Insurers classification and prediction

Ironhack Sep.2022

Chahra Hamitouche

Final Project

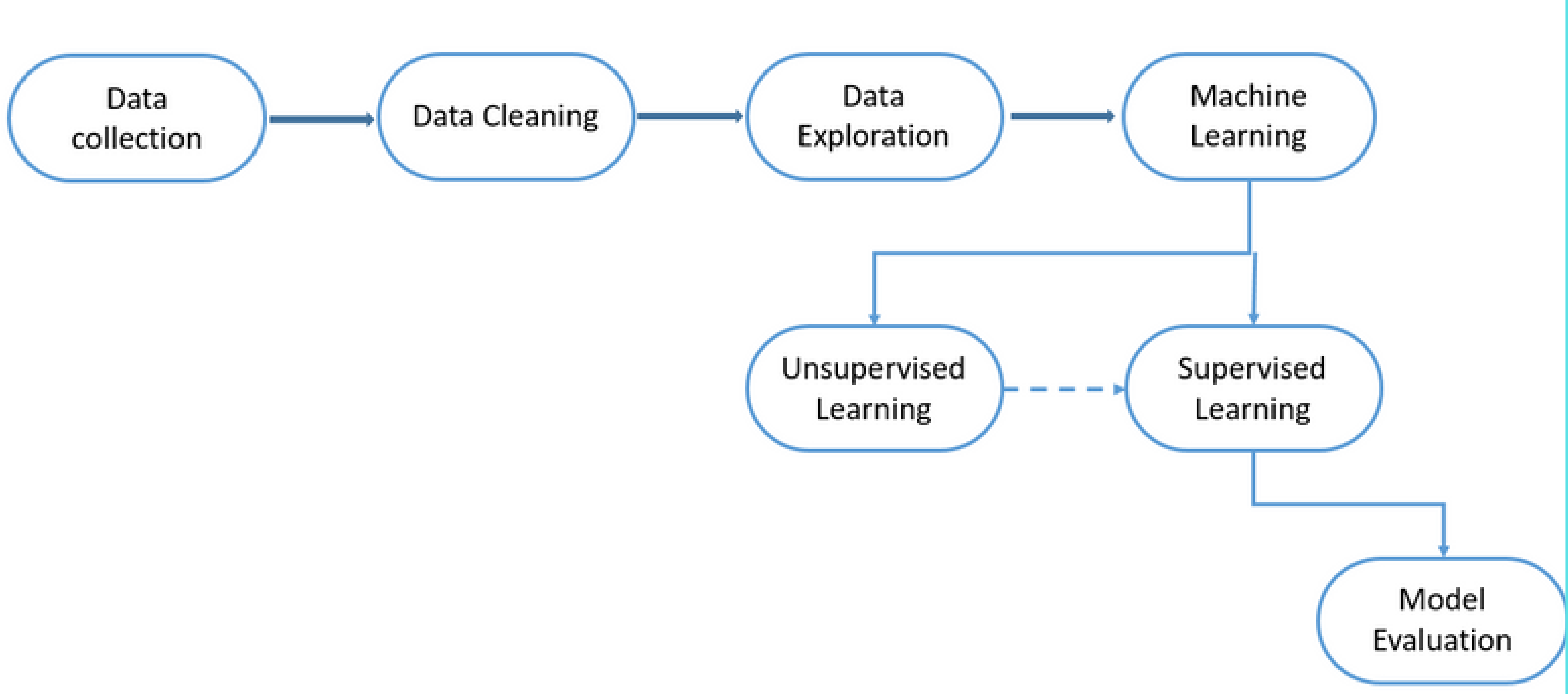
Why Insurers classification?

What we are looking for?

Understand auto insurers features

Make Predictions

How to proceed?



Data Collection

```
import pandas as pd
```

```
auto15=pd.read_excel(r'C:\Users\hamit\Desktop\Project_final\BASEAUTO.xls', sheet_name='DATA_2015')
auto16=pd.read_excel(r'C:\Users\hamit\Desktop\Project_final\BASEAUTO.xls', sheet_name='DATA_2016')
auto17=pd.read_excel(r'C:\Users\hamit\Desktop\Project_final\BASEAUTO.xls', sheet_name='DATA_2017')
auto18=pd.read_excel(r'C:\Users\hamit\Desktop\Project_final\BASEAUTO.xls', sheet_name='DATA_2018')
auto19=pd.read_excel(r'C:\Users\hamit\Desktop\Project_final\BASEAUTO.xls', sheet_name='DATA_2019')
```

```
lst=[auto15,auto16,auto17,auto18,auto1
for i in lst:
   print(i.shape)
```

```
(9662, 19)
(15373, 19)
(11981, 19)
(9768, 19)
(8044, 19)
```

19 potential variables

Analysing Dataframe

Dtypes

```
for i in lst:
    print(i.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9662 entries, 0 to 9661
Data columns (total 19 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   NUMCNT     9662 non-null   int64
 1   NBAP       9662 non-null   float64
 2   CSP        9662 non-null   int64
 3   CHARGETOT  9662 non-null   float64
 4   K8000      9662 non-null   object
```

Duplicales

```
for i in lst:
    print(i.duplicated().sum())
```

```
2
2
3
0
0
```

Drop

```
for i in lst:
    i.drop_duplicates(inplace=True)
```

```
for i in lst:
    print(i.duplicated().sum())
```

```
0
0
0
0
0
```

Missing Values

```
for i in lst:
    print(i.isnull().sum())
```

```
NUMCNT       0
NBAP         0
CSP          0
CHARGETOT    0
K8000        0
STATUT       0
USAGE        0
ENE          0
ACV          0
SEXE         0
AGECOND      0
PERMIS       0
CRM          0
GARAGE       0
SEGM         0
ALI          0
VITMAX       0
CAR          0
CLAPRIX      0
dtype: int64
```

Check the unique values

```
['AGECOND'].unique()
```

```
y(['21-25 ANS', '<= 20 ANS', '26-30 ANS', '51-60 ANS', '41-50
   '31-40 ANS', '61-65 ANS', '71 ANS ET PLUS', '66-70 ANS'],
  dtype=object)
```

```
auto.loc[auto['AGECOND']== '21-25 ANS', 'AGECOND_T']=21
auto.loc[auto['AGECOND']== '<= 20 ANS', 'AGECOND_T']=20
auto.loc[auto['AGECOND']== '26-30 ANS', 'AGECOND_T']=26
auto.loc[auto['AGECOND']== '51-60 ANS', 'AGECOND_T']=51
auto.loc[auto['AGECOND']== '41-50 ANS', 'AGECOND_T']=41
auto.loc[auto['AGECOND']== '31-40 ANS', 'AGECOND_T']=31
auto.loc[auto['AGECOND']== '61-65 ANS', 'AGECOND_T']=61
auto.loc[auto['AGECOND']== '71 ANS ET PLUS', 'AGECOND_T']=71
auto.loc[auto['AGECOND']== '66-70 ANS', 'AGECOND_T']=66
```

Columns

```
auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51293 entries, 0 to 51292
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Amount_paid             51293 non-null  float64
 1   Limited_Milesage_option 51293 non-null  object 
 2   Marital_status          51293 non-null  object 
 3   Vehicle_use             51293 non-null  object 
 4   Energy_type             51293 non-null  object 
 5   Vehicle_age             51293 non-null  float64
 6   Sex                     51293 non-null  object 
 7   Driver_age              51293 non-null  float64
 8   License_issuance        51293 non-null  float64
 9   Bonus_malus             51293 non-null  float64
 10  Garage                  51293 non-null  object 
 11  Vehicle_segment         51293 non-null  object 
 12  Max_speed               51293 non-null  float64
 13  Car_type                51293 non-null  object 
 14  Price_class_vehicle     51293 non-null  object 
dtypes: float64(6), object(9)
memory usage: 5.9+ MB
```

