Orion Joyner

Professor Alvarez

CSC 461/ Final Report

26th December 2022

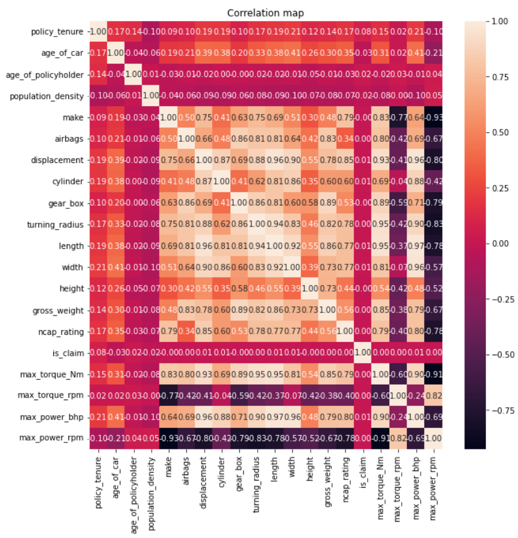
Final Report

Introduction:

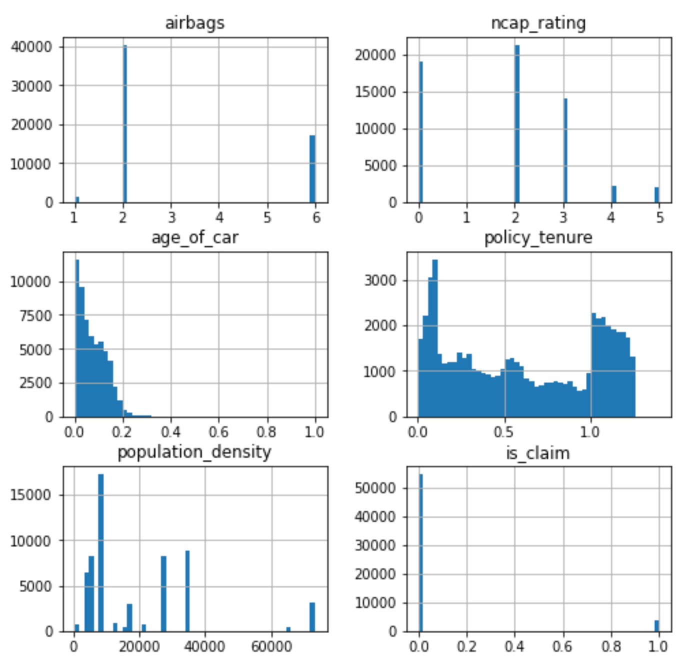
This project is a study of predicting car insurance claims in the next six months within machine learning algorithms. The dataset was taken from Kaggle.com and was used in a competition on the website. The name of the dataset is Car Insurance Claim Prediction The objective of using this dataset is to predict what factors contribute most to the policy holder's claim and if we are able to predict it ourselves. It is made up of a record of 10,302 observations and 27 variables. First, I had to analyze the dataset and become more familiar with it. This step involved looking at the various features, checking for null values, and performing some preprocessing. I constructed one base model and one model in order to compare accuracies. Finally, the model we use was submitted on kaggle in order to measure its best accuracy/precision.

EDA:

As said before, in order to understand the data set completely, I need to use preprocessing and analyze different features. The training data was imported using the pandas library. The dataset has 15120 instances and 55 attributes. 54 of those attributes are features of the forest areas while the last attribute is the forest cover type or label. In order to analysis this more, a correlation matrix was made. Correlations between the non-categorical variables were analyzed and plotted in the figure below. All of these strong correlations seem to be slightly higher than 0.75 in magnitude.



By also cleaning up the data, we were able to determine the six main contributing factors to the dataset itself. The figure is shown below for reference.



Source Code:

The following listed is the actual source code needed to represent our approach to this problem.

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import io

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import roc_auc_score, accuracy_score, f1_score,
recall_score, precision_score
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingRandomSearchCV
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.pipeline import make_pipeline
from google.colab import files

from imblearn.over_sampling import RandomOverSampler

from scipy.stats import randint
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Gathering the Data

In [68]:

```
data = files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session.

Please rerun this cell to enable.

Saving test.csv.zip to test.csv (1).zip

Saving train.csv.zip to train.csv (1).zip

In [69]:

```
train=pd.read_csv("train.csv.zip")

test=pd.read_csv("test.csv.zip")
```

In [70]:

```
pd.set_option('display.max_rows',100)

pd.set_option('display.max_columns',100)
```

EDA(Data Analysis)

[redacted]

