary variable indicating whether the car has power steering.
39. **is_driver_seat_height_adjustable**: A binary variable indicating whether the driver's seat height is adjustable.
40. **is_day_night_rear_view_mirror**: A binary variable indicating whether the car has a day/night rearview mirror
41. **is_ecw**: A binary variable indicating whether the car has an electronic crash warning (ECW) system. ECW systems use sensors and algorithms to detect potential collisions and provide warnings to the driver.
42. **is_speed_alert**: A binary variable indicating whether the car has a speed alert system. Speed alert systems typically monitor the vehicle's speed and provide warnings or alerts to the driver when they exceed a predetermined speed limit.
43. **ncap_rating**: The safety rating of the car according to the New Car Assessment Program (NCAP). NCAP is a government-backed program that evaluates and rates the safety performance of new car models in various crash tests and assessments. The rating is usually represented by a star system, with a higher number of stars indicating a better safety performance.
44. **is_claim**: A binary variable indicating whether an insurance claim has been filed for the car policy. This variable determines whether an insurance event has occurred for a given policy, with a value of 1 indicating that a claim was filed and 0 indicating no claim was filed.

[1]:
```
# Importing basic analytical libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()
import warnings
warnings.filterwarnings("ignore")
pd.options.display.float_format = '{:.2f}'.format
pd.options.display.max_columns = None
```

4 1. Data Ingestion

[2]:
```
# Importing Dataset
dataset = pd.read_csv('Insurance_data.csv')
df = dataset.copy()
```

[3]:
```
# Dataset exploration
def exploring_dataset(data):
    print('')
    print('\x1b[1;
↪41m************************************Data_Head************************************')
    display(data.head())
    print('')
    print('\x1b[1;
↪41m************************************Data_Tail************************************')
    display(data.tail())
    print('')
  ␣
↪print('\x1b[1m************************************Data_Info********************************
    display(data.info())
    print('')
    print('\x1b[1;41m************************************Unique␣
↪Rows************************************')
    display(data.nunique().sort_values())
    print('')
    print('\x1b[1;
↪41m************************************Data_Missing_Values****************************
    display(data.isnull().sum().sort_values(ascending=False))
    print('')
    print('\x1b[1;
↪41m************************************Data_Describe************************************')
    display(data.describe())
exploring_dataset(df)
```

```
************************************Data_Head************************************

  policy_id  policy_tenure  age_of_car  age_of_policyholder area_cluster  \
0   ID00001           0.52        0.05                 0.64           C1
1   ID00002           0.67        0.02                 0.38           C2
2   ID00003           0.84        0.02                 0.38           C3
3   ID00004           0.90        0.11                 0.43           C4
4   ID00005           0.60        0.11                 0.63           C5

   population_density  make segment model fuel_type     max_torque  \
0                4990     1       A    M1       CNG  60Nm@3500rpm
1               27003     1       A    M1       CNG  60Nm@3500rpm
2                4076     1       A    M1       CNG  60Nm@3500rpm
```

```
3              21622     1       C1    M2    Petrol  113Nm@4400rpm
4              34738     2        A    M3    Petrol   91Nm@4250rpm

          max_power         engine_type  airbags is_esc  \
0  40.36bhp@6000rpm   F8D Petrol Engine        2     No
1  40.36bhp@6000rpm   F8D Petrol Engine        2     No
2  40.36bhp@6000rpm   F8D Petrol Engine        2     No
3  88.50bhp@6000rpm  1.2 L K12N Dualjet        2    Yes
4  67.06bhp@5500rpm             1.0 SCe        2     No

  is_adjustable_steering is_tpms is_parking_sensors is_parking_camera  \
0                     No      No                Yes                No
1                     No      No                Yes                No
2                     No      No                Yes                No
3                    Yes      No                Yes               Yes
4                     No      No                 No               Yes

  rear_brakes_type  displacement  cylinder transmission_type  gear_box  \
0             Drum           796         3            Manual         5
1             Drum           796         3            Manual         5
2             Drum           796         3            Manual         5
3             Drum          1197         4         Automatic         5
4             Drum           999         3         Automatic         5

  steering_type  turning_radius  length  width  height  gross_weight  \
0         Power            4.60    3445   1515    1475          1185
1         Power            4.60    3445   1515    1475          1185
2         Power            4.60    3445   1515    1475          1185
3      Electric            4.80    3995   1735    1515          1335
4      Electric            5.00    3731   1579    1490          1155

  is_front_fog_lights is_rear_window_wiper is_rear_window_washer  \
0                  No                   No                    No
1                  No                   No                    No
2                  No                   No                    No
3                 Yes                   No                    No
4                  No                   No                    No

  is_rear_window_defogger is_brake_assist is_power_door_locks  \
0                      No              No                  No
1                      No              No                  No
2                      No              No                  No
3                     Yes             Yes                 Yes
4                      No              No                 Yes

  is_central_locking is_power_steering is_driver_seat_height_adjustable  \
0                 No               Yes                               No
1                 No               Yes                               No
```

```
2                 No                Yes                                 No
3                Yes                Yes                                Yes
4                Yes                Yes                                 No

  is_day_night_rear_view_mirror is_ecw is_speed_alert  ncap_rating  is_claim
0                            No     No            Yes            0         0
1                            No     No            Yes            0         0
2                            No     No            Yes            0         0
3                           Yes    Yes            Yes            2         0
4                           Yes    Yes            Yes            2         0
```

```
***********************************Data_Tail***********************************
      policy_id  policy_tenure  age_of_car  age_of_policyholder area_cluster  \
58587   ID58588           0.36        0.13                 0.64           C8
58588   ID58589           1.20        0.02                 0.52          C14
58589   ID58590           1.16        0.05                 0.45           C5
58590   ID58591           1.24        0.14                 0.56           C8
58591   ID58592           0.12        0.02                 0.44           C8

       population_density  make segment model fuel_type     max_torque  \
58587                8794     2       A    M3    Petrol   91Nm@4250rpm
58588                7788     1       A    M1       CNG   60Nm@3500rpm
58589               34738     1       A    M1       CNG   60Nm@3500rpm
58590                8794     1      B2    M6    Petrol  113Nm@4400rpm
58591                8794     3      C2    M4    Diesel  250Nm@2750rpm

               max_power        engine_type  airbags is_esc  \
58587   67.06bhp@5500rpm           1.0 SCe        2     No
58588   40.36bhp@6000rpm  F8D Petrol Engine        2     No
58589   40.36bhp@6000rpm  F8D Petrol Engine        2     No
58590   88.50bhp@6000rpm  K Series Dual jet        2     No
58591  113.45bhp@4000rpm      1.5 L U2 CRDi        6    Yes

      is_adjustable_steering is_tpms is_parking_sensors is_parking_camera  \
58587                     No      No                 No               Yes
58588                     No      No                Yes                No
58589                     No      No                Yes                No
58590                    Yes      No                Yes                No
58591                    Yes     Yes                Yes               Yes

      rear_brakes_type  displacement  cylinder transmission_type  gear_box  \
58587             Drum           999         3         Automatic         5
58588             Drum           796         3            Manual         5
58589             Drum           796         3            Manual         5
58590             Drum          1197         4            Manual         5
58591             Disc          1493         4         Automatic         6
```

	steering_type	turning_radius	length	width	height	gross_weight
58587	Electric	5.00	3731	1579	1490	1155
58588	Power	4.60	3445	1515	1475	1185
58589	Power	4.60	3445	1515	1475	1185
58590	Electric	4.80	3845	1735	1530	1335
58591	Power	5.20	4300	1790	1635	1720

	is_front_fog_lights	is_rear_window_wiper	is_rear_window_washer
58587	No	No	No
58588	No	No	No
58589	No	No	No
58590	Yes	No	No
58591	Yes	Yes	Yes

	is_rear_window_defogger	is_brake_assist	is_power_door_locks
58587	No	No	Yes
58588	No	No	No
58589	No	No	No
58590	No	Yes	Yes
58591	Yes	Yes	Yes

	is_central_locking	is_power_steering	is_driver_seat_height_adjustable
58587	Yes	Yes	No
58588	No	Yes	No
58589	No	Yes	No
58590	Yes	Yes	Yes
58591	Yes	Yes	Yes

	is_day_night_rear_view_mirror	is_ecw	is_speed_alert	ncap_rating
58587	Yes	Yes	Yes	2
58588	No	No	Yes	0
58589	No	No	Yes	0
58590	Yes	Yes	Yes	2
58591	No	Yes	Yes	3

	is_claim
58587	0
58588	0
58589	0
58590	0
58591	0

```
*********************************Data_Info******************************
*****
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58592 entries, 0 to 58591
Data columns (total 44 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   policy_id                         58592 non-null  object
 1   policy_tenure                     58592 non-null  float64
 2   age_of_car                        58592 non-null  float64
 3   age_of_policyholder               58592 non-null  float64
 4   area_cluster                      58592 non-null  object
 5   population_density                58592 non-null  int64
 6   make                              58592 non-null  int64
 7   segment                           58592 non-null  object
 8   model                             58592 non-null  object
 9   fuel_type                         58592 non-null  object
 10  max_torque                        58592 non-null  object
 11  max_power                         58592 non-null  object
 12  engine_type                       58592 non-null  object
 13  airbags                           58592 non-null  int64
 14  is_esc                            58592 non-null  object
 15  is_adjustable_steering            58592 non-null  object
 16  is_tpms                           58592 non-null  object
 17  is_parking_sensors                58592 non-null  object
 18  is_parking_camera                 58592 non-null  object
 19  rear_brakes_type                  58592 non-null  object
 20  displacement                      58592 non-null  int64
 21  cylinder                          58592 non-null  int64
 22  transmission_type                 58592 non-null  object
 23  gear_box                          58592 non-null  int64
 24  steering_type                     58592 non-null  object
 25  turning_radius                    58592 non-null  float64
 26  length                            58592 non-null  int64
 27  width                             58592 non-null  int64
 28  height                            58592 non-null  int64
```

None

```
**********************************UniqueRows**********************************
cylinder                                2
transmission_type                       2
gear_box                                2
is_front_fog_lights                     2
is_rear_window_wiper                    2
is_rear_window_washer                   2
is_rear_window_defogger                 2
rear_brakes_type                        2
is_brake_assist                         2
is_central_locking                      2
is_power_steering                       2
is_driver_seat_height_adjustable        2
is_day_night_rear_view_mirror           2
is_ecw                                  2
is_speed_alert                          2
is_power_door_locks                     2
is_parking_camera                       2
is_claim                                2
is_tpms                                 2
is_adjustable_steering                  2
is_parking_sensors                      2
is_esc                                  2
airbags                                 3
steering_type                           3
fuel_type                               3
ncap_rating                             5
make                                    5
segment                                 6
turning_radius                          9
displacement                            9
max_power                               9
max_torque                              9
length                                  9
gross_weight                           10
width                                  10
engine_type                            11
height                                 11
model                                  11
population_density                     22
area_cluster                           22
age_of_car                             49
age_of_policyholder                    75
policy_tenure                       58591
policy_id                           58592
```

```
dtype: int64
```

```
************************************Data_Missing_Values************************************
policy_id                           0
policy_tenure                       0
steering_type                       0
turning_radius                      0
length                              0
width                               0
height                              0
gross_weight                        0
is_front_fog_lights                 0
is_rear_window_wiper                0
is_rear_window_washer               0
is_rear_window_defogger             0
is_brake_assist                     0
is_power_door_locks                 0
is_central_locking                  0
is_power_steering                   0
is_driver_seat_height_adjustable    0
is_day_night_rear_view_mirror       0
is_ecw                              0
is_speed_alert                      0
ncap_rating                         0
gear_box                            0
transmission_type                   0
cylinder                            0
max_torque                          0
age_of_car                          0
age_of_policyholder                 0
area_cluster                        0
population_density                  0
make                                0
segment                             0
model                               0
fuel_type                           0
max_power                           0
displacement                        0
engine_type                         0
airbags                             0
is_esc                              0
is_adjustable_steering              0
is_tpms                             0
is_parking_sensors                  0
is_parking_camera                   0
rear_brakes_type                    0
is_claim                            0
```