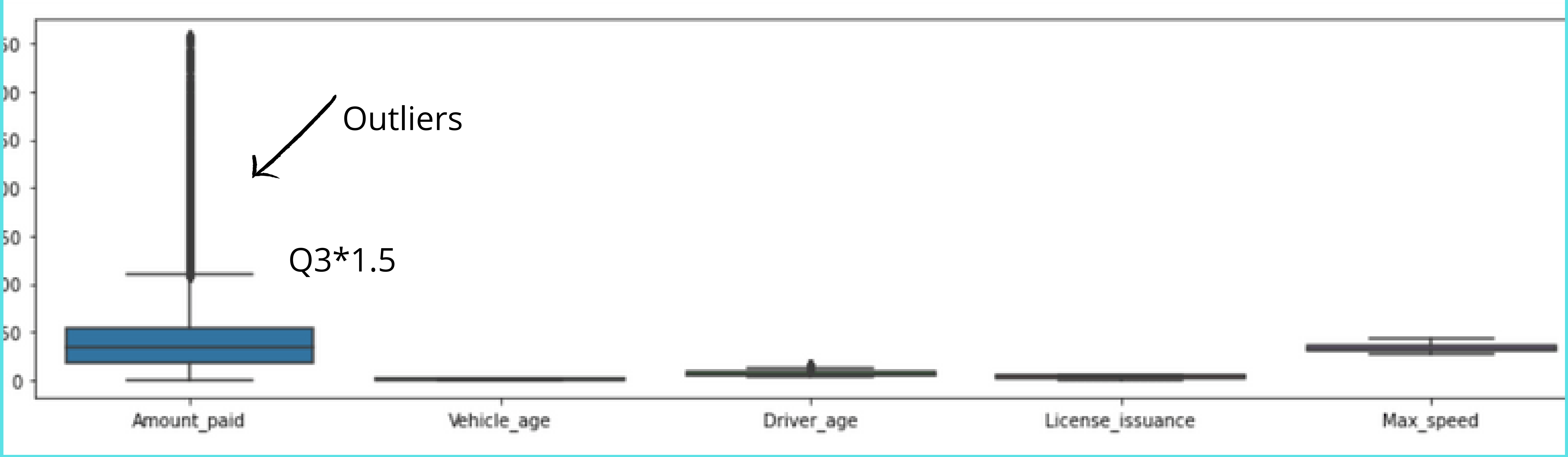
Numerical columns

```
auto.describe()
```

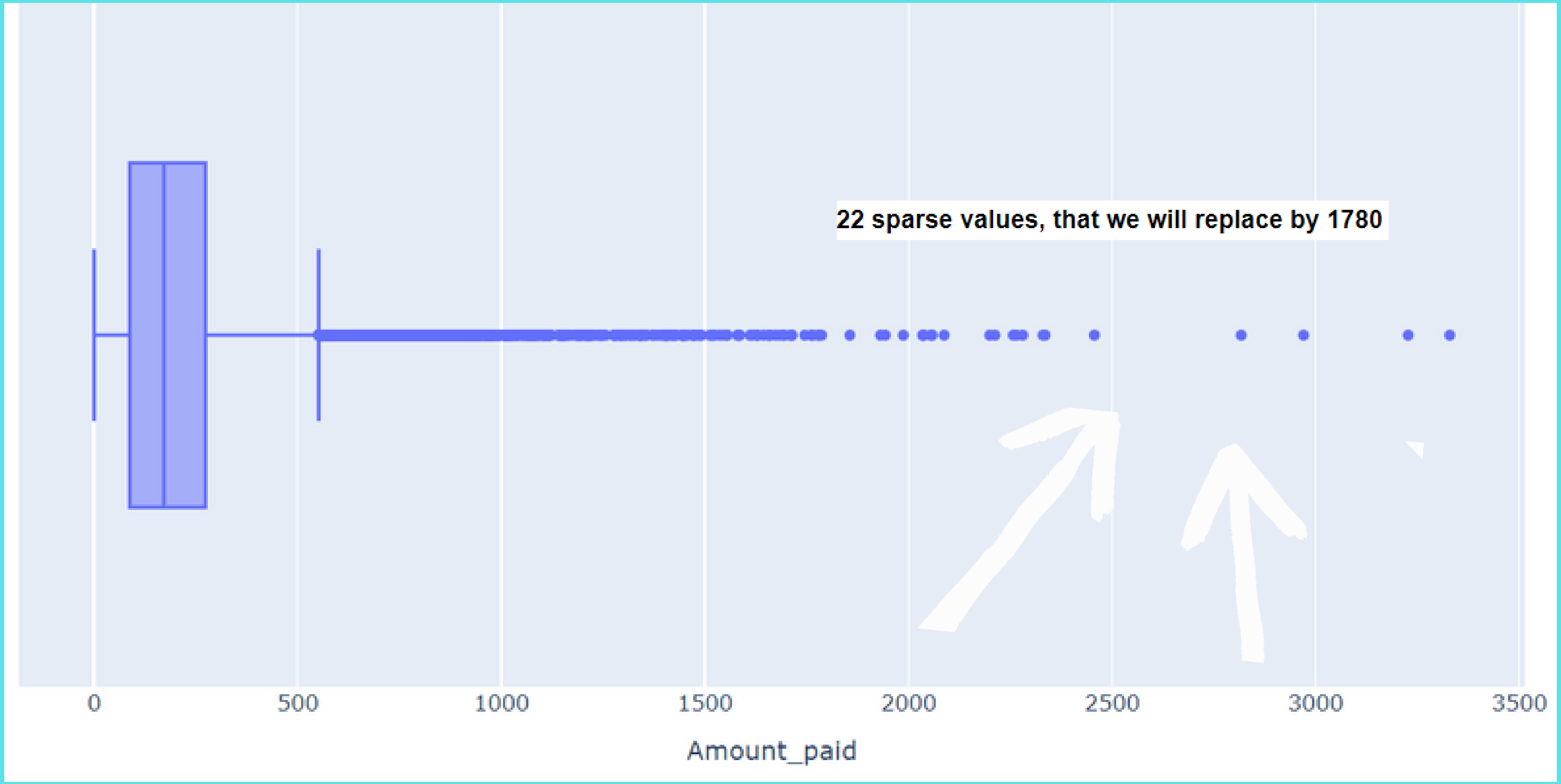
	Amount_paid	Vehicle_age	Driver_age	License_issuance	Bonus_malus	Max_speed
count	51293.000000	51293.000000	51293.000000	51293.000000	51293.000000	51293.000000
mean	207.911114	7.569045	36.137465	16.385832	3.054569	165.932837
std	178.466435	3.792030	13.462191	10.388156	3.425185	20.677004
min	0.180000	0.000000	20.000000	0.000000	-1.000000	140.000000
25%	87.650000	4.000000	26.000000	7.000000	0.000000	151.000000
50%	171.750000	8.000000	31.000000	17.000000	1.000000	161.000000
75%	273.160000	11.000000	41.000000	24.500000	6.000000	181.000000
max	3329.600000	11.000000	71.000000	30.000000	9.000000	220.000000

What about outliers?

```
t.figure(figsize=(16,4))
s.boxplot(data=auto)
t.show()
```



Column 'Amount_paid"