```
train.shape
```

[redacted]

```
test.shape
```

[redacted]

```
train.head()
```

[redacted]

policy_id	policy_tenure	age_of_car	age_of_policyhold	area_cluster	population_densi	make	segment	model	fuel_type	max_torque	max_power	engine_type	airbags	is_esc	is_adjustable_steer	is_tpms	is_parking_senso	is_parking_camera	rear_brakes_type	displacement	cylinder	transmission_type	gear_box	steering_type	turning_radius	length	width	height	gross_weight	is_front_fog_ligh	is_rear_window_wip	is_rear_window_wash	is_rear_window_defo	is_brake_assist	is_power_door_loc	is_central_locki	is_power_steering	is_driver_seat_height_adjustable	is_day_night_rear_view_m

				er		ty										ing		rs													ts	er	er	gger		ks	ng			irror
0	ID00001	0.515874	0.05	0.644231	C1	4990	1	A	M1	CNG	60Nm@3500rpm	40.36bhp@6000rpm	F8D Petrol Engine	2	No	No	No	Yes	No	Drum	796	3	Manual	5	Power	4.6	3445	1515	1475	1185	No	No	No	No	No	No	No	Yes	No	No
1	ID00002	0.672619	0.02	0.375000	C2	27003	1	A	M1	CNG	60Nm@3500rpm	40.36bhp@6000rpm	F8D Petrol Engine	2	No	No	No	Yes	No	Drum	796	3	Manual	5	Power	4.6	3445	1515	1475	1185	No	No	No	No	No	No	No	Yes	No	No
2	ID00003	0.841110	0.02	0.384615	C3	4076	1	A	M1	CNG	60Nm@3500r	40.36bhp@6	F8D PetrolE	2	No	No	No	Yes	No	Drum	796	3	Manual	5	Power	4.6	3445	1515	1475	1185	No	No	No	No	No	No	No	Yes	No	No

											pm	000rpm	ngine																											
3	ID00004	0.900277	0.11	0.432692	C4	21622	1	C1	M2	Petrol	113Nm@4400rpm	88.50bhp@6000rpm	1.2 L K12N Dualjet	2	Yes	Yes	No	Yes	Yes	Drum	1197	4	Automatic	5	Electric	4.8	3995	1735	1515	1335	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	ID00005	0.596403	0.11	0.634615	C5	34738	2	A	M3	Petrol	91Nm@4250rpm	67.06bhp@5500rpm	1.0 SCe	2	No	No	No	No	Yes	Drum	999	3	Automatic	5	Electric	5.0	3731	1579	1490	1155	No	No	No	No	No	Yes	Yes	Yes	No	Yes

In [74]:

```
test.head()
```

Out[74]:

	policy_id	policy_tenure	age_of_car	age_of_policyholder	area_cluster	population_density	make	segment	model	fuel_type	max_torque	max_power	engine_type	airbags	is_esc	is_adjustable_steering	is_tpms	is_parking_sensors	is_parking_camera	rear_brakes_type	displacement	cylinder	transmission_type	gear_box	steering_type	turning_radius	length	width	height	gross_weight	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
0	ID585933	0.341732	0.00	0.586538	C3	4076	1	A	M1	CNG	60Nm@3500rpm	40.36bhp@6000rpm	F8D Petrol Engine	2	No	No	No	Yes	No	Drum	796	3	Manual	5	Power	4.6	3445	1515	1475	1185	No	No	No	No	No	No	No	Yes	No	No
1	ID585944	0.307241	0.13	0.442308	C8	8794	1	B2	M6	Petrol	113Nm@440	88.50bhp@	K Series Du	2	No	Yes	No	Yes	No	Drum	1197	4	Manual	5	Electric	4.8	3845	1735	1530	1335	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes

											0rpm	6000rpm	aljet																											
2	ID58595	0.327924	0.12	0.451923	C8	8794	2	A	M3	Petrol	91Nm@4250rpm	67.06bhp@5500rpm	1.0SCe	2	No	No	No	No	Yes	Drum	999	3	Automatic	5	Electric	5.0	3731	1579	1490	1155	No	No	No	No	No	Yes	Yes	Yes	No	Yes
3	ID58596	0.782654	0.01	0.461538	C5	34738	1	A	M1	CNG	60Nm@3500rpm	40.36bhp@6000rpm	F8DPetrolEngine	2	No	No	No	Yes	No	Drum	796	3	Manual	5	Power	4.6	3445	1515	1475	1185	No	No	No	No	No	No	No	Yes	No	No
4	ID585	1.2334	0.02	0.6346	C5	34738	1	A	M1	CNG	60Nm@35	40.36bh	F8DPetr	2	No	No	No	Yes	No	Drum	796	3	Manual	5	Power	4.6	3445	1515	1475	1185	No	No	No	No	No	No	No	Yes	No	No

9 0 1 0 p o
7 4 5 0 @ l
r 6 E
p 0 n
m 0 g
0 i
r n
p e
m

[redacted]