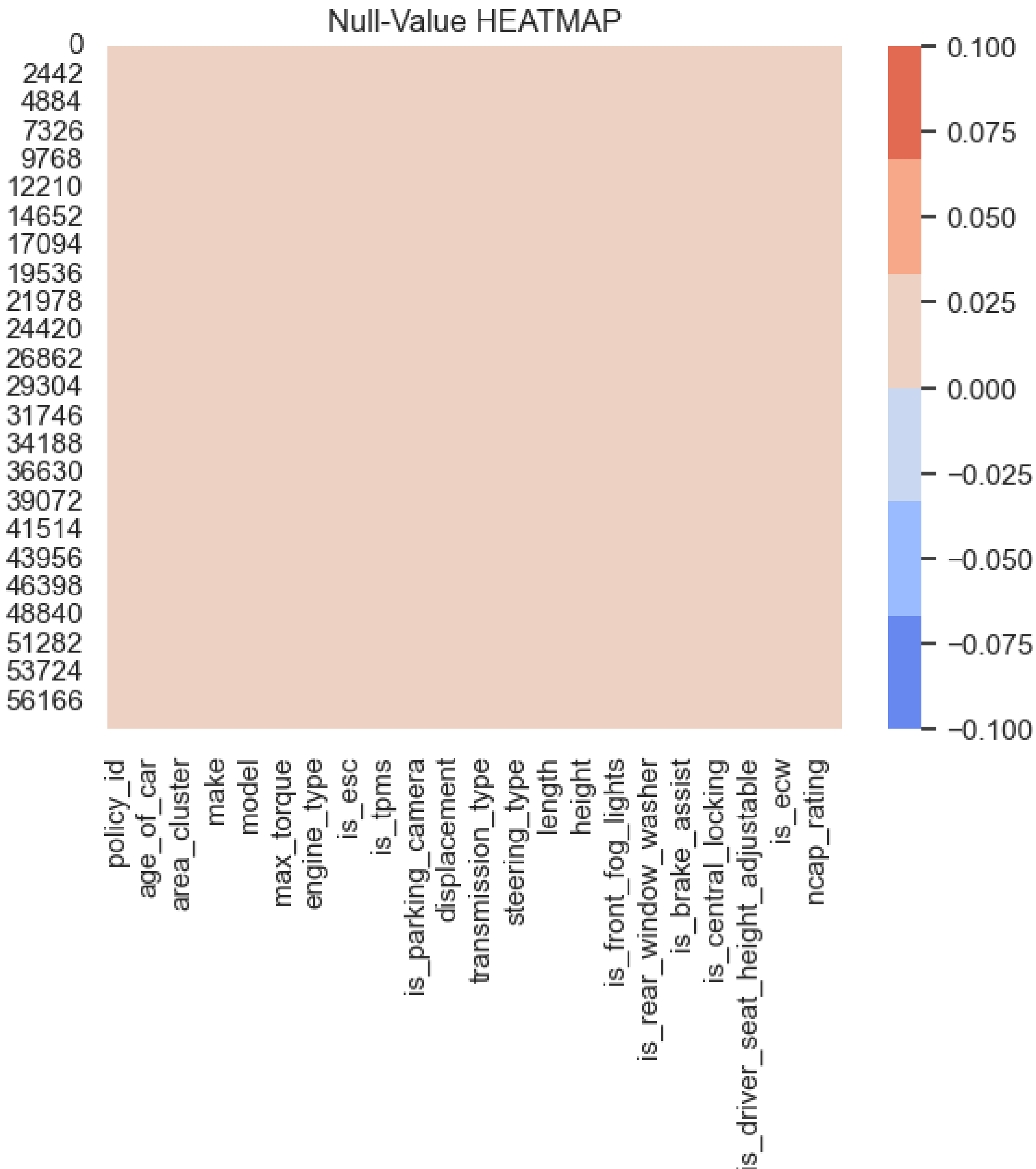
```
dtype: int64
```

************************************Data_Describe************************************

```
       policy_tenure  age_of_car  age_of_policyholder  population_density  \
count       58592.00    58592.00             58592.00            58592.00
mean            0.61        0.07                 0.47            18826.86
std             0.41        0.06                 0.12            17660.17
min             0.00        0.00                 0.29              290.00
25%             0.21        0.02                 0.37             6112.00
50%             0.57        0.06                 0.45             8794.00
75%             1.04        0.11                 0.55            27003.00
max             1.40        1.00                 1.00            73430.00

          make  airbags  displacement  cylinder  gear_box  turning_radius  \
count 58592.00 58592.00      58592.00  58592.00  58592.00        58592.00
mean      1.76     3.14       1162.36      3.63      5.25            4.85
std       1.14     1.83        266.30      0.48      0.43            0.23
min       1.00     1.00        796.00      3.00      5.00            4.50
25%       1.00     2.00        796.00      3.00      5.00            4.60
50%       1.00     2.00       1197.00      4.00      5.00            4.80
75%       3.00     6.00       1493.00      4.00      5.00            5.00
max       5.00     6.00       1498.00      4.00      6.00            5.20

        length    width   height  gross_weight  ncap_rating  is_claim
count 58592.00 58592.00 58592.00      58592.00     58592.00  58592.00
mean   3850.48  1672.23  1553.34       1385.28         1.76      0.06
std     311.46   112.09    79.62        212.42         1.39      0.24
min    3445.00  1475.00  1475.00       1051.00         0.00      0.00
25%    3445.00  1515.00  1475.00       1185.00         0.00      0.00
50%    3845.00  1735.00  1530.00       1335.00         2.00      0.00
75%    3995.00  1755.00  1635.00       1510.00         3.00      0.00
max    4300.00  1811.00  1825.00       1720.00         5.00      1.00
```

- **Checking null values through heatmap**

[4]:
```python
sns.heatmap(df.isnull(),cmap=sns.color_palette('coolwarm'))
plt.title('Null-Value HEATMAP')
plt.show()
```



[5]:
```python
# Checking for duplicates
df.duplicated().sum()
```

[5]: 0

[6]:
```python
# Dropping "Policy ID" column not required
df.drop(columns='policy_id',inplace=True)
```

5 2. Exploratory Data Analysis (EDA)

[7]:
```
# Segregating features and target variable
target = 'is_claim'
labels = ['Not Claimed', 'Claimed']
features = [i for i in df.columns.values if i not in target]
```

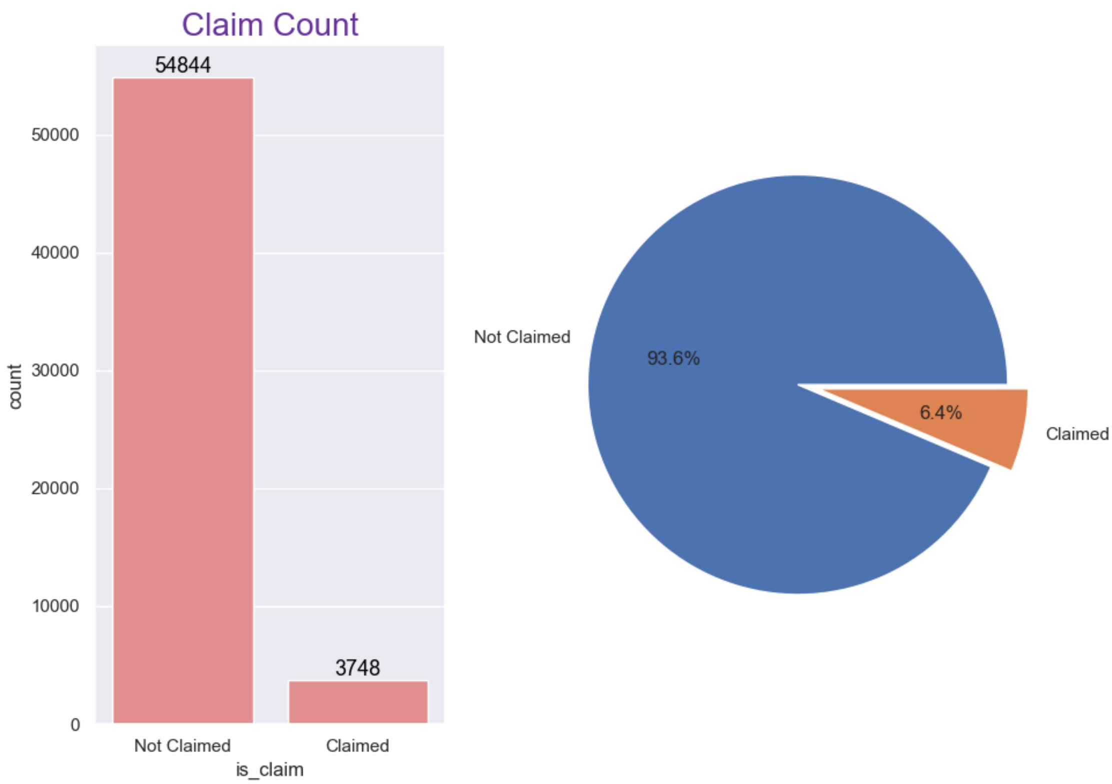
[8]:
```
# Segregating features and target variable coulmn names
unnq = df[features].nunique().sort_values()
nf = []; cf = []; nnf = 0; ncf = 0; #numerical & categorical features

for i in range(df[features].shape[1]):
    if unnq.values[i]<=7:cf.append(unnq.index[i])
    else: nf.append(unnq.index[i])

# Re-allocating correct dtypes as per categorical and numerical features
cf.extend(['max_power','max_torque', 'model', 'engine_type', 'area_cluster'])
nf = df[nf].select_dtypes(exclude='object').columns.values.tolist()
cf.remove('airbags')
nf.append('airbags')

nnf = len(nf); ncf = len(cf)
print('\n\x1b[7mInference:\x1b[0m The Datset has {} numerical & {} categorical features.'.format(nnf,ncf))
```

```
Inference: The Datset has 11 numerical & 31 categorical features.
```

[9]:
```
MAP={}
for e, i in enumerate(sorted(df[target].unique())):
    MAP[i]=labels[e]
# MAP={0:'Not-Claimed',1:'Claimed'}
df[target]=df[target].map(MAP)
explode=np.zeros(len(labels))
explode[-1]=0.15
print('\033[7m\t  Target Variable Distribution'.center(55))

fig = plt.subplots(nrows=1,ncols=2,figsize=(10, 7))
plt.subplot(1,2,1)
plots = sns.barplot(df[target].value_counts(),color='lightcoral')
for bar in plots.patches:
    plots.annotate(int(bar.get_height()),
                   (bar.get_x() + bar.get_width() / 2,
                    bar.get_height()), ha='center', va='bottom',
                   size=13,color='black')
plt.title("Claim Count",fontdict={'fontsize':20,'color':'rebeccapurple'})

plt.subplot(1,2,2)
```

```
plt.pie(df[target].value_counts(), labels=df[target].value_counts().index, counterclock=True,explode=explode, autopct='%1.1f%%',radius=1.5)
plt.tight_layout()
plt.show()
```

```
    Target Variable Distribution
```



From the above visualization we can conclude that only 6.4% of policyholder of total record data has claimed.