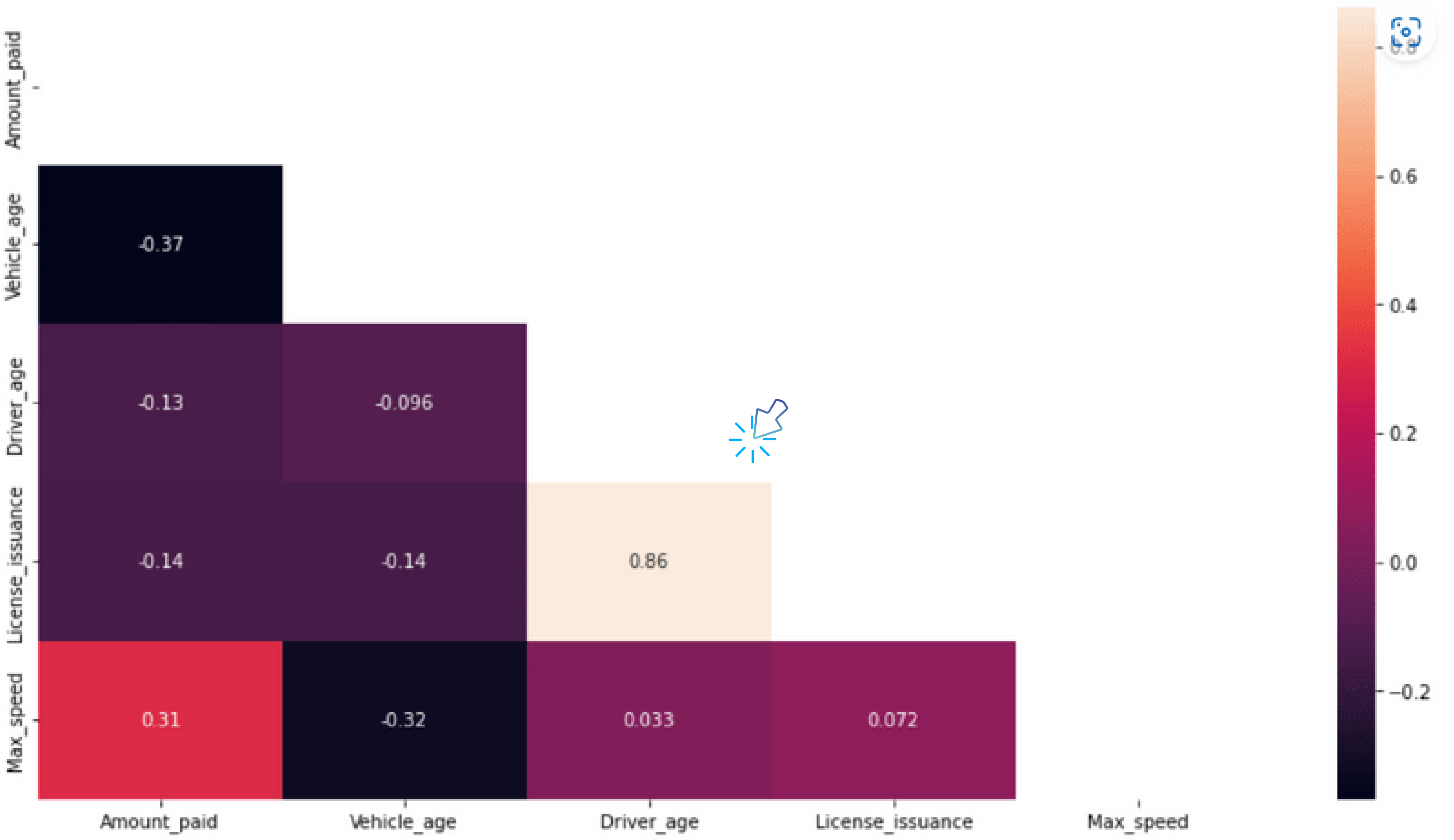



```
auto.loc[auto['Amount_paid']>1800, 'Amount_paid']=1780
```

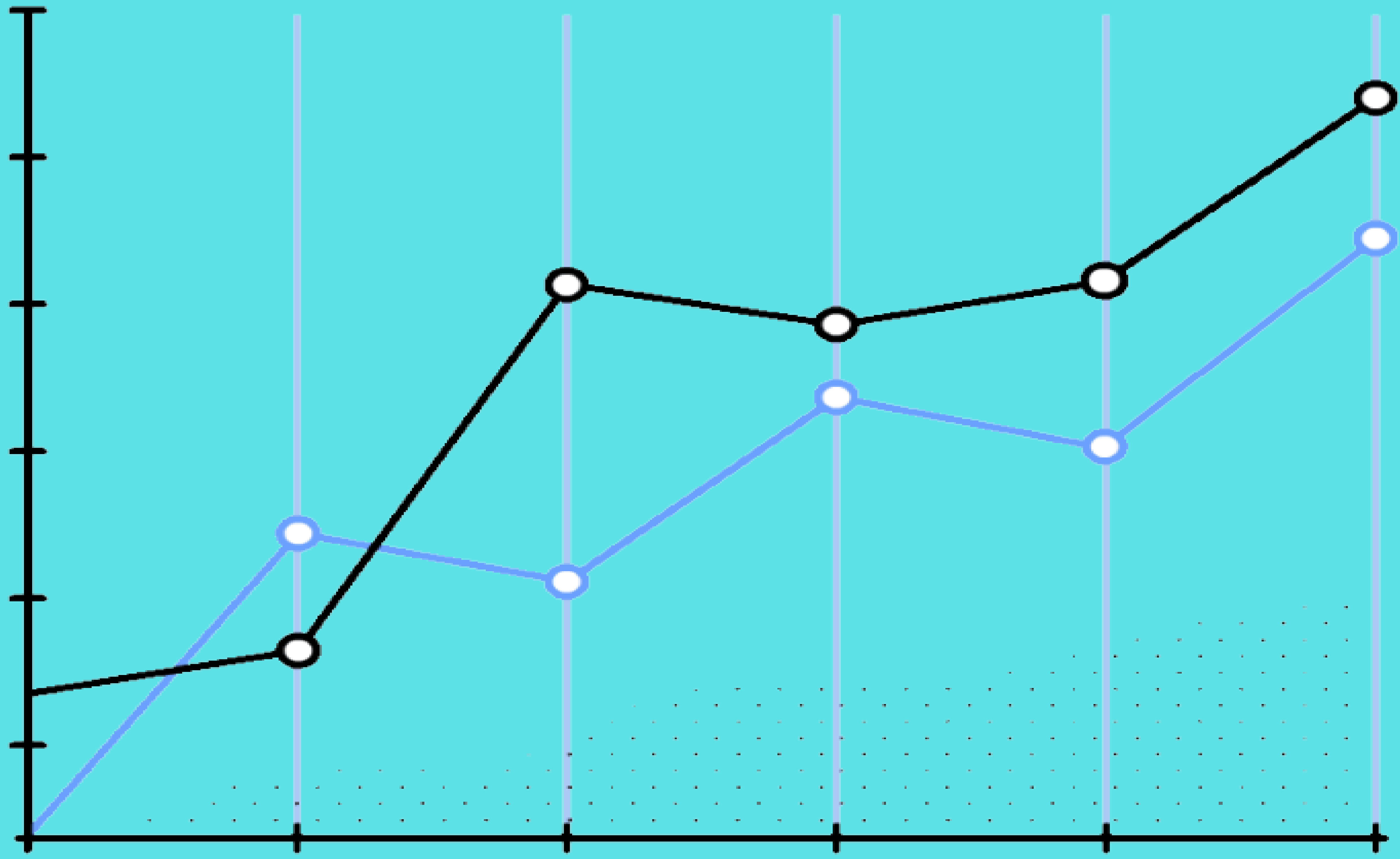
Numerical columns correlation

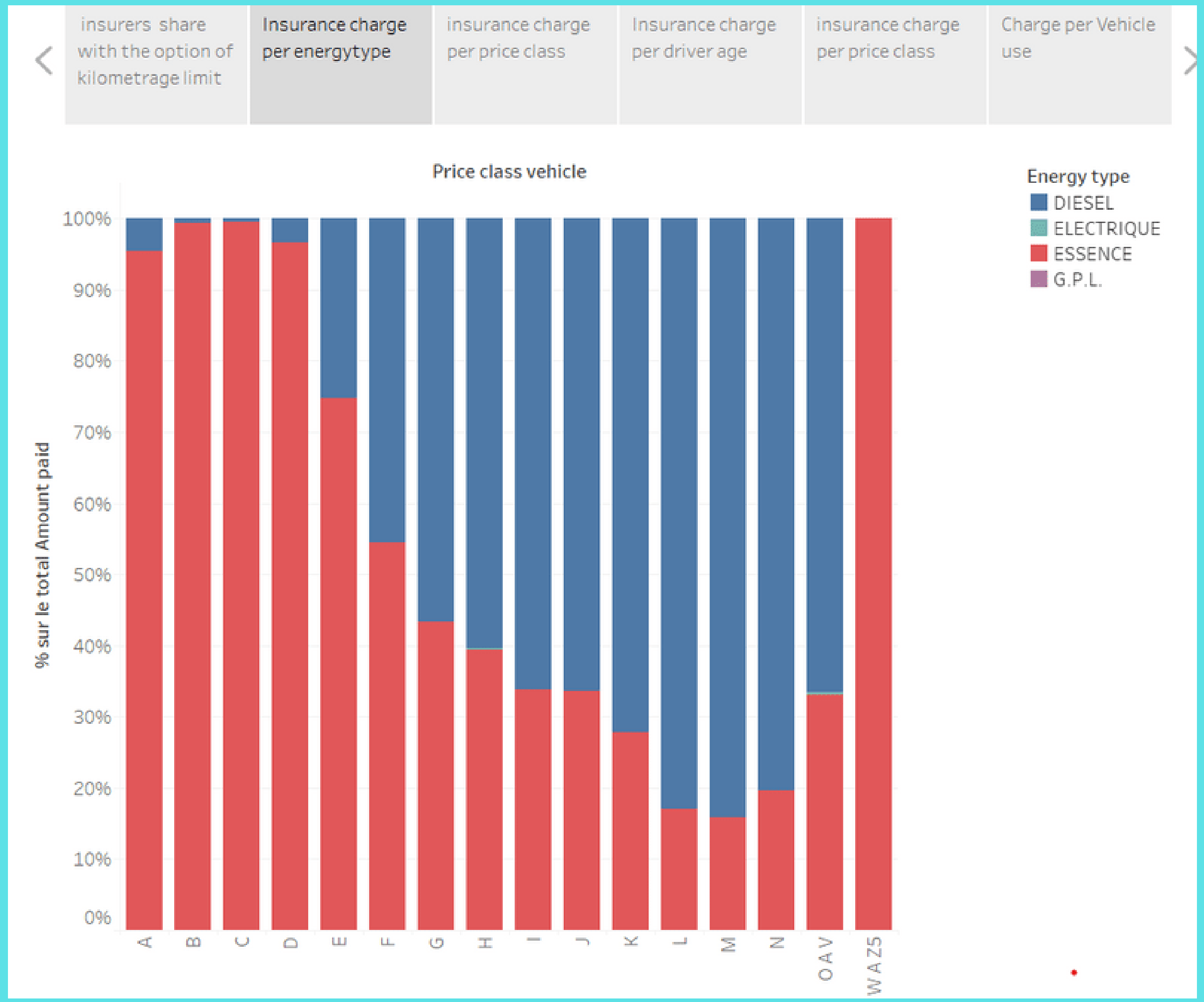
```
corr=auto.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
plt.figure(figsize=(15,8))
sns.heatmap(corr,annot=True, mask=mask)
```