```
train.info()
```

[redacted]

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

29 gross_weight   58592 non-null int64

30 is_front_fog_lights   58592 non-null object

31 is_rear_window_wiper   58592 non-null object

32 is_rear_window_washer   58592 non-null object

33 is_rear_window_defogger   58592 non-null object

34 is_brake_assist   58592 non-null object

35 is_power_door_locks   58592 non-null object

36 is_central_locking   58592 non-null object

37 is_power_steering   58592 non-null object

38 is_driver_seat_height_adjustable 58592 non-null object

39 is_day_night_rear_view_mirror   58592 non-null object

40 is_ecw   58592 non-null object

41 is_speed_alert   58592 non-null object

42 ncap_rating   58592 non-null int64

43 is_claim   58592 non-null int64

dtypes: float64(4), int64(12), object(28)

memory usage: 19.7+ MB

[redacted]

```
test.info()
```

[redacted]

RangeIndex: 39063 entries, 0 to 39062

Data columns (total 43 columns):

Column   Non-Null Count Dtype

--- ------   -------------- -----

0   policy_id   39063 non-null object

1   policy_tenure   39063 non-null float64

2   age_of_car   39063 non-null float64

3   age_of_policyholder   39063 non-null float64

4   area_cluster   39063 non-null object

5   population_density   39063 non-null int64

6   make   39063 non-null int64

7   segment   39063 non-null object

8   model   39063 non-null object

9   fuel_type   39063 non-null object

10 max_torque   39063 non-null object

11 max_power   39063 non-null object

12	engine_type	39063	non-null	object
13	airbags	39063	non-null	int64
14	is_esc	39063	non-null	object
15	is_adjustable_steering	39063	non-null	object
16	is_tpms	39063	non-null	object
17	is_parking_sensors	39063	non-null	object
18	is_parking_camera	39063	non-null	object
19	rear_brakes_type	39063	non-null	object
20	displacement	39063	non-null	int64
21	cylinder	39063	non-null	int64
22	transmission_type	39063	non-null	object
23	gear_box	39063	non-null	int64
24	steering_type	39063	non-null	object
25	turning_radius	39063	non-null	float64
26	length	39063	non-null	int64
27	width	39063	non-null	int64
28	height	39063	non-null	int64
29	gross_weight	39063	non-null	int64
30	is_front_fog_lights	39063	non-null	object
31	is_rear_window_wiper	39063	non-null	object
32	is_rear_window_washer	39063	non-null	object
33	is_rear_window_defogger	39063	non-null	object
34	is_brake_assist	39063	non-null	object
35	is_power_door_locks	39063	non-null	object
36	is_central_locking	39063	non-null	object
37	is_power_steering	39063	non-null	object
38	is_driver_seat_height_adjustable	39063	non-null	object

39 is_day_night_rear_view_mirror   39063 non-null object

40 is_ecw   39063 non-null object

41 is_speed_alert   39063 non-null object

42 ncap_rating   39063 non-null int64

dtypes: float64(4), int64(11), object(28)

memory usage: 12.8+ MB

[redacted]

```
#This shows the data set is heavily imbalanced
train['is_claim'].value_counts()/train.shape[0] * 100
```

[redacted]

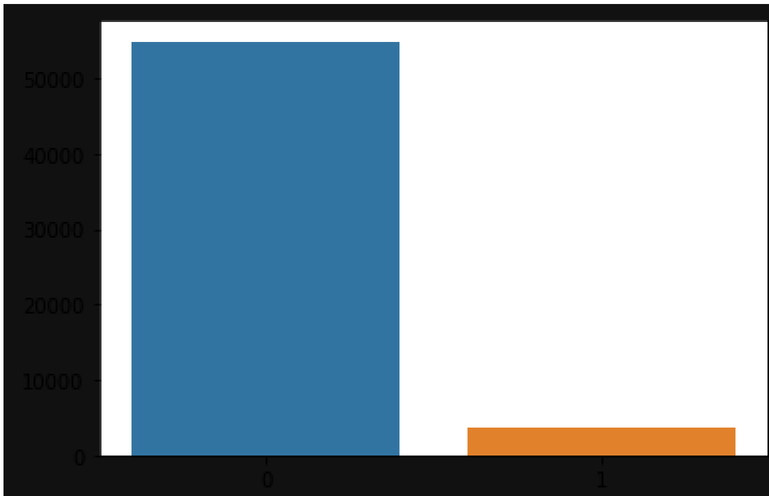
0   93.603222

1   6.396778

[redacted]