5.0.1 Visualising the Categorical features

[10]:
```
colors=['red', 'lawngreen']
fig = plt.subplots(nrows=16, ncols=2, figsize=(30,100))
for n,column in enumerate(cf):
    plot=plt.subplot(16,2,n+1)
    ax = sns.countplot(x=cf[n], data=df, hue=target, palette=colors, edgecolor='black')
    for rect in ax.patches:
        if rect.get_xy() != (0,0):
            ax.text(rect.get_x() + rect.get_width()/2,rect.get_height() + 2, rect.get_height(),
```

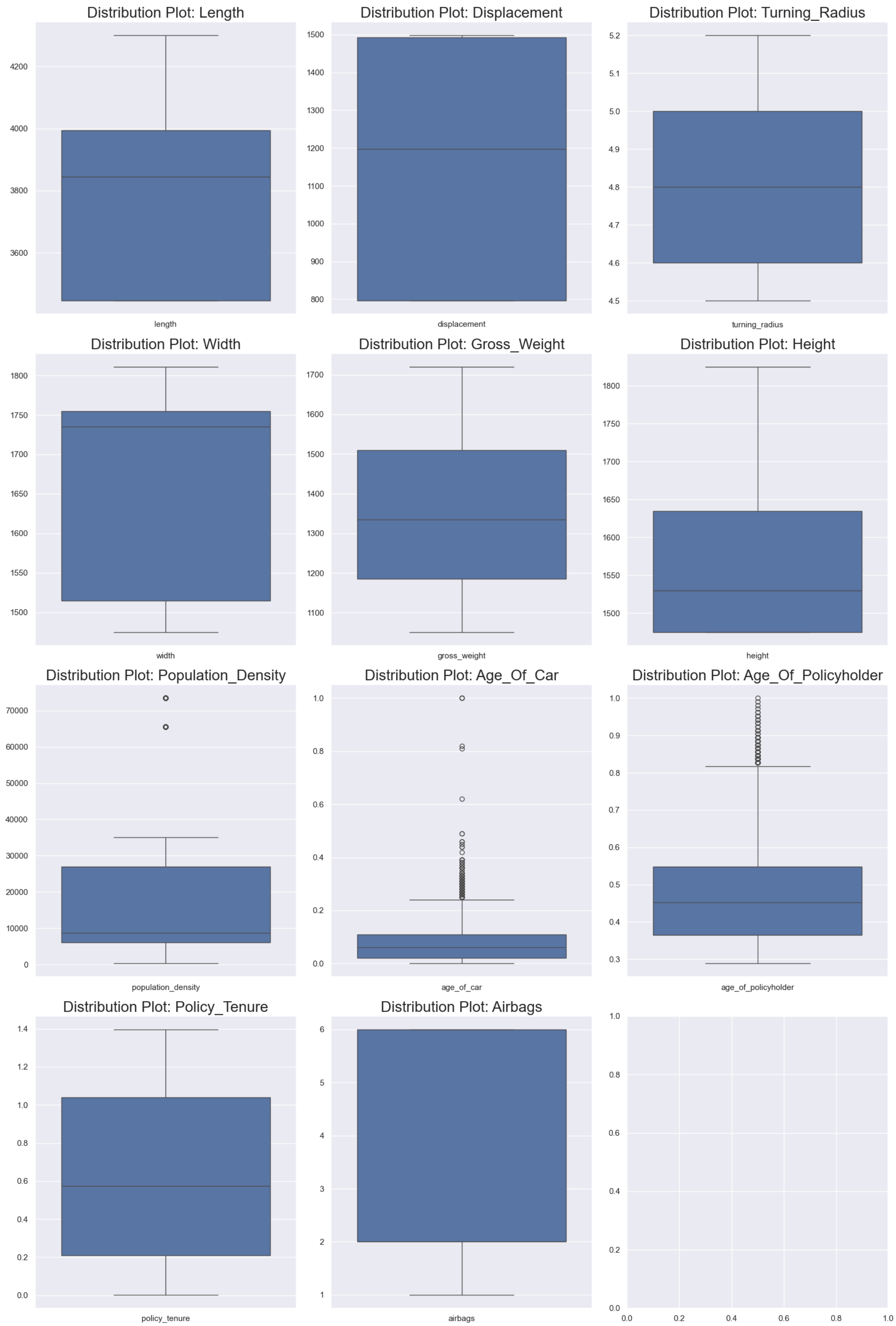
```
                ha='center',va='bottom',fontdict={'fontsize':15})
            title = cf[n] + ' vs Claim'
            plt.title(title.title(),fontdict={'fontsize':20}, weight='bold');
    plt.tight_layout()
plt.show()
```


Cylinder Vs Claim
Transmission_Type Vs Claim
Is_Speed_Alert Vs Claim
Is_Front_Fog_Lights Vs Claim
Rear_Brakes_Type Vs Claim
Is_Parking_Camera Vs Claim
Is_Parking_Sensors Vs Claim
Is_Tpms Vs Claim
Is_Adjustable_Steering Vs Claim
Is_Esc Vs Claim
Gear_Box Vs Claim
Is_Rear_Window_Wiper Vs Claim
Is_Rear_Window_Defogger Vs Claim
Is_Brake_Assist Vs Claim
Is_Power_Door_Locks Vs Claim
Is_Central_Locking Vs Claim
Is_Power_Steering Vs Claim
Is_Driver_Seat_Height_Adjustable Vs Claim
Is_Day_Night_Rear_View_Mirror Vs Claim
Is_Ecw Vs Claim
Is_Rear_Window_Washer Vs Claim
Fuel_Type Vs Claim
Steering_Type Vs Claim
Ncap_Rating Vs Claim
Make Vs Claim
Segment Vs Claim
Max_Power Vs Claim
Max_Torque Vs Claim
Model Vs Claim
Engine_Type Vs Claim
Area_Cluster Vs Claim

[11]:

```
fig = plt.subplots(nrows=4,ncols=3,figsize=(17,25))
for n,column in enumerate(nf):
    plt.subplot(4,3,n+1)
    sns.distplot(df[[nf[n]]],color='mediumslateblue',bins=20)
    title = 'Distribution plot: '+nf[n]
    plt.title(title.title(),fontdict={'fontsize':20})
plt.tight_layout()
plt.show()

fig = plt.subplots(nrows=4,ncols=3,figsize=(17,25))
for n,column in enumerate(nf):
    plt.subplot(4,3,n+1)
    sns.boxplot(df[[nf[n]]])
    title = 'Distribution plot: '+nf[n]
    plt.title(title.title(),fontdict={'fontsize':20})
plt.tight_layout()
plt.show()
```


Distribution Plot: Length
Distribution Plot: Displacement
Distribution Plot: Turning_Radius
Distribution Plot: Width
Distribution Plot: Gross_Weight
Distribution Plot: Height
Distribution Plot: Population_Density
Distribution Plot: Age_Of_Car
Distribution Plot: Age_Of_Policyholder
Distribution Plot: Policy_Tenure
Distribution Plot: Airbags
Density
1e−5



Distribution Plot: Length
length
Distribution Plot: Displacement
displacement
Distribution Plot: Turning_Radius
turning_radius
Distribution Plot: Width
width
Distribution Plot: Gross_Weight
gross_weight
Distribution Plot: Height
height
Distribution Plot: Population_Density
population_density
Distribution Plot: Age_Of_Car
age_of_car
Distribution Plot: Age_Of_Policyholder
age_of_policyholder
Distribution Plot: Policy_Tenure
policy_tenure
Distribution Plot: Airbags
airbags

- ***The above distribution plot and box plot shows features 'Population Density', 'Age of Car' and 'Age of policy holder' have outliers.***

6 3. Data Pre-Processing

[12]:
```
# Outlier Treatment
for col in ['population_density', 'age_of_car', 'age_of_policyholder']:
    Q3 = np.quantile(a=df[col],q=0.75)
    Q1 = np.quantile(a=df[col],q=0.25)
    iqr = Q3 - Q1
    UL = Q3 + 1.5*iqr
    LL = Q1 - 1.5*iqr
    df[col].clip(lower=LL,upper=UL,inplace=True)
```

[13]:
```
# Sorting categorical columns out of numerical columns for label encoding
text_data_features = [i for i in df.columns if i not in df.describe().columns]
print(text_data_features)
```

```
['area_cluster', 'segment', 'model', 'fuel_type', 'max_torque', 'max_power',
'engine_type', 'is_esc', 'is_adjustable_steering', 'is_tpms',
'is_parking_sensors', 'is_parking_camera', 'rear_brakes_type',
'transmission_type', 'steering_type', 'is_front_fog_lights',
'is_rear_window_wiper', 'is_rear_window_washer', 'is_rear_window_defogger',
'is_brake_assist', 'is_power_door_locks', 'is_central_locking',
'is_power_steering', 'is_driver_seat_height_adjustable',
'is_day_night_rear_view_mirror', 'is_ecw', 'is_speed_alert', 'is_claim']
```