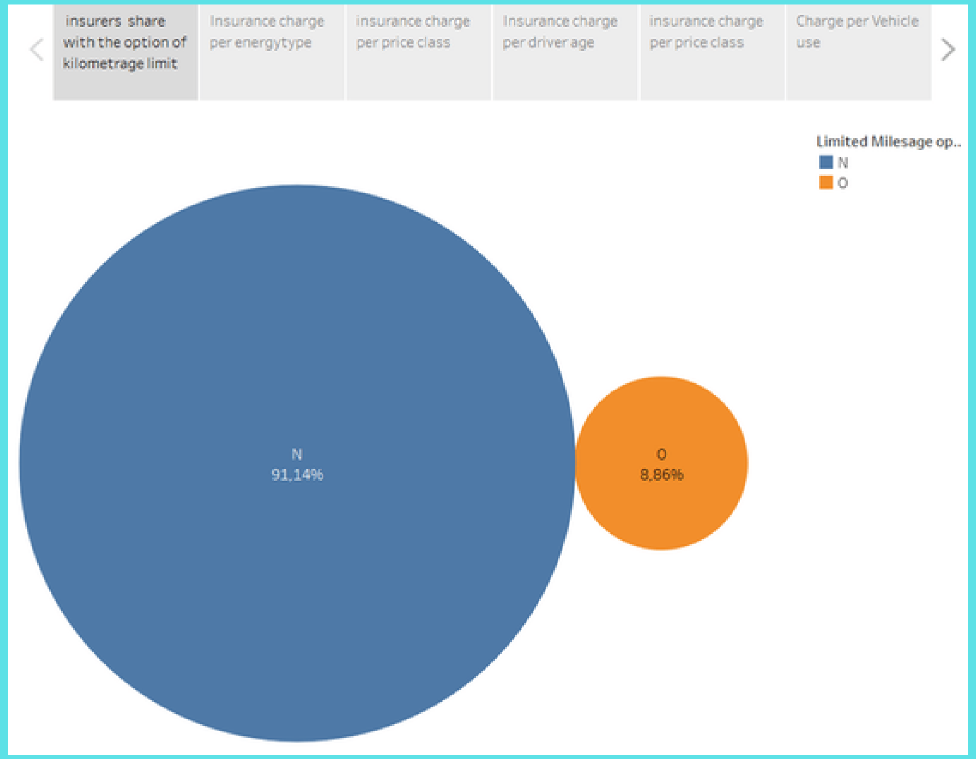
```
<AxesSubplot:>
```

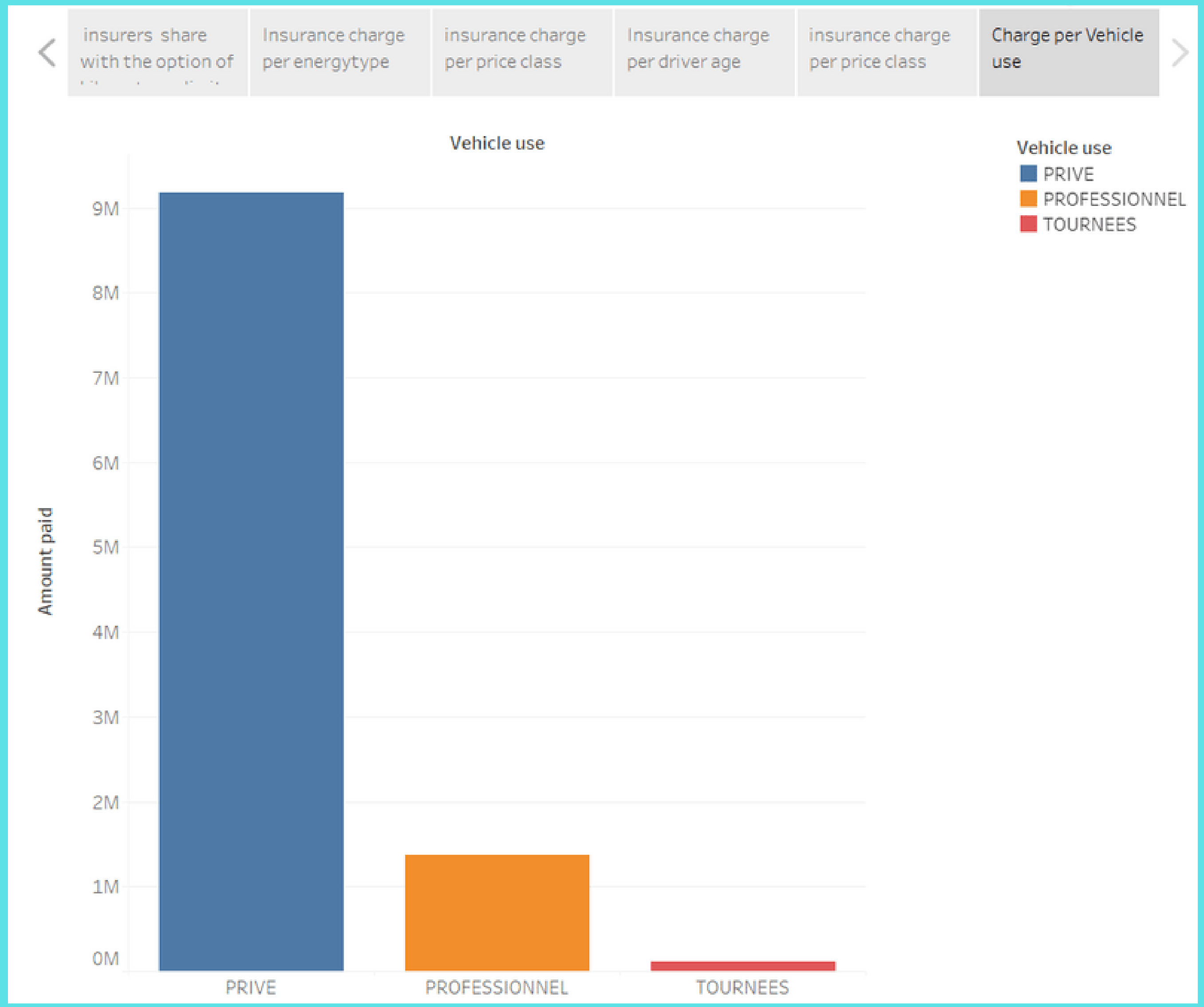


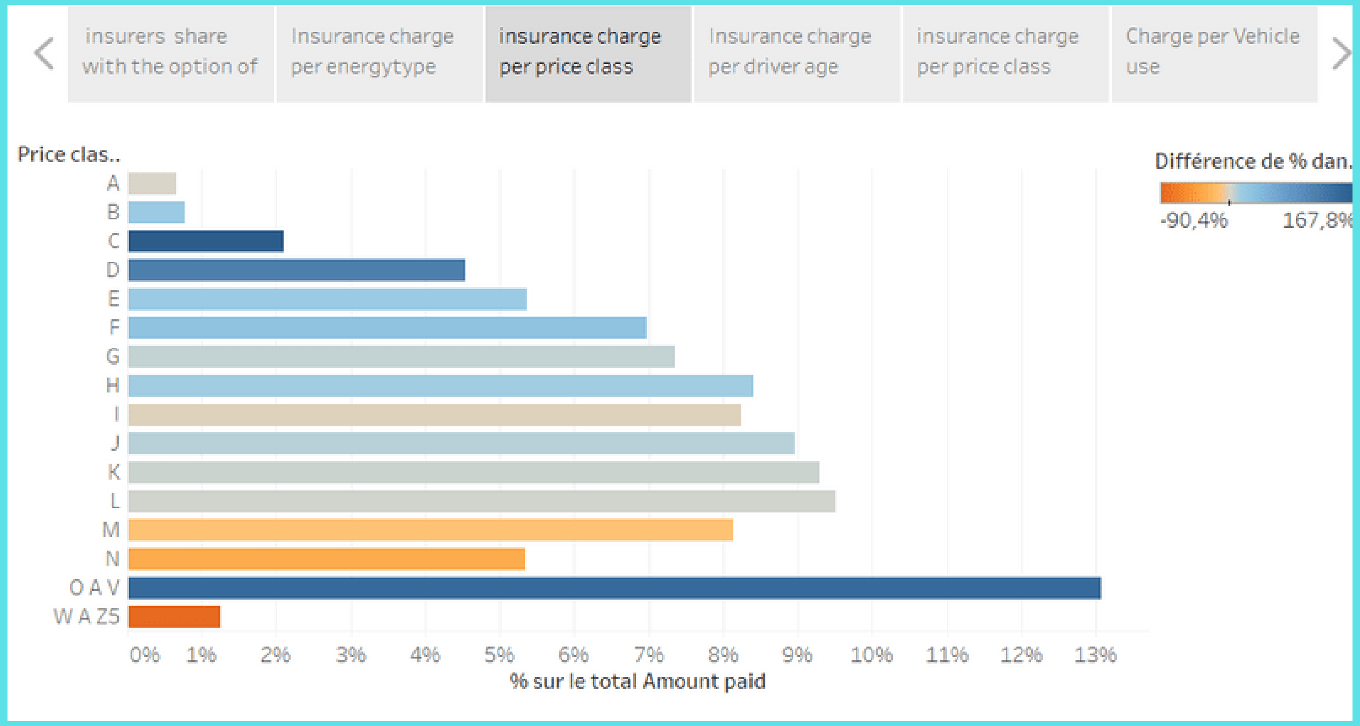
Now... ready for visualization!

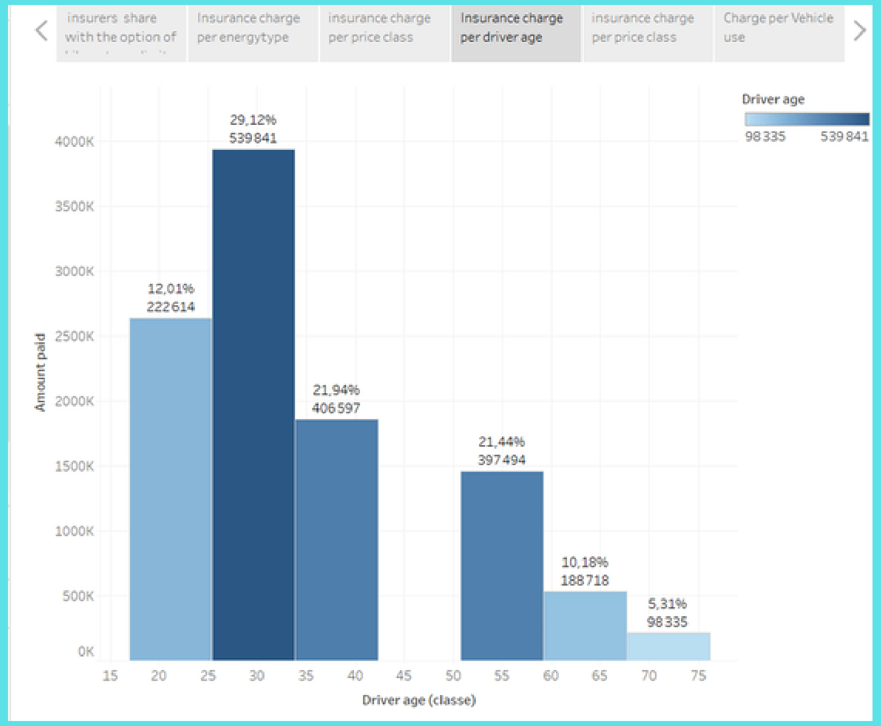


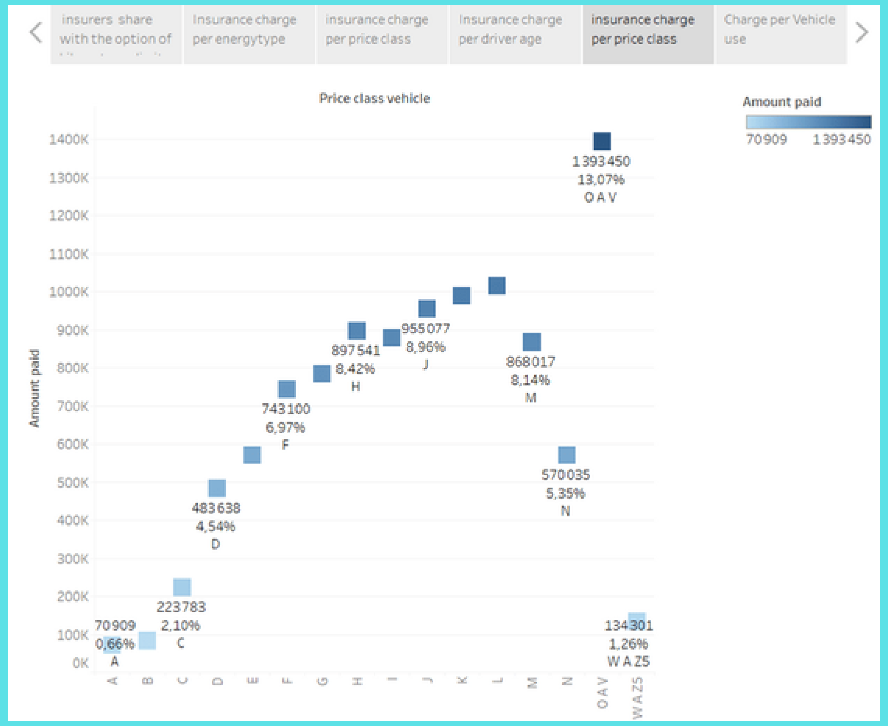

insurers share with the option of kilometrage limit
Insurance charge per energytype
insurance charge per price class
Insurance charge per driver age
insurance charge per price class
Charge per Vehicle use
Price class vehicle
Energy type
DIESEL
ELECTRIQUE
ESSENCE
G.P.L.
% sur le total Amount paid
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O A V
W A Z5


insurers share with the option of kilometrage limit
Insurance charge per energytype
insurance charge per price class
Insurance charge per driver age
insurance charge per price class
Charge per Vehicle use
Limited Mileage op..
N
O
N
91,14%
O
8,86%


insurers share with the option of
Insurance charge per energytype
insurance charge per price class
Insurance charge per driver age
insurance charge per price class
Charge per Vehicle use
Vehicle use
Vehicle use
PRIVE
PROFESSIONNEL
TOURNEES
Amount paid
9M
8M
7M
6M
5M
4M
3M
2M
1M
0M
PRIVE
PROFESSIONNEL
TOURNEES


insurers share with the option of
Insurance charge per energytype
insurance charge per price class
Insurance charge per driver age
insurance charge per price class
Charge per Vehicle use
Price clas..
Différence de % dan..
-90,4%
167,8%
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O A V
W A Z5
0%
1%
2%
3%
4%
5%
6%
7%
8%
9%
10%
11%
12%
13%
% sur le total Amount paid