```
sns.barplot(x = [0, 1], y = [54844, 3784])
```

[redacted]



In [79]:

```
train["policy_id"].duplicated().sum()
```

Out[79]:

0

In [81]:

```
data = train.set_index("policy_id")
```

In [82]:

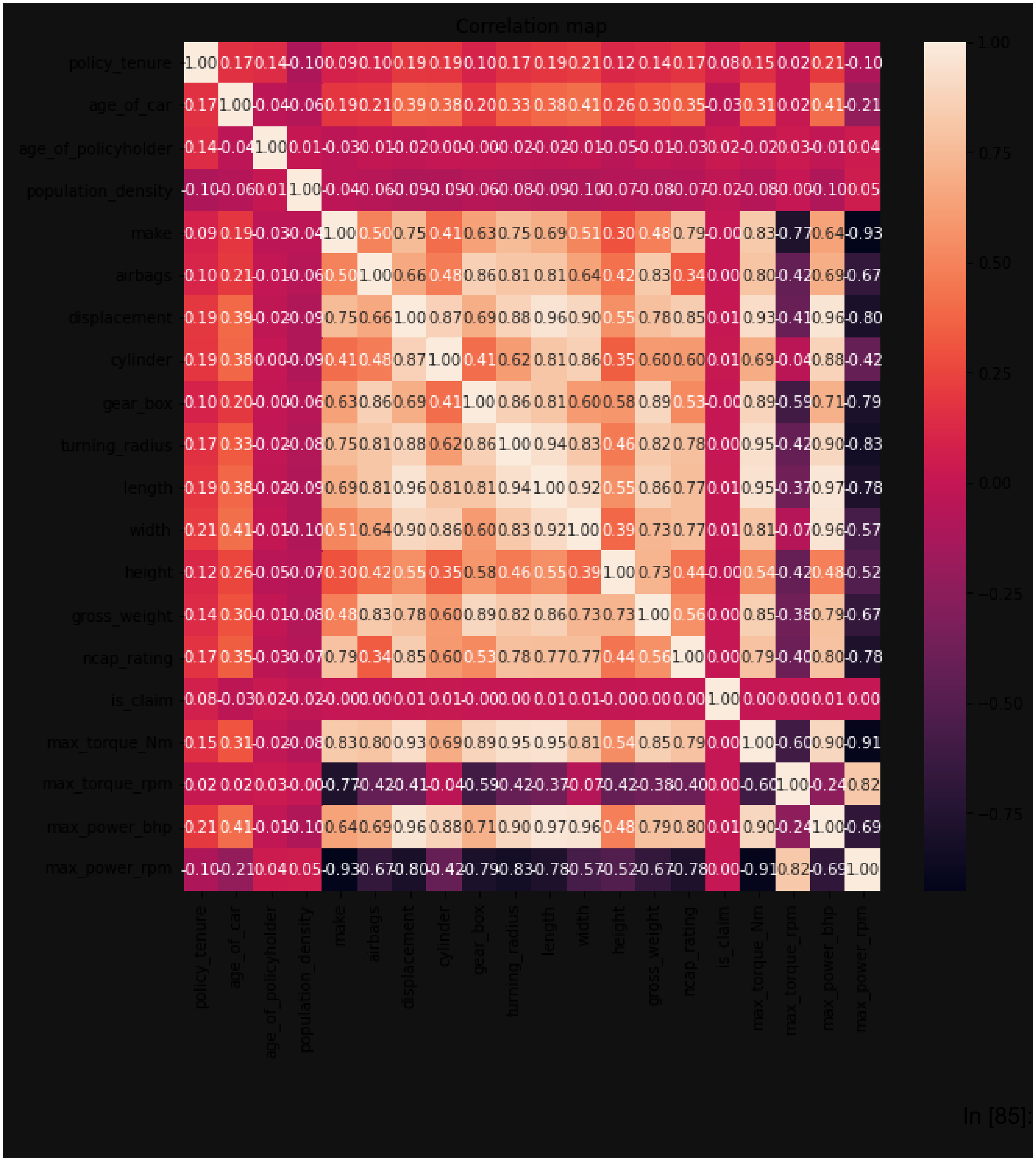
```
def data_solver_strings(data):

    data["max_torque_Nm"] =
data["max_torque"].str.extract(r"([-+]?[0-9]*\.?[0-9]+)(?=\s*Nm)").astype(
'float64')
```