[14]:
```
# Label Encoding categorical columns
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in text_data_features:
    df[i] = le.fit_transform(df[i])
    print(i,':',df[i].unique(),'=',le.inverse_transform(df[i].unique()))
```

```
area_cluster : [ 0 11 15 16 17 18 19 20 21  1  2  3  4  5  6  7  8  9 10 12 13
14] = ['C1' 'C2' 'C3' 'C4' 'C5' 'C6' 'C7' 'C8' 'C9' 'C10' 'C11' 'C12' 'C13'
 'C14' 'C15' 'C16' 'C17' 'C18' 'C19' 'C20' 'C21' 'C22']
segment : [0 3 4 2 1 5] = ['A' 'C1' 'C2' 'B2' 'B1' 'Utility']
model : [ 0  3  4  5  6  7  8  9 10  1  2] = ['M1' 'M2' 'M3' 'M4' 'M5' 'M6' 'M7'
'M8' 'M9' 'M10' 'M11']
fuel_type : [0 2 1] = ['CNG' 'Petrol' 'Diesel']
max_torque : [5 0 8 4 3 6 2 7 1] = ['60Nm@3500rpm' '113Nm@4400rpm'
'91Nm@4250rpm' '250Nm@2750rpm'
 '200Nm@3000rpm' '82.1Nm@3400rpm' '200Nm@1750rpm' '85Nm@3000rpm'
 '170Nm@4000rpm']
```

```
max_power : [2 6 5 0 7 3 8 4 1] = ['40.36bhp@6000rpm' '88.50bhp@6000rpm'
'67.06bhp@5500rpm'
 '113.45bhp@4000rpm' '88.77bhp@4000rpm' '55.92bhp@5300rpm'
 '97.89bhp@3600rpm' '61.68bhp@6000rpm' '118.36bhp@5500rpm']
engine_type : [ 6  2  0  3  4  8  1  9 10  7  5] = ['F8D Petrol Engine' '1.2 L
K12N Dualjet' '1.0 SCe' '1.5 L U2 CRDi'
 '1.5 Turbocharged Revotorq' 'K Series Dual jet' '1.2 L K Series Engine'
 'K10C' 'i-DTEC' 'G12B' '1.5 Turbocharged Revotron']
is_esc : [0 1] = ['No' 'Yes']
is_adjustable_steering : [0 1] = ['No' 'Yes']
is_tpms : [0 1] = ['No' 'Yes']
is_parking_sensors : [1 0] = ['Yes' 'No']
is_parking_camera : [0 1] = ['No' 'Yes']
rear_brakes_type : [1 0] = ['Drum' 'Disc']
transmission_type : [1 0] = ['Manual' 'Automatic']
steering_type : [2 0 1] = ['Power' 'Electric' 'Manual']
is_front_fog_lights : [0 1] = ['No' 'Yes']
is_rear_window_wiper : [0 1] = ['No' 'Yes']
is_rear_window_washer : [0 1] = ['No' 'Yes']
is_rear_window_defogger : [0 1] = ['No' 'Yes']
is_brake_assist : [0 1] = ['No' 'Yes']
is_power_door_locks : [0 1] = ['No' 'Yes']
is_central_locking : [0 1] = ['No' 'Yes']
is_power_steering : [1 0] = ['Yes' 'No']
is_driver_seat_height_adjustable : [0 1] = ['No' 'Yes']
is_day_night_rear_view_mirror : [0 1] = ['No' 'Yes']
is_ecw : [0 1] = ['No' 'Yes']
is_speed_alert : [1 0] = ['Yes' 'No']
is_claim : [1 0] = ['Not Claimed' 'Claimed']
```

[15]:
```
# Reversing label encoding for claims
df.is_claim.replace({1:0,0:1},inplace=True)
```

[16]:
```
df.describe().T[df.describe().T['mean']>10]
```

[16]:

	count	mean	std	min	25%	50%	\
area_cluster	58592.00	13.04	6.80	0.00	6.00	15.00	
population_density	58592.00	17953.59	15146.18	290.00	6112.00	8794.00	
displacement	58592.00	1162.36	266.30	796.00	796.00	1197.00	
length	58592.00	3850.48	311.46	3445.00	3445.00	3845.00	
width	58592.00	1672.23	112.09	1475.00	1515.00	1735.00	
height	58592.00	1553.34	79.62	1475.00	1475.00	1530.00	
gross_weight	58592.00	1385.28	212.42	1051.00	1185.00	1335.00	

	75%	max
area_cluster	20.00	21.00
population_density	27003.00	58339.50

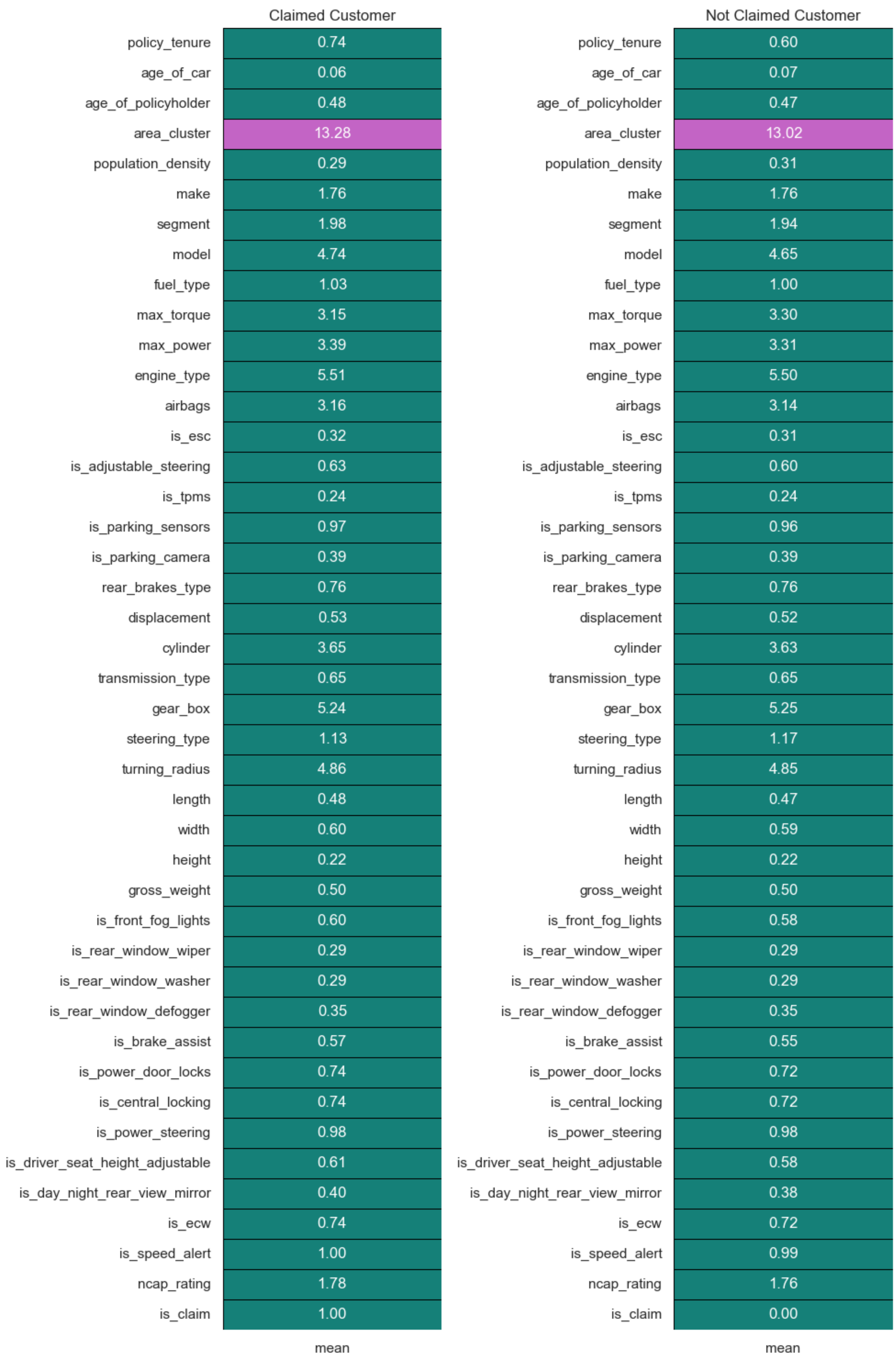
```
displacement        1493.00  1498.00
length              3995.00  4300.00
width               1755.00  1811.00
height              1635.00  1825.00
gross_weight        1510.00  1720.00
```

[17]:
```
scale_index = np.where([df.describe().T['mean']>100])[1].tolist()
scale_index
```

[17]:
```
[4, 19, 25, 26, 27, 28]
```

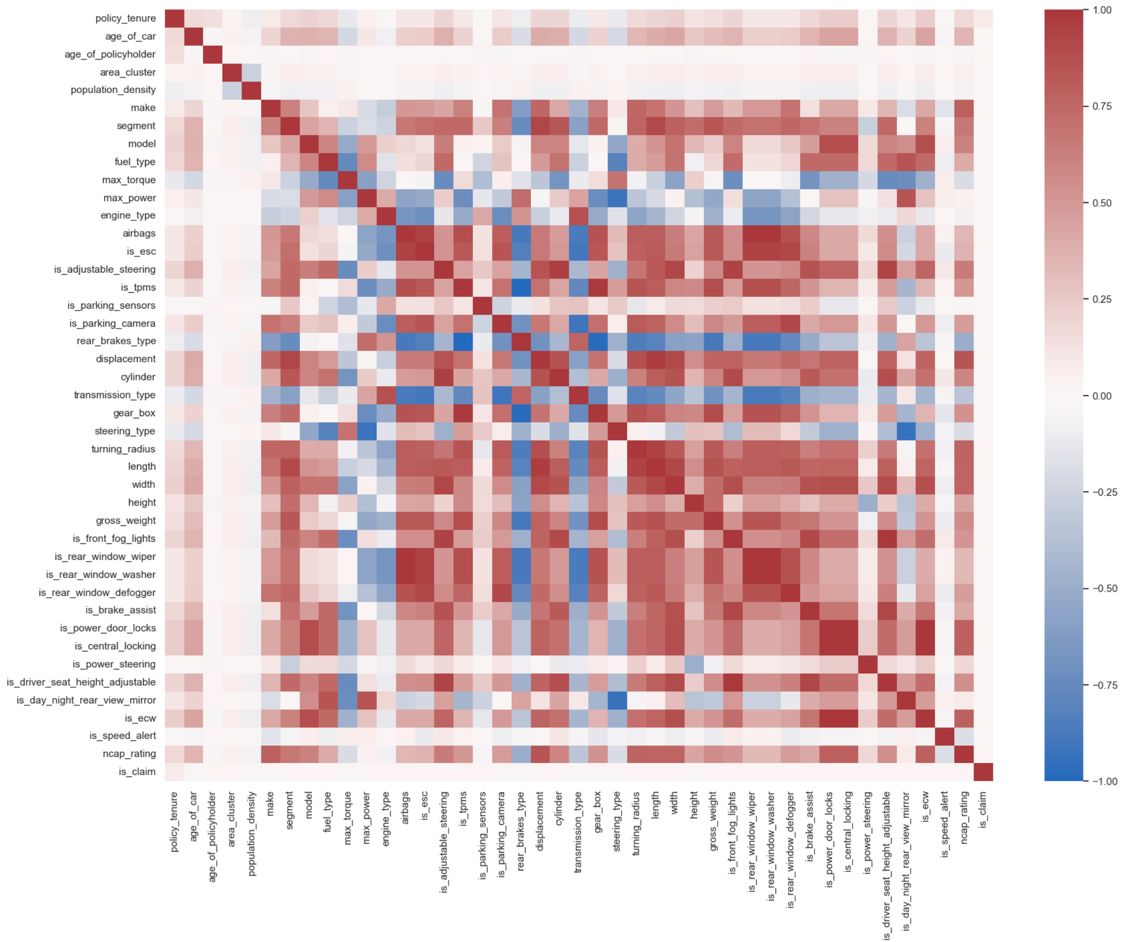
[18]:
```
# Standard scaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df.iloc[:,[4, 19, 25, 26, 27, 28]] = scaler.fit_transform(df.iloc[:,[4, 19, 25,␣
 ↪26, 27, 28]])
```