Driver age
98 335
539 841
Amount paid
12,01%
222 614
29,12%
539 841
21,94%
406 597
21,44%
397 494
10,18%
188 718
5,31%
98 335
Driver age (classe)


Price class vehicle
Amount paid
70 909
1 393 450
1 393 450
13,07%
O A V
955 077
8,96%
J
897 541
8,42%
H
868 017
8,14%
M
743 100
6,97%
F
570 035
5,35%
N
483 638
4,54%
D
223 783
2,10%
C
70 909
0,66%
A
134 301
1,26%
W A Z5

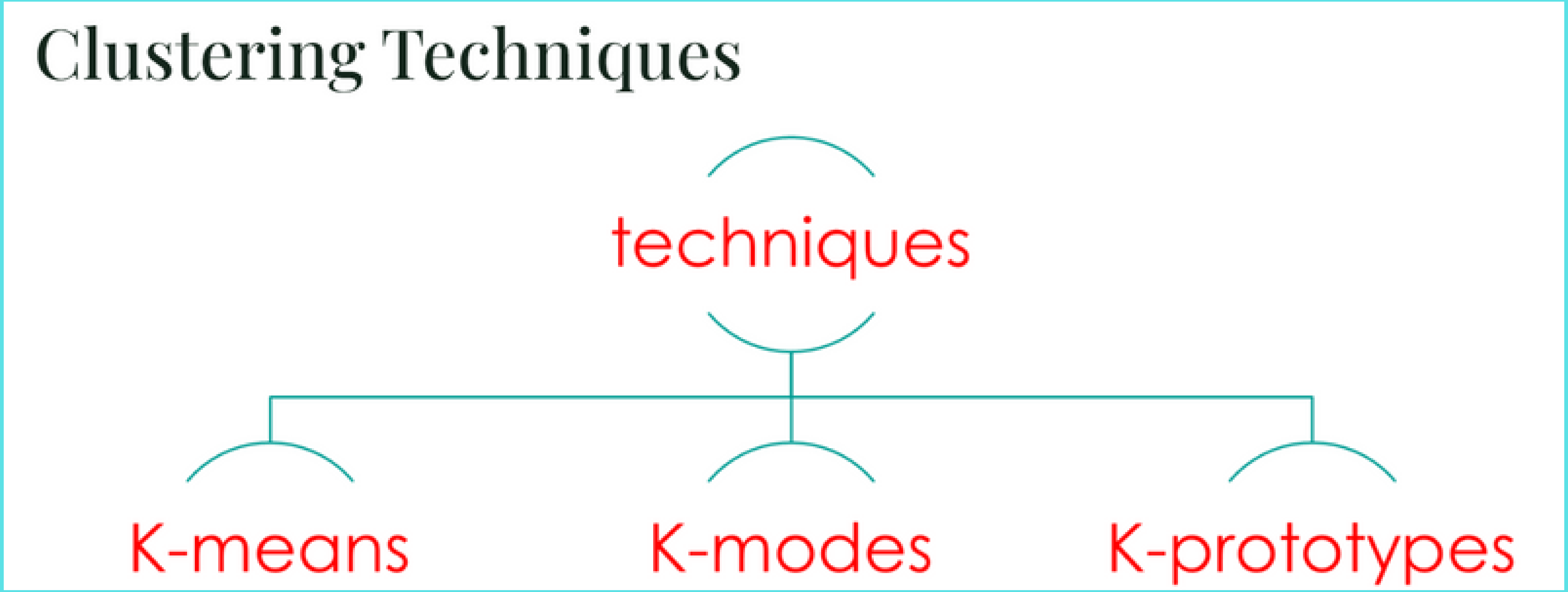
Selected columns for unsupervised

Categorical columns	Numerical columns
'Limited_Milesage_option'	'Amount_paid'
'Vehicle_use'	'Vehicle_age'
'Energy_type'	'Driver_age'
'Garage'	'License_issuance'
'Price_class_vehicle'	'Max_speed'

Insurers clustering

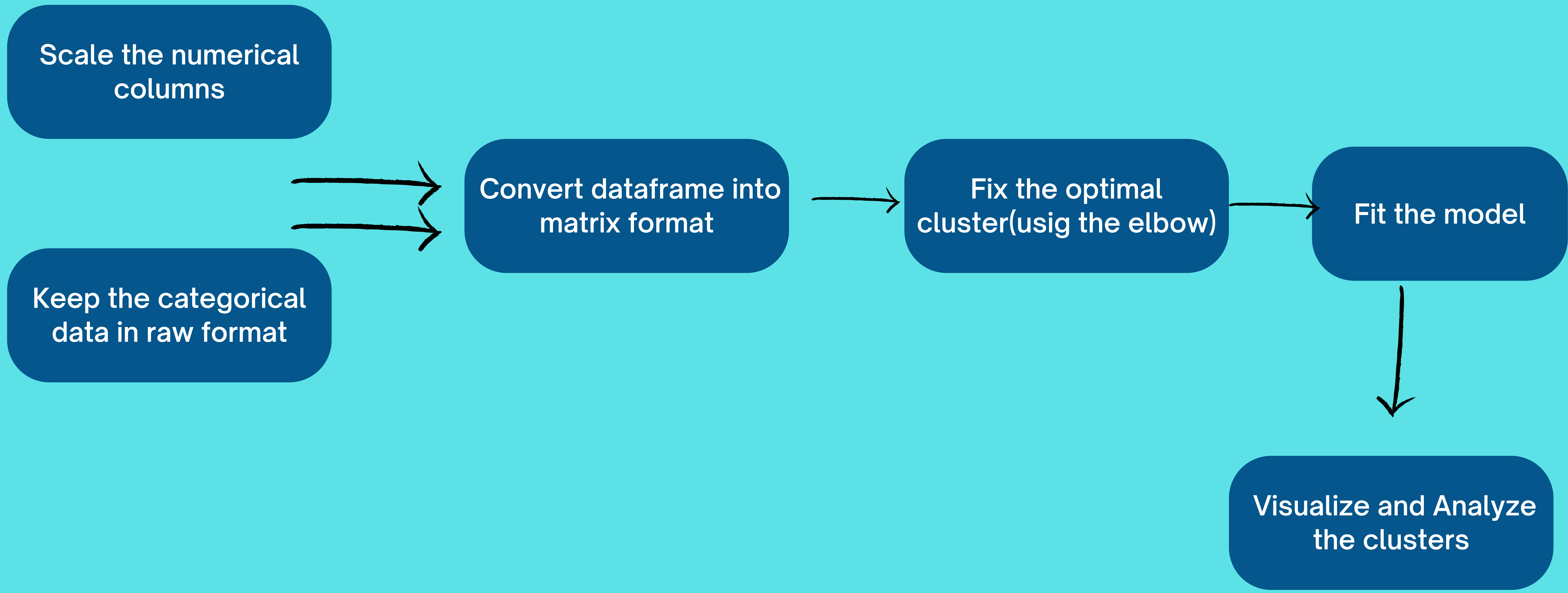
[unsupervised Learning]

Which algorithm should be use?