```
    data["max_torque_rpm"] =
data["max_torque"].str.extract(r"([-+]?[0-9]*\.?[0-9]+)(?=\s*rpm)").astype
('float64')

    data["max_power_bhp"] =
data["max_power"].str.extract(r"([-+]?[0-9]*\.?[0-9]+)(?=\s*bhp)").astype(
'float64')
    data["max_power_rpm"] =
data["max_power"].str.extract(r"([-+]?[0-9]*\.?[0-9]+)(?=\s*rpm)").astype(
'float64')

    data.drop(["max_torque","max_power"], axis=1, inplace=True)

data_solver_strings(data)
```

[redacted]

```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(data.corr(), annot=True, fmt=".2f", ax=ax )
plt.title("Correlation map")
plt.show()
```

[redacted]




```
# TARGET  BALANCE

train["is_claim"].value_counts(normalize=True).plot(kind="bar")

plt.ylabel("Relative Frequency")

plt.xlabel("Target")
```

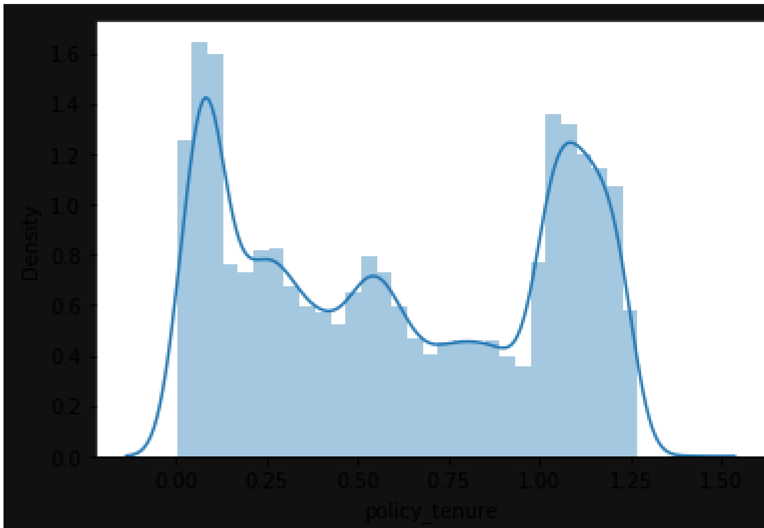
```
plt.title("Target Balance");
```

In [105]:

```
# checking for outliers
sns.distplot(train["policy_tenure"]);
```

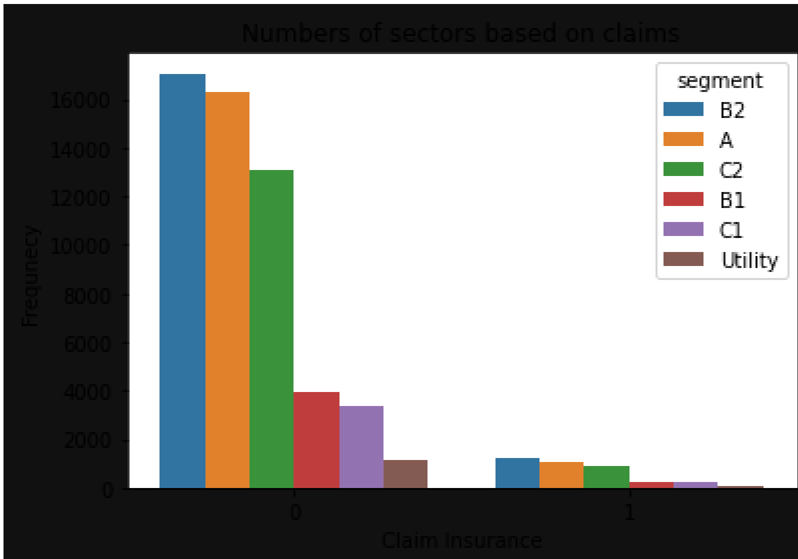
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

  warnings.warn(msg, FutureWarning)



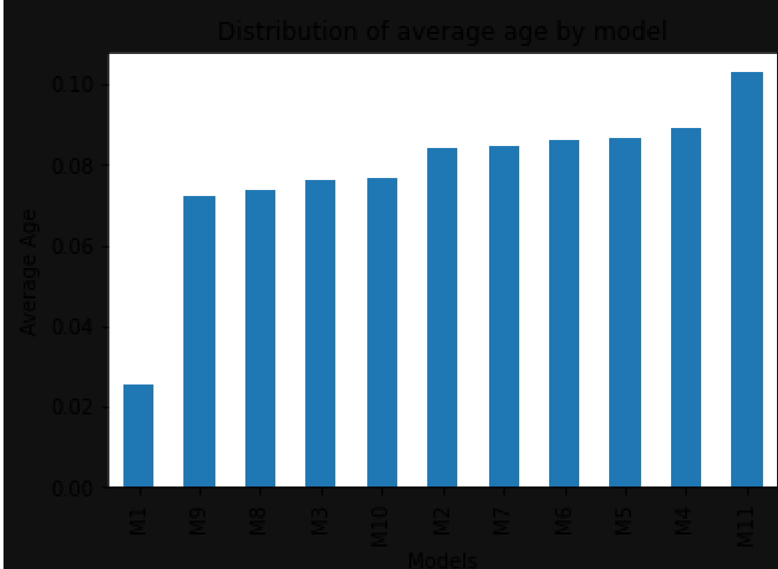
In [86]:

```
# Numbers of segments based on cliams


ffg =
train.groupby("is_claim")["segment"].value_counts().rename("Frequency").to
_frame().reset_index()
sns.barplot(x="is_claim",
            y="Frequency",
            hue="segment",
            data=ffg)
plt.xlabel("Claim Insurance")
plt.ylabel("Frequnecy")
plt.title("Numbers of sectors based on claims");
```



In [87]:

```
## distribution of average age by model
train.groupby("model")["age_of_car"].mean().sort_values().plot(kind="bar")
plt.xlabel("Models")
plt.ylabel("Average Age")
plt.title("Distribution of average age by model");
```



[redacted]

```
target = "is_claim"
X = train.drop(target, axis=1)
y = train[target]
print(X.shape, y.shape)
```

[redacted]

(58592, 43) (58592,)

[redacted]

```
X_train,X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,
random_state=1)
```

[redacted]

```
from imblearn.over_sampling import RandomOverSampler
sampler = RandomOverSampler(random_state=1)
X_train_over, y_train_over = sampler.fit_resample(X_train,y_train)
print(X_train_over.shape, y_train_over.shape)
```

[redacted]

(87764, 43) (87764,)

[redacted]

```
model_baseline = y_train.value_counts(normalize=True).max()
model_baseline
```

[redacted]

[redacted]

```python
def vif_check(data_check):
    vif_data = pd.DataFrame()
    vif_data["feature"] = data_check.columns

    # calculating VIF for each feature
    vif_data["VIF"] = [variance_inflation_factor(data_check.values, i)
                              for i in range(len(data_check.columns))]

    return vif_data.sort_values("VIF", ascending=False).reset_index()
```

[redacted]

```python
data_check = data.select_dtypes(exclude=["object"]).drop("is_claim", 
axis=1)

#cycle which returns features having VIF < 5, or Featues with low 
correlation
while vif_check(data_check)["VIF"][0] > 5:
    data_check.drop(vif_check(data_check)["feature"][0], axis=1, 
inplace=True)
feature_vif_table = vif_check(data_check)

feature_vif_table
```

[redacted]

/usr/local/lib/python3.8/dist-packages/statsmodels/stats/outliers_influence.py:193: RuntimeWarning: divide by zero encountered in double_scalars

  vif = 1. / (1. - r_squared_i)

[redacted]

	index	feature	VIF
0	3	airbags	3.587326
1	4	ncap_rating	3.221815
2	1	age_of_car	2.779371
3	0	policy_tenure	2.698111
4	2	population_density	1.684834

[redacted]