[19]:
```
# Visualising mean of features for Claimed and Not_Claimed customers through␣
 ↪heatmap
colors = ['#158078', '#C364C5']

claim = df[df['is_claim']==1].describe().T
not_claim = df[df['is_claim']==0].describe().T

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10,15))
plt.subplot(1,2,1)
sns.heatmap(claim[['mean']], annot=True, cmap=colors, linewidths=0.4, linecolor␣
 ↪= 'black',
            cbar=False, fmt='.2f')

plt.title('Claimed Customer')

plt.subplot(1,2,2)
sns.heatmap(not_claim[['mean']], annot=True, cmap=colors, linewidths=0.4,␣
 ↪linecolor = 'black',
            cbar=False, fmt='.2f')
plt.title('Not Claimed Customer')

fig.tight_layout(pad=1)
```


Claimed Customer
policy_tenure 0.74
age_of_car 0.06
age_of_policyholder 0.48
area_cluster 13.28
population_density 0.29
make 1.76
segment 1.98
model 4.74
fuel_type 1.03
max_torque 3.15
max_power 3.39
engine_type 5.51
airbags 3.16
is_esc 0.32
is_adjustable_steering 0.63
is_tpms 0.24
is_parking_sensors 0.97
is_parking_camera 0.39
rear_brakes_type 0.76
displacement 0.53
cylinder 3.65
transmission_type 0.65
gear_box 5.24
steering_type 1.13
turning_radius 4.86
length 0.48
width 0.60
height 0.22
gross_weight 0.50
is_front_fog_lights 0.60
is_rear_window_wiper 0.29
is_rear_window_washer 0.29
is_rear_window_defogger 0.35
is_brake_assist 0.57
is_power_door_locks 0.74
is_central_locking 0.74
is_power_steering 0.98
is_driver_seat_height_adjustable 0.61
is_day_night_rear_view_mirror 0.40
is_ecw 0.74
is_speed_alert 1.00
ncap_rating 1.78
is_claim 1.00
mean
Not Claimed Customer
policy_tenure 0.60
age_of_car 0.07
age_of_policyholder 0.47
area_cluster 13.02
population_density 0.31
make 1.76
segment 1.94
model 4.65
fuel_type 1.00
max_torque 3.30
max_power 3.31
engine_type 5.50
airbags 3.14
is_esc 0.31
is_adjustable_steering 0.60
is_tpms 0.24
is_parking_sensors 0.96
is_parking_camera 0.39
rear_brakes_type 0.76
displacement 0.52
cylinder 3.63
transmission_type 0.65
gear_box 5.25
steering_type 1.17
turning_radius 4.85
length 0.47
width 0.59
height 0.22
gross_weight 0.50
is_front_fog_lights 0.58
is_rear_window_wiper 0.29
is_rear_window_washer 0.29
is_rear_window_defogger 0.35
is_brake_assist 0.55
is_power_door_locks 0.72
is_central_locking 0.72
is_power_steering 0.98
is_driver_seat_height_adjustable 0.58
is_day_night_rear_view_mirror 0.38
is_ecw 0.72
is_speed_alert 0.99
ncap_rating 1.76
is_claim 0.00
mean

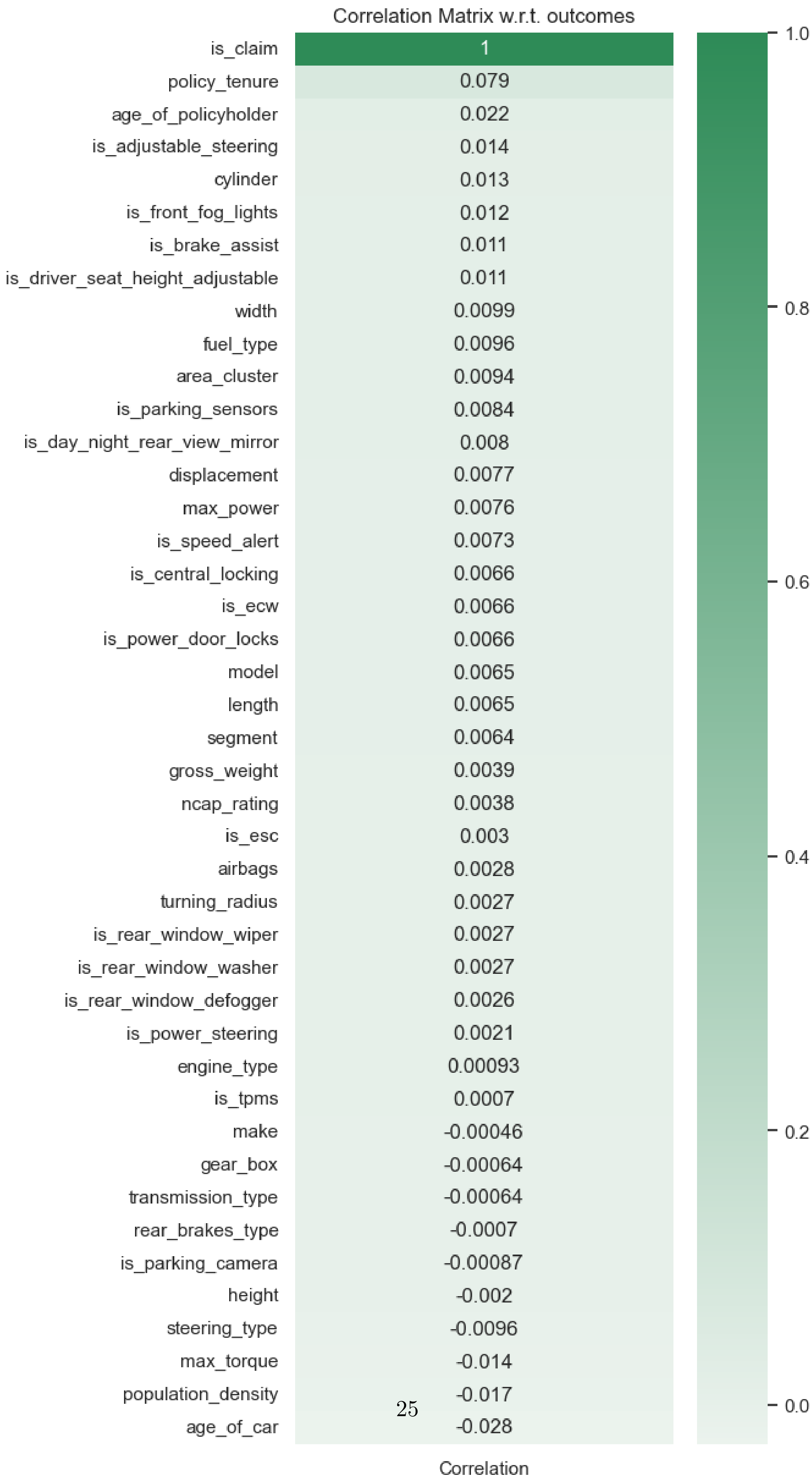
- **Observing the correlation of each of the features in the dataset using heatmap**

[20]:
```
# Correlation of the dataset
sns.heatmap(df.corr(),cmap=sns.color_palette("vlag", as_cmap=True),figure = plt.
 ↪figure(figsize=(20,15)))
plt.show()
```



- **Correlation of independent features vs dependent features**

[21]:
```
corr = df.corrwith(df['is_claim']).sort_values(ascending = False).to_frame()
corr.columns = ['Correlation']
plt.subplots(figsize = (5,15))
sns.heatmap(corr, annot=True, cmap=sns.light_palette('seagreen', as_cmap=True))
plt.title("Correlation Matrix w.r.t. outcomes")
plt.show()
```


Correlation Matrix w.r.t. outcomes
is_claim 1
policy_tenure 0.079
age_of_policyholder 0.022
is_adjustable_steering 0.014
cylinder 0.013
is_front_fog_lights 0.012
is_brake_assist 0.011
is_driver_seat_height_adjustable 0.011
width 0.0099
fuel_type 0.0096
area_cluster 0.0094
is_parking_sensors 0.0084
is_day_night_rear_view_mirror 0.008
displacement 0.0077
max_power 0.0076
is_speed_alert 0.0073
is_central_locking 0.0066
is_ecw 0.0066
is_power_door_locks 0.0066
model 0.0065
length 0.0065
segment 0.0064
gross_weight 0.0039
ncap_rating 0.0038
is_esc 0.003
airbags 0.0028
turning_radius 0.0027
is_rear_window_wiper 0.0027
is_rear_window_washer 0.0027
is_rear_window_defogger 0.0026
is_power_steering 0.0021
engine_type 0.00093
is_tpms 0.0007
make -0.00046
gear_box -0.00064
transmission_type -0.00064
rear_brakes_type -0.0007
is_parking_camera -0.00087
height -0.002
steering_type -0.0096
max_torque -0.014
population_density -0.017
age_of_car -0.028
Correlation
1.0
0.8
0.6
0.4
0.2
0.0