[Source:
Customer Segmentation Project using K-prototypes with Code Source - AI decoder (decoderai.com)]

K-prototypes technique



Scale the numerical columns (MinMaxScaler)

```python
from sklearn.preprocessing import MinMaxScaler
from sklearn.compose import make_column_transformer
```

```python
transformer=make_column_transformer((MinMaxScaler(),['Amount_paid','Vehicle_age', 'Driver_age',  'License_issuance', 'Max_speed'
a=transformer.fit_transform(auto)
a
```

```
array([[0.03459993, 1.        , 0.01960784, 0.1       , 0.1375    ],
       [0.00219433, 1.        , 0.01960784, 0.1       , 0.1375    ],
       [0.01522038, 1.        , 0.        , 0.        , 0.1375    ],
       ...,
       [0.16478768, 1.        , 0.11764706, 0.23333333, 0.3875    ],
       [0.0394644 , 1.        , 0.11764706, 0.23333333, 0.3875    ],
       [0.02089094, 1.        , 0.11764706, 0.23333333, 0.3875    ]])
```

	Amount_paid	Vehicle_age	Driver_age	License_issuance	Max_speed
0	0.034600	1.0	0.019608	0.100000	0.1375
1	0.002194	1.0	0.019608	0.100000	0.1375
2	0.015220	1.0	0.000000	0.000000	0.1375
3	0.015623	1.0	0.117647	0.233333	0.1375

Keep the categorical data in raw format

```
b=auto[['Limited_Milesage_option', 'Vehicle_use', 'Energy_type', 'Garage',  'Price_class_vehicle' ]]
```

```
auto_scal=pd.concat([a,b], axis=1)
auto_scal.head()
```

	Amount_paid	Vehicle_age	Driver_age	License_issuance	Max_speed	Limited_Milesage_option	Vehicle_use	Energy_type	Garage	Price_class_vehicle
0	0.034600	1.0	0.019608	0.100000	0.1375	N	PRIVE	ESSENCE	AUTRES	E
1	0.002194	1.0	0.019608	0.100000	0.1375	N	PRIVE	ESSENCE	AUTRES	E
2	0.015220	1.0	0.000000	0.000000	0.1375	N	PRIVE	ESSENCE	AUTRES	C
3	0.015623	1.0	0.117647	0.233333	0.1375	N	PRIVE	DIESEL	AUTRES	H
4	0.088742	1.0	0.117647	0.233333	0.1375	N	PRIVE	DIESEL	AUTRES	H

Get the position of the categorical data

```
catColumnsPos = [auto_scal.columns.get_loc(col) for col in list(auto_scal.select_dtypes('object').columns)]

print('Categorical columns          : {}'.format(list(auto_scal.select_dtypes('object').columns)))
print('Categorical columns position  : {}'.format(catColumnsPos))
```

```
Categorical columns           : ['Limited_Milesage_option', 'Vehicle_use', 'Energy_type', 'Garage', 'Price_class_vehicle']
Categorical columns position  : [5, 6, 7, 8, 9]
```

Convert dataframe into matrix format

```
import numpy as np
auto_array=auto_scal.to_numpy()
```

```
auto_array
```

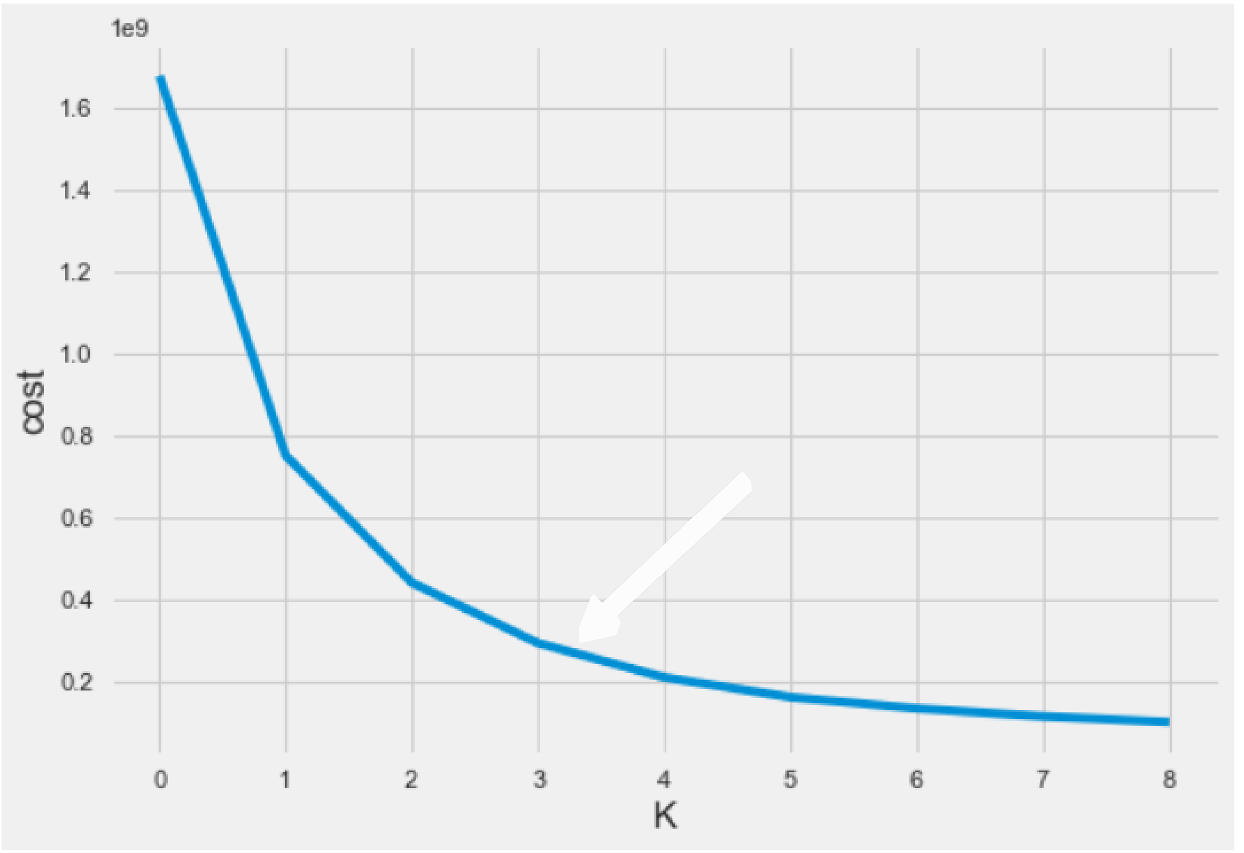
```
array([[0.03459992610920165, 1.0, 0.019607843137254888, ..., 'ESSENCE',
        'AUTRES', 'E'],
       [0.0021943327996775674, 1.0, 0.019607843137254888, ..., 'ESSENCE',
        'AUTRES', 'E'],
       [0.015220384903885987, 1.0, 0.0, ..., 'ESSENCE', 'AUTRES', 'C'],
       ...,
       [0.16478767591048019, 1.0, 0.11764705882352938, ..., 'DIESEL',
        'INDIVIDUEL CLOS', 'J'],
       [0.03946440366767054, 1.0, 0.11764705882352938, ..., 'DIESEL',
        'INDIVIDUEL CLOS', 'J'],
       [0.02089094389897113, 1.0, 0.11764705882352938, ..., 'DIESEL',
        'INDIVIDUEL CLOS', 'J']], dtype=object)
```

Fix the optimal cluster(usig the elbow method)

```python
from kmodes.kprototypes import KPrototypes

cost = []
for cluster in range(1, 10):
    try:
        kprototype = KPrototypes(n_jobs = -1, n_clusters = cluster, init = 'Huang', random_state = 0)
        kprototype.fit_predict(auto_array, categorical = catColumnsPos)
        cost.append(kprototype.cost_)
        print('Cluster initiation: {}'.format(cluster))
    except:
        break

plt.plot(cost)
plt.xlabel('K')
plt.ylabel('cost')
```




```python
from kneed import KneeLocator
cost_knee_c3 = KneeLocator(
        x=range(1,10),
        y=cost,
        S=0.1, curve="convex", direction="decreasing", online=True)

K_cost_c3 = cost_knee_c3.elbow
print("elbow at k =", f'{K_cost_c3:.0f} clusters')
```

```
elbow at k = 3 clusters
```

New dataframe with columns 'clusters'

ergy_type	Vehicle_age	Sex	Driver_age	License_issuance	Bonus_malus	Garage	Vehicle_segment	Max_speed	Car_type	Price_class_vehicle	clusters
ESSENCE	11.0	HOMME	21.0	3.0	8.0	AUTRES	B	151.0	BERLINE	E	2
ESSENCE	11.0	HOMME	21.0	3.0	8.0	AUTRES	B	151.0	BERLINE	E	2
ESSENCE	11.0	FEMME	20.0	0.0	9.0	AUTRES	B	151.0	BERLINE	C	2
DIESEL	11.0	HOMME	26.0	7.0	8.0	AUTRES	M1	151.0	BERLINE	H	2
DIESEL	11.0	HOMME	26.0	7.0	8.0	AUTRES	M1	151.0	BERLINE	H	2
...	...	...	...	...	...	...	...	...	...	...	...
DIESEL	11.0	HOMME	51.0	30.0	0.0	AUTRES	M2	161.0	BERLINE	H	0
DIESEL	11.0	HOMME	51.0	30.0	0.0	AUTRES	M2	161.0	BERLINE	H	0
DIESEL	11.0	HOMME	26.0	7.0	3.0	INDIVIDUEL CLOS	M1	171.0	BERLINE	J	2
DIESEL	11.0	HOMME	26.0	7.0	2.0	INDIVIDUEL CLOS	M1	171.0	BERLINE	J	2
DIESEL	11.0	HOMME	26.0	7.0	2.0	INDIVIDUEL CLOS	M1	171.0	BERLINE	J	2