```
data_clean = pd.concat([data.select_dtypes("object"),
                        data[list(feature_vif_table["feature"])],
                        data["is_claim"]], axis=1)
```

[redacted]

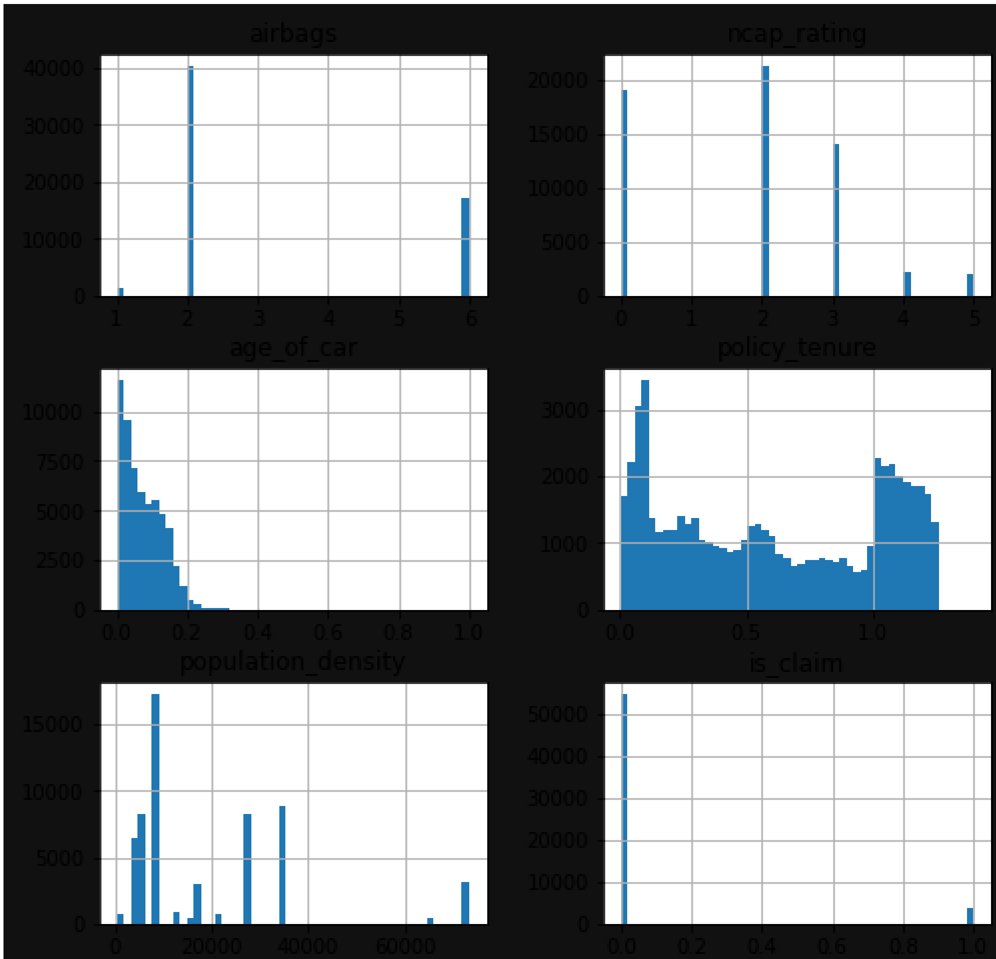
```
data_clean.describe()
```

[redacted]

	airbags	ncap_rating	age_of_car	policy_tenure	population_density	is_claim
count	58592.000000	58592.000000	58592.000000	58592.000000	58592.000000	58592.000000
mean	3.137066	1.759950	0.069424	0.611246	18826.858667	0.063968
std	1.832641	1.389576	0.056721	0.414156	17660.174792	0.244698
min	1.000000	0.000000	0.000000	0.002735	290.000000	0.000000
25%	2.000000	0.000000	0.020000	0.210250	6112.000000	0.000000
50%	2.000000	2.000000	0.060000	0.573792	8794.000000	0.000000
75%	6.000000	3.000000	0.110000	1.039104	27003.000000	0.000000
max	6.000000	5.000000	1.000000	1.396641	73430.000000	1.000000

[redacted]

```
data_clean.hist(bins=50, figsize=(8,8))
plt.show()
```



In [97]:

```
data_clean = data_clean[data_clean["age_of_car"] < 0.25]
```

Preprocessing

In [98]:

```
X = data_clean.drop("is_claim", axis=1)

y = data_clean["is_claim"]
```

```
X_train, X_valid, y_train, y_valid = train_test_split(X, y,

random_state=737, test_size=0.2, stratify=y)
```

In [99]:

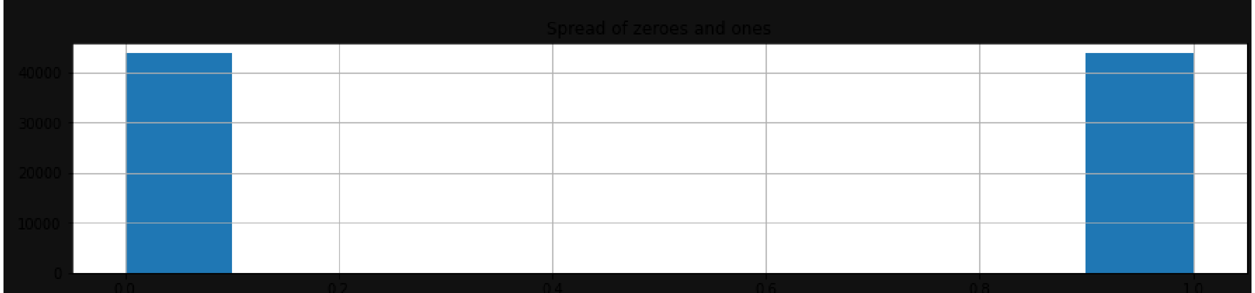
```
X_train = pd.get_dummies(X_train, drop_first=True)