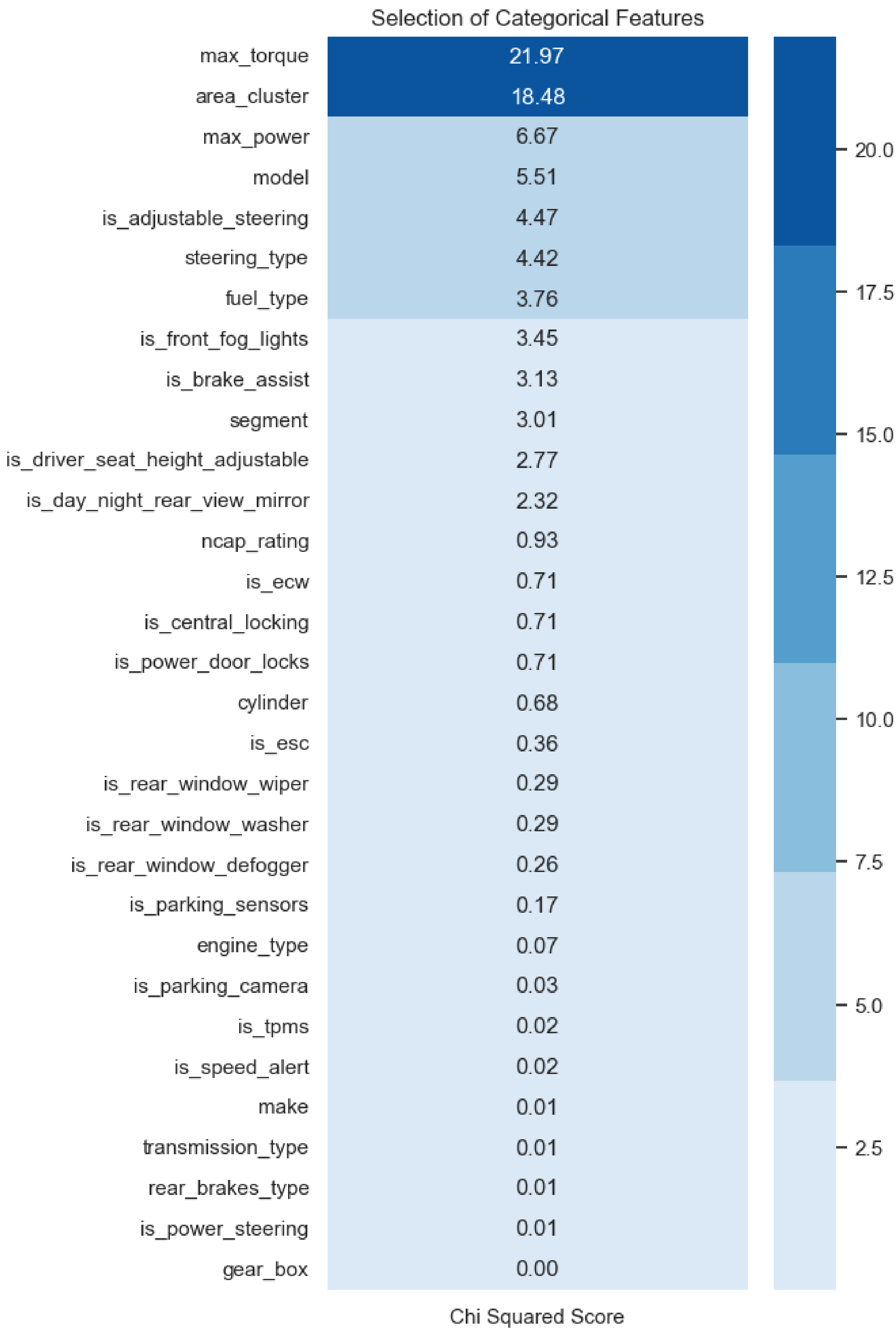
7 4. Feature Selection/Extraction

7.0.1 Feature Importance of categorical fields using Chi-Square method

- Here we are segregating categorical features that are heavily impacting the Claims (dependent variable) and will remove features that are least impacting.

[22]:
```
# Chi-Square Test
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, mutual_info_classif
```

[23]:
```
# Heatmap of categorical features
best_features = SelectKBest(score_func = chi2, k='all')
fit = best_features.fit(df[cf], df[target])

featureScores_1 = pd.DataFrame(data = fit.scores_, index= fit.feature_names_in_,
                             columns = ['Chi Squared Score'])

plt.subplots(figsize=(5,12))
sns.heatmap(featureScores_1.sort_values(ascending = False, by = 'Chi Squared␣
 ↪Score'),
            annot=True, cmap = sns.color_palette("Blues"), fmt = '.2f');
plt.title('Selection of Categorical Features')
plt.show()
```


Selection of Categorical Features
max_torque
area_cluster
max_power
model
is_adjustable_steering
steering_type
fuel_type
is_front_fog_lights
is_brake_assist
segment
is_driver_seat_height_adjustable
is_day_night_rear_view_mirror
ncap_rating
is_ecw
is_central_locking
is_power_door_locks
cylinder
is_esc
is_rear_window_wiper
is_rear_window_washer
is_rear_window_defogger
is_parking_sensors
engine_type
is_parking_camera
is_tpms
is_speed_alert
make
transmission_type
rear_brakes_type
is_power_steering
gear_box
21.97
18.48
6.67
5.51
4.47
4.42
3.76
3.45
3.13
3.01
2.77
2.32
0.93
0.71
0.71
0.71
0.68
0.36
0.29
0.29
0.26
0.17
0.07
0.03
0.02
0.02
0.01
0.01
0.01
0.01
0.00
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
Chi Squared Score

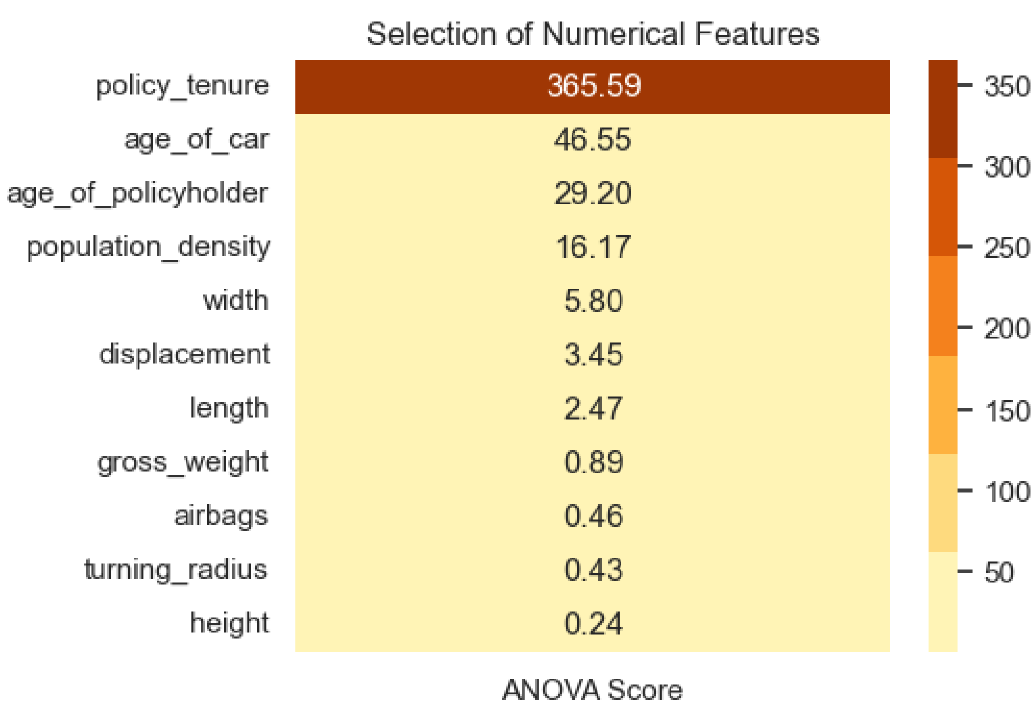
- ***From the above feature selection of categorical columns using Chi-Square test, we can observe that the features 'max_torque' and 'area_cluster' is highly impacting the Claims (dependent variable) whereas 'gear_box', 'power_steering', 'rear_brakes_type' etc. being the least important.***

7.0.2 Feature Importance of numerical fields using ANOVA testing

- Here we are segregating numerical features that are heavily impacting the Claims (dependent variable) and will remove features that are least impacting.

[24]:
```
# Imporing ANOVA feature selection
from sklearn.feature_selection import f_classif
```

[25]:
```
# Heatmap of numerical features
best_features = SelectKBest(score_func = f_classif, k='all')
fit = best_features.fit(df[nf], df[target])

featureScores_2 = pd.DataFrame(data = fit.scores_, index= fit.feature_names_in_,
                               columns = ['ANOVA Score'])

plt.subplots(figsize=(5,4))
sns.heatmap(featureScores_2.sort_values(ascending = False, by = 'ANOVA Score'),
            annot=True, cmap = sns.color_palette("YlOrBr"), fmt = '.2f');
plt.title('Selection of Numerical Features')
plt.show()
```



- ***From the above feature selection of numerical columns using ANOVA test, we can observe that the feature 'policy_tenure' is highly impacting the Claims (dependent variable) whereas 'height', 'turning_radius', 'airbags' etc. being the least important.***

[26]:
```
# Removing less important independent variables
df.drop(columns=featureScores_1[featureScores_1['Chi Squared Score']<0.1].
 ↪index,inplace=True)
df.drop(columns=featureScores_2[featureScores_2['ANOVA Score']<1].
 ↪index,inplace=True)
```

[27]:
```
df['is_claim'].value_counts()
```

[27]:
```
is_claim
0    54844
1     3748
Name: count, dtype: int64
```

- **In the above analysis, we understood that our dataset is imbalanced and hence we fixed the same using SMOTE technique.**

  In Layman's terms - SMOTE technique is used to balance the number of counts for both the classes (0 and 1). Hence in the below cell we can find out that the data is now balanced.

[28]:
```
# Handling imbalance dataset
from imblearn.over_sampling import SMOTE
smote = SMOTE()
f1 = df.iloc[:,:-1]
t1 = df.iloc[:,-1]
f1,t1 = smote.fit_resample(f1,t1)
```

8 5. Predictive Modelling

[29]:
```
# Model Building
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪RepeatedStratifiedKFold
from sklearn.metrics import confusion_matrix,classification_report,␣
 ↪accuracy_score, roc_auc_score, roc_curve,␣
 ↪precision_recall_curve,RocCurveDisplay, auc
```

[30]:
```
x_train, x_test, y_train, y_test = train_test_split(f1,t1, test_size=0.2,␣
 ↪random_state=101)
```

[46]:
```
def model_eval(*classifier):
    for estimator in classifier:
```

```
        print()
        print('**********************',str(type(estimator)).split('.')[-1][:
↪-2],'*******************')
        print(' ')
        estimator.fit(x_train, y_train)
        cm = confusion_matrix(y_test,estimator.predict(x_test))
        names= ['True Negative','False Positive','False Negative','True␣
↪Positive']
        counts = [value for value in cm.flatten()]
        percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.
↪sum(cm)]
        labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names, counts,␣
↪percentages)]
        labels = np.asarray(labels).reshape(2,2)
        sns.heatmap(cm, annot=labels, cmap='Blues',fmt='')
        print(classification_report(y_test,estimator.predict(x_test)))
        plt.show()
```

-

[32]:
```
# RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
Random_Forest = RandomForestClassifier(n_estimators=300,␣
↪min_samples_split=5,min_samples_leaf=4,
                          max_depth=5000,␣
↪bootstrap=True,max_features='sqrt',criterion="entropy",random_state=5)
```

[33]:
```
# XGBoost Classifier
from xgboost import XGBClassifier
XGBoost = XGBClassifier(learning_rate=0.01,n_estimators=500)
```

[34]:
```
# LightGBM Classifier
from lightgbm import LGBMClassifier
LightGBM = LGBMClassifier(learning_rate=0.01,␣
↪n_estimators=500,force_col_wise=True)
```

[35]:
```
# Naive Bayes Classifier
from sklearn.naive_bayes import GaussianNB
Gaussian_Naive_Bayes = GaussianNB()
```

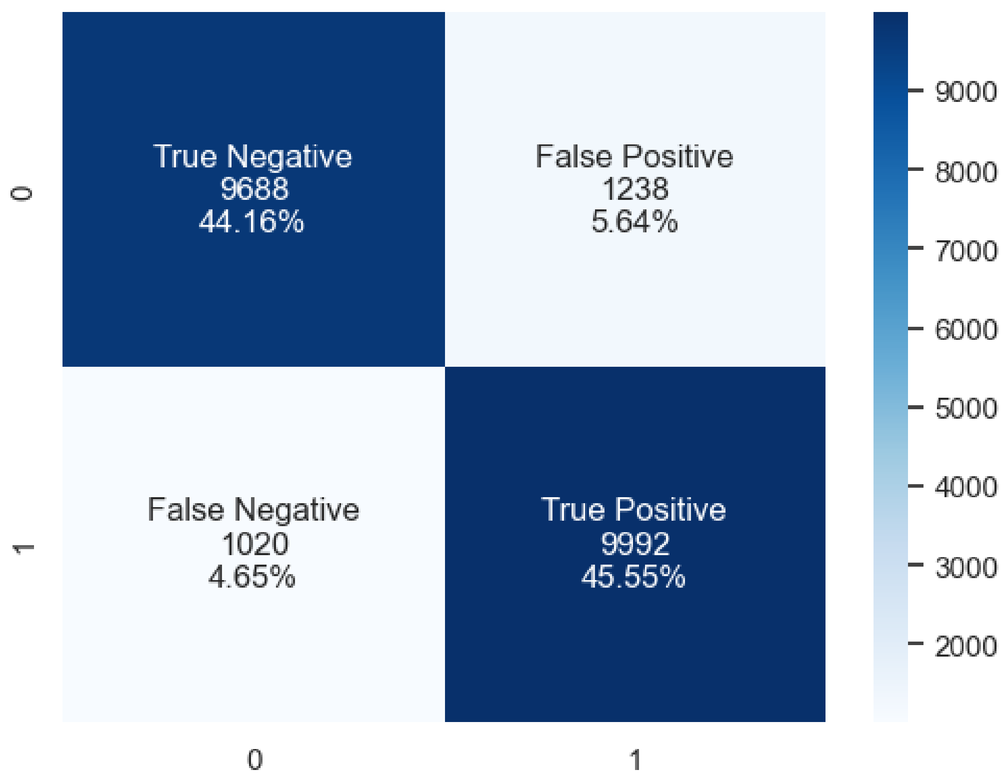
[36]:
```
# CatBoost Classifier
from catboost import CatBoostClassifier
Cat_Boost = CatBoostClassifier(learning_rate=0.
↪01,eval_metric='AUC',logging_level='Silent')
```

[37]:
```
from sklearn.svm import SVC
SVM_Classifier = SVC()
```

[47]:

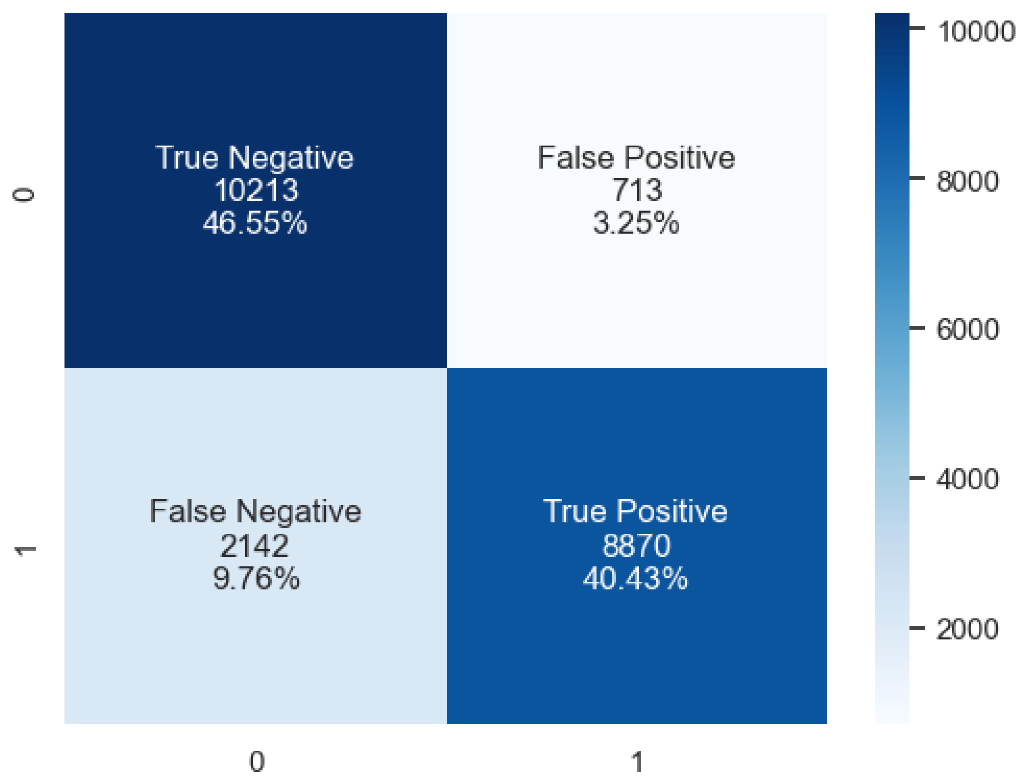
```
model_eval(Random_Forest,XGBoost,LightGBM,Gaussian_Naive_Bayes,Cat_Boost)
```

```
********************* RandomForestClassifier ******************

              precision    recall  f1-score   support

           0       0.90      0.89      0.90     10926
           1       0.89      0.91      0.90     11012

    accuracy                           0.90     21938
   macro avg       0.90      0.90      0.90     21938
weighted avg       0.90      0.90      0.90     21938
```



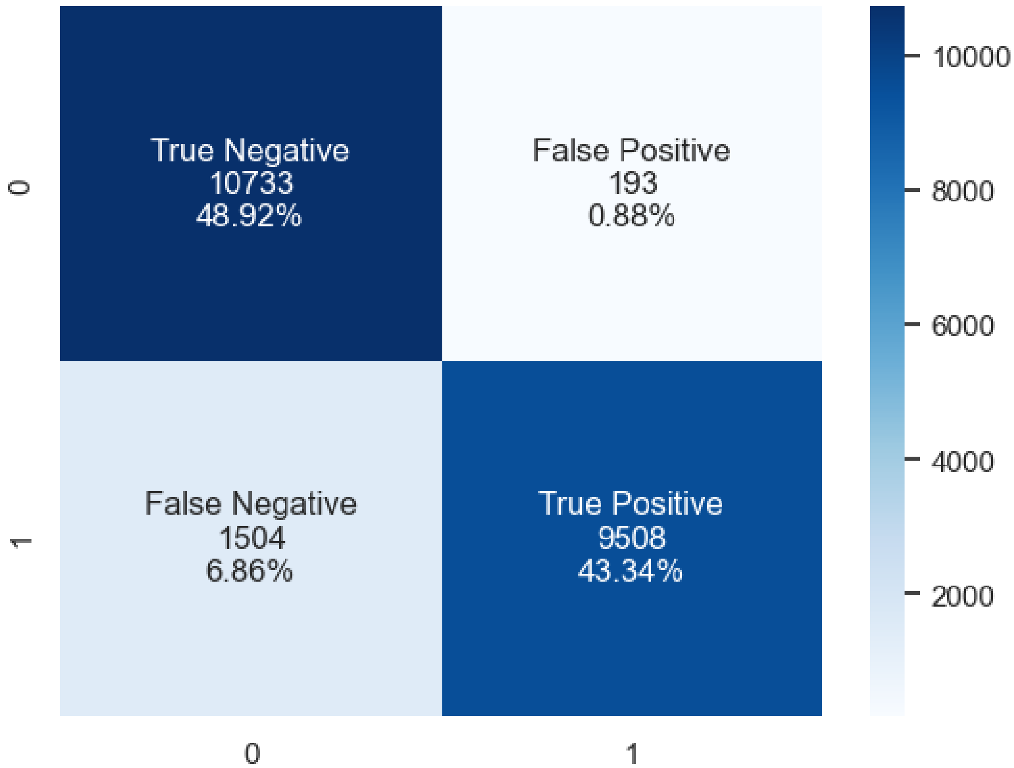

```
********************* XGBClassifier ******************
```

	precision	recall	f1-score	support
0	0.83	0.93	0.88	10926
1	0.93	0.81	0.86	11012
accuracy			0.87	21938
macro avg	0.88	0.87	0.87	21938
weighted avg	0.88	0.87	0.87	21938



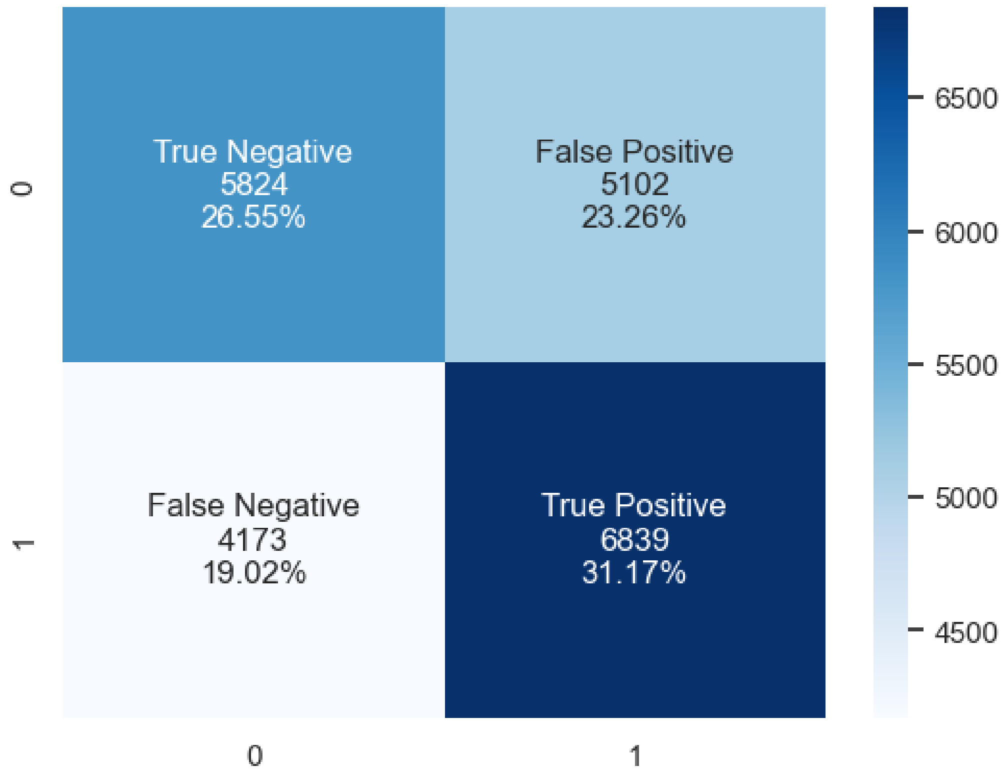

```
********************** LGBMClassifier ******************

[LightGBM] [Info] Number of positive: 43832, number of negative: 43918
[LightGBM] [Info] Total Bins 1162
[LightGBM] [Info] Number of data points in the train set: 87750, number of used
features: 29
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499510 -> initscore=-0.001960
[LightGBM] [Info] Start training from score -0.001960
              precision    recall  f1-score   support
```

0	0.88	0.98	0.93	10926
1	0.98	0.86	0.92	11012
accuracy			0.92	21938
macro avg	0.93	0.92	0.92	21938
weighted avg	0.93	0.92	0.92	21938