X_valid = pd.get_dummies(X_valid, drop_first=True)
```

In [100]:

```
sampler = RandomOverSampler(random_state=737)

X_train_over, y_train_over = sampler.fit_resample(X_train,y_train)
```

In [101]:

```
y_train_over.hist(figsize=(15,3))

plt.title("Spread of zeroes and ones")

plt.show()
```



[redacted]

```python
dummy = DummyClassifier(strategy="stratified", random_state=737)
dummy.fit(X_train_over, y_train_over)
predicted_valid = dummy.predict(X_valid)
accuracy = accuracy_score(y_valid, predicted_valid)
ras = roc_auc_score(y_valid, dummy.predict_proba(X_valid)[:, 1])
f1 = f1_score(y_valid, predicted_valid)
recall = recall_score(y_valid, predicted_valid)
precision = precision_score(y_valid, predicted_valid)
print("Accuracy score = ", accuracy)
print("AUC-ROC score = ", ras)
print("F1 score = ", f1)
print("Recall score = ", recall)
print("Precision score = ", precision)
```

Accuracy score = 0.4964423489069867

AUC-ROC score = 0.4974766456084045

F1 score = 0.11268882175226586

Recall score = 0.49866310160427807

Precision score = 0.06352179836512262

[redacted]

```
model = RandomForestClassifier(random_state=737)
np.random.seed(737)

param_distributions = {"max_depth": randint(3,10),
                       "min_samples_split": randint(2, 9),
                       "min_samples_leaf": randint(2, 9)
                      }
search = HalvingRandomSearchCV(model, param_distributions,
resource="n_estimators",
                               max_resources=50,
                               random_state=737,
                               scoring="f1",
                               cv=5).fit(X_train_over, y_train_over)
search.best_params_
```

[redacted]

{'max_depth': 9,

'min_samples_leaf': 8,

'min_samples_split': 8,

[redacted]

```
model = RandomForestClassifier(random_state=737, **search.best_params_)
```

```
model.fit(X_train_over, y_train_over)

predicted_valid = model.predict(X_valid)

accuracy = accuracy_score(y_valid, predicted_valid)

ras = roc_auc_score(y_valid, model.predict_proba(X_valid)[:, 1])

f1 = f1_score(y_valid, predicted_valid)

recall = recall_score(y_valid, predicted_valid)

precision = precision_score(y_valid, predicted_valid)

print("Accuracy score = ", accuracy)

print("AUC-ROC score = ", ras)

print("F1 score = ", f1)

print("Recall score = ", recall)

print("Precision score = ", precision)
```

[redacted]

Accuracy score = 0.5873124732104587

AUC-ROC score = 0.6416906566268867

F1 score = 0.16481609993060375

Recall score = 0.6350267379679144

[redacted]

Conclusion:

The classifier we performed the best with an accuracy of 63%. This was a very disappointing result and was simply not expected. It is possible that if we had included more methods and algorithms into our data, we would've created a better model. It is possible that overfitting occurred even though cross-validation was used. The results obtained were not good enough to place well on the leaderboard. While the results were not as satisfactory as expected, this project was a great introduction to using different types of machine learning methods on a real world dataset. This project significantly improved their understanding of the classifiers used and their competence of implementing these models in Python. Further work could be done to better fit the data and possibly achieve better performance results. It is possible that other algorithms and methods would be better suited to this dataset than the ones used in this study. In conclusion, much was learned from performing this study but further work would have to be done in order to improve the results obtained.

References:

[1] Kaggle. (2014). Car Insurance Prediction. Available at:
https://www.kaggle.com/datasets/ifteshanajnin/carinsuranceclaimprediction-classification

[2] Kaggle. (2014). Car Insurance Prediction. Available at:
https://www.kaggle.com/code/gautamchettiar/claim-predictions-encoding-classifiers-eda

[3] RandomForestClassifer. Available at:https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.htmlhttps://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[4] Model Classification Accuracy
https://developers.google.com/machine-learning/crash-course/classification/accuracy