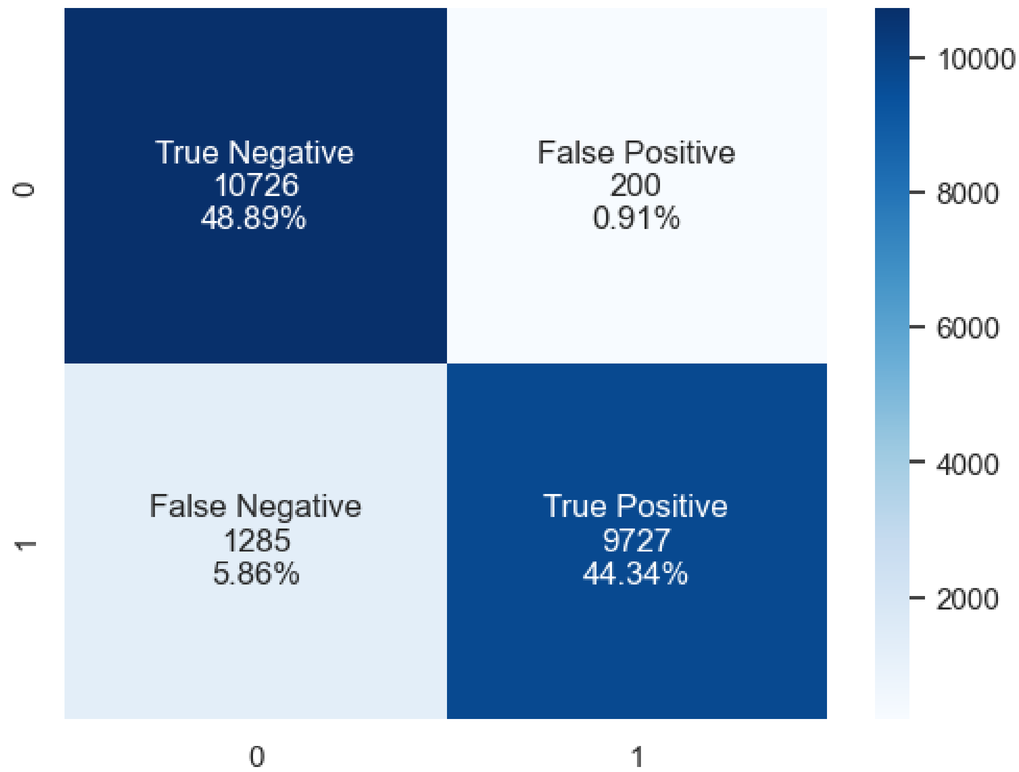



```
********************* GaussianNB ******************
```

	precision	recall	f1-score	support
0	0.58	0.53	0.56	10926
1	0.57	0.62	0.60	11012
accuracy			0.58	21938
macro avg	0.58	0.58	0.58	21938
weighted avg	0.58	0.58	0.58	21938




```
********************** CatBoostClassifier *******************

              precision    recall  f1-score   support

           0       0.89      0.98      0.94     10926
           1       0.98      0.88      0.93     11012

    accuracy                           0.93     21938
   macro avg       0.94      0.93      0.93     21938
weighted avg       0.94      0.93      0.93     21938
```



- **Above information depicts the classification report where we can establish the fact that LGBMClassifier, Cat Boosting and Random Forest Classifier technique gives us the max accuracy whereas Gaussian Naive Bayes being the least**
- Confusion matrix is represented using heatmap.

8.0.1 Fitting the dataset into the models.

[39]:
```
Random_Forest.fit(x_train, y_train)
XGBoost.fit(x_train, y_train)
LightGBM.fit(x_train, y_train)
Cat_Boost.fit(x_train,y_train)
Gaussian_Naive_Bayes.fit(x_train,y_train)
```

```
[LightGBM] [Info] Number of positive: 43832, number of negative: 43918
[LightGBM] [Info] Total Bins 1162
[LightGBM] [Info] Number of data points in the train set: 87750, number of used
features: 29
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499510 -> initscore=-0.001960
```

```
[LightGBM] [Info] Start training from score -0.001960
```

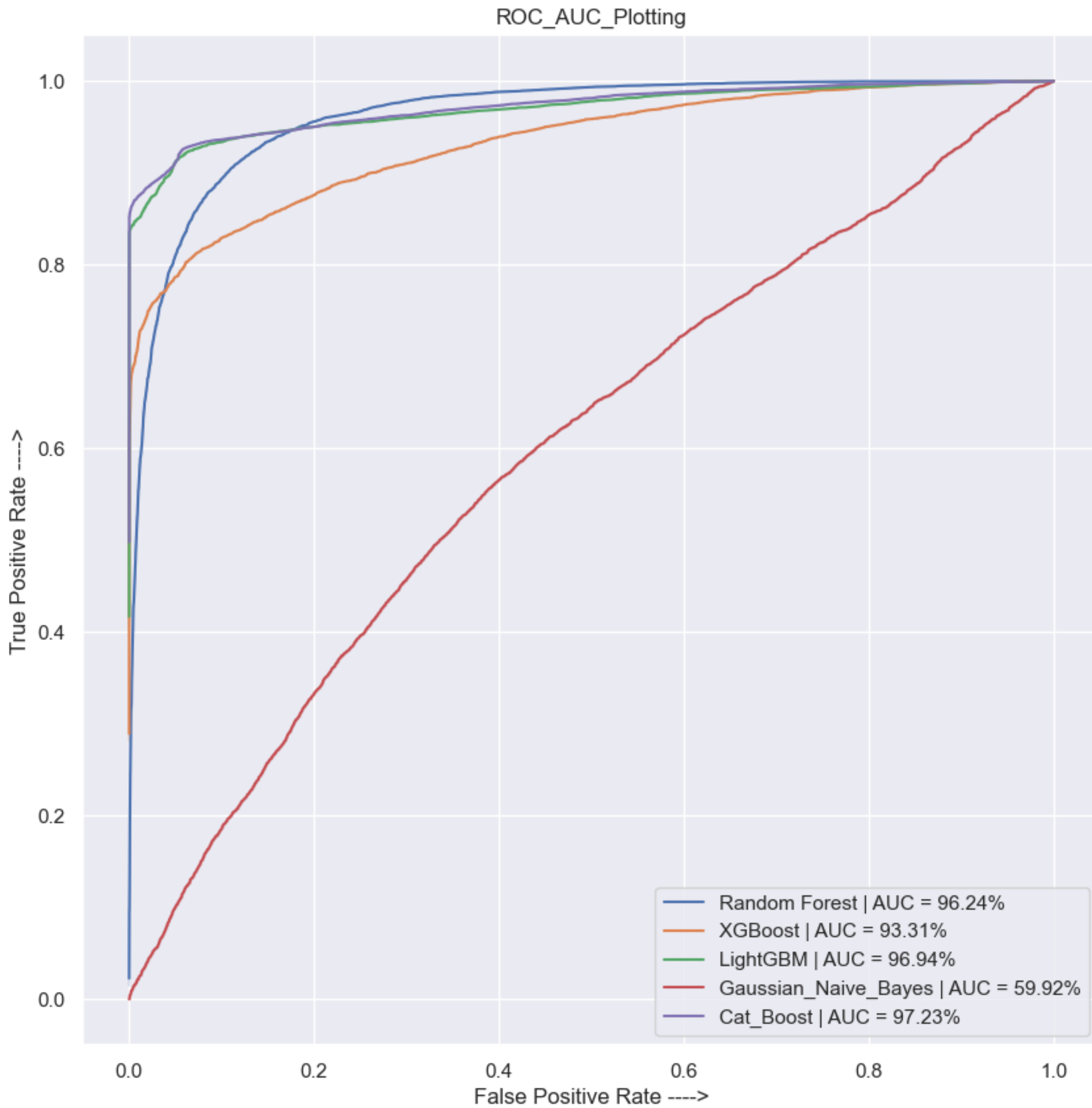
[39]:
```
GaussianNB()
```

[40]:
```
rf_probs = Random_Forest.predict_proba(x_test)[:,1]
xgb_probs = XGBoost.predict_proba(x_test)[:,1]
lgbm_probs = LightGBM.predict_proba(x_test)[:,1]
nb_probs = Gaussian_Naive_Bayes.predict_proba(x_test)[:,1]
cat_probs = Cat_Boost.predict_proba(x_test)[:,1]
```

[41]:
```
rf_auc = roc_auc_score(y_test,rf_probs) * 100
xgb_auc = roc_auc_score(y_test,xgb_probs) * 100
lgbm_auc = roc_auc_score(y_test,lgbm_probs) * 100
nb_auc = roc_auc_score(y_test,nb_probs) * 100
cat_auc = roc_auc_score(y_test,cat_probs) * 100
```

[42]:
```
rf_fpr,rf_tpr,rf_threshold = roc_curve(y_test,rf_probs)
xgb_fpr,xgb_tpr,xgb_threshold = roc_curve(y_test,xgb_probs)
lgbm_fpr,lgbm_tpr,lgbm_threshold = roc_curve(y_test,lgbm_probs)
nb_fpr,nb_tpr,nb_threshold = roc_curve(y_test,nb_probs)
cat_fpr,cat_tpr,cat_threshold = roc_curve(y_test,cat_probs)
```

[43]:
```
plt.figure(figsize=(10,10))
sns.lineplot(x=rf_fpr,y=rf_tpr,label='Random Forest | AUC = {:.2f}%'.
 ↪format(rf_auc))
sns.lineplot(x=xgb_fpr,y=xgb_tpr,label='XGBoost | AUC = {:.2f}%'.
 ↪format(xgb_auc))
sns.lineplot(x=lgbm_fpr,y=lgbm_tpr,label='LightGBM | AUC = {:.2f}%'.
 ↪format(lgbm_auc))
sns.lineplot(x=nb_fpr,y=nb_tpr,label='Gaussian_Naive_Bayes | AUC = {:.2f}%'.
 ↪format(nb_auc))
sns.lineplot(x=cat_fpr,y=cat_tpr,label='Cat_Boost | AUC = {:.2f}%'.
 ↪format(cat_auc))
plt.xlabel("False Positive Rate ---->")
plt.ylabel("True Positive Rate ---->")
plt.title("ROC_AUC_Plotting")
plt.legend()
plt.show()
```



9 What we started with?

- We started with a sample of dataset related to insurance industry where the problem statement was to classify the customers who will claim for insurance based on numerical and categorical features. Along with we also had to explore the dataset and on analyze whether the data is imbalanced. We had certain dataset attributes where each features described its dependency on the final outcome.

10 What we observed and course of actions initiated?

- The very first step involved data ingestion and a clear understanding of metadata, that is knowing data about the data where I get to know the dataset attributes, it's outline analysis, count of total records, finding null/missing values, understanding features and segregating independent variable to target dependency. Formal proceedings included data cleaning where importing of basic analytical and visualisation libraries (like numpy, pandas, seaborn, etc). We imported the dataset into Jupyter notebook and check for the missing values. Dropped missing data where necessary. Encoded the categorical columns using label encoder and scaled the numerical columns. I also checked for any imbalances in the dataset, fixed using oversampling technique called SMOTE and outliers were treated using outlier treatment. Correlations were drawn out with each of the independent features with respect to Churn data (dependent column) and using sklearn's feature importance technique I able to drop the least important features for better model prediction.

11 Conclusion

- Here I have used 5 of the machine learning classifiers — RandomForest, XGBoosting, LGBM-Classifier, Cat Boosting and Gaussian Naive Bayes out of which CatBoosting Classifier proved to provide the best accuracy score. We can see the same in the ROC_AUC curve where the area under the same is maximum, that is 97.20%.