PCA

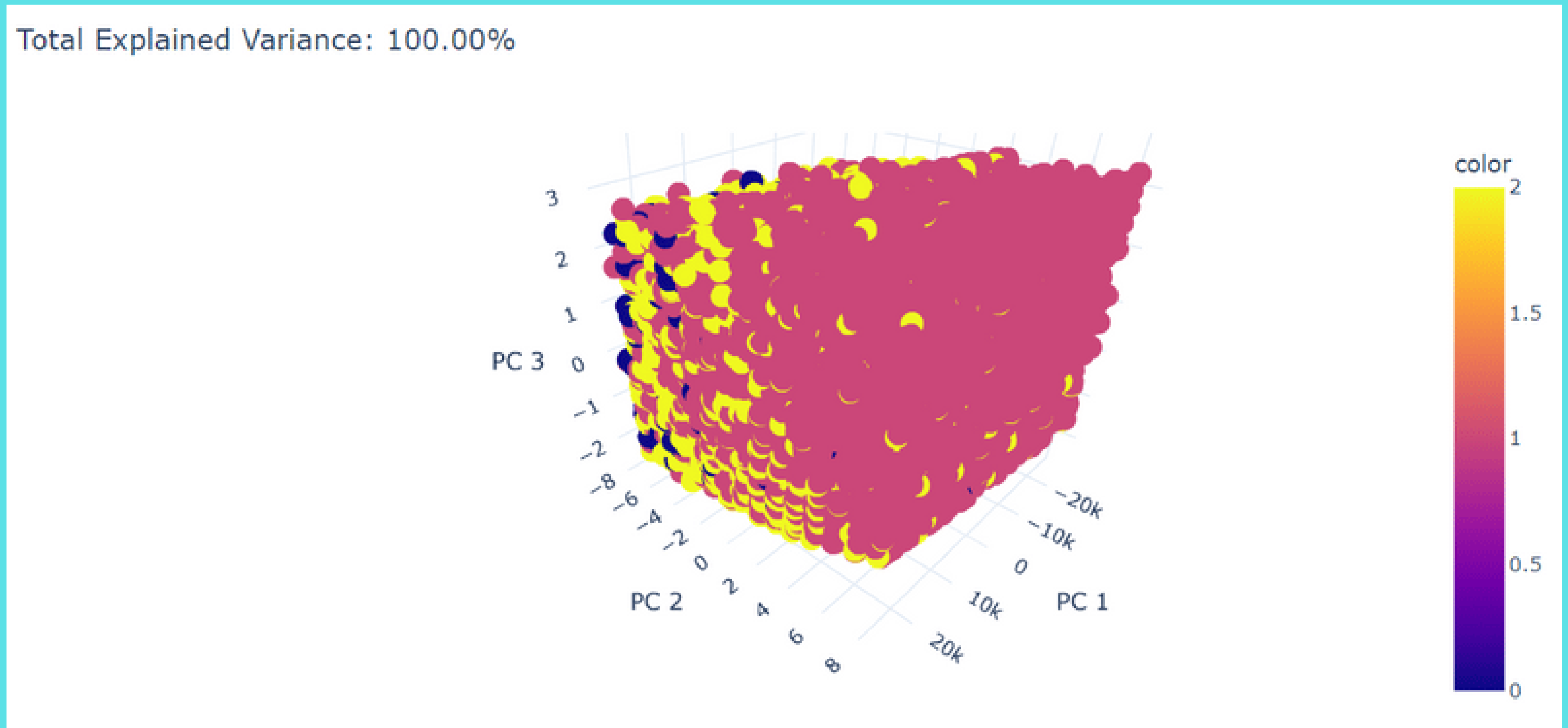
Standardization + Encoding

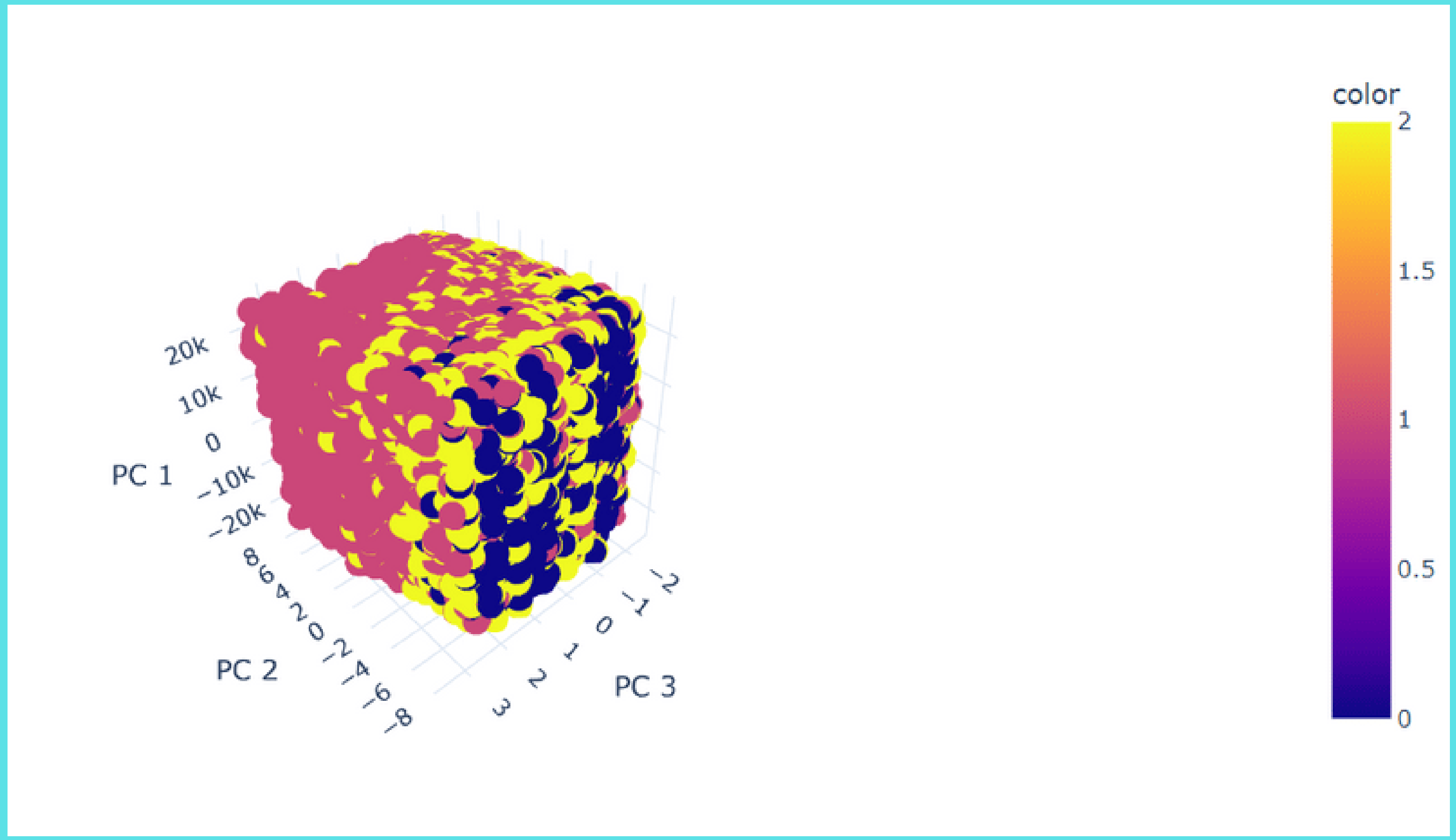
import labraryit the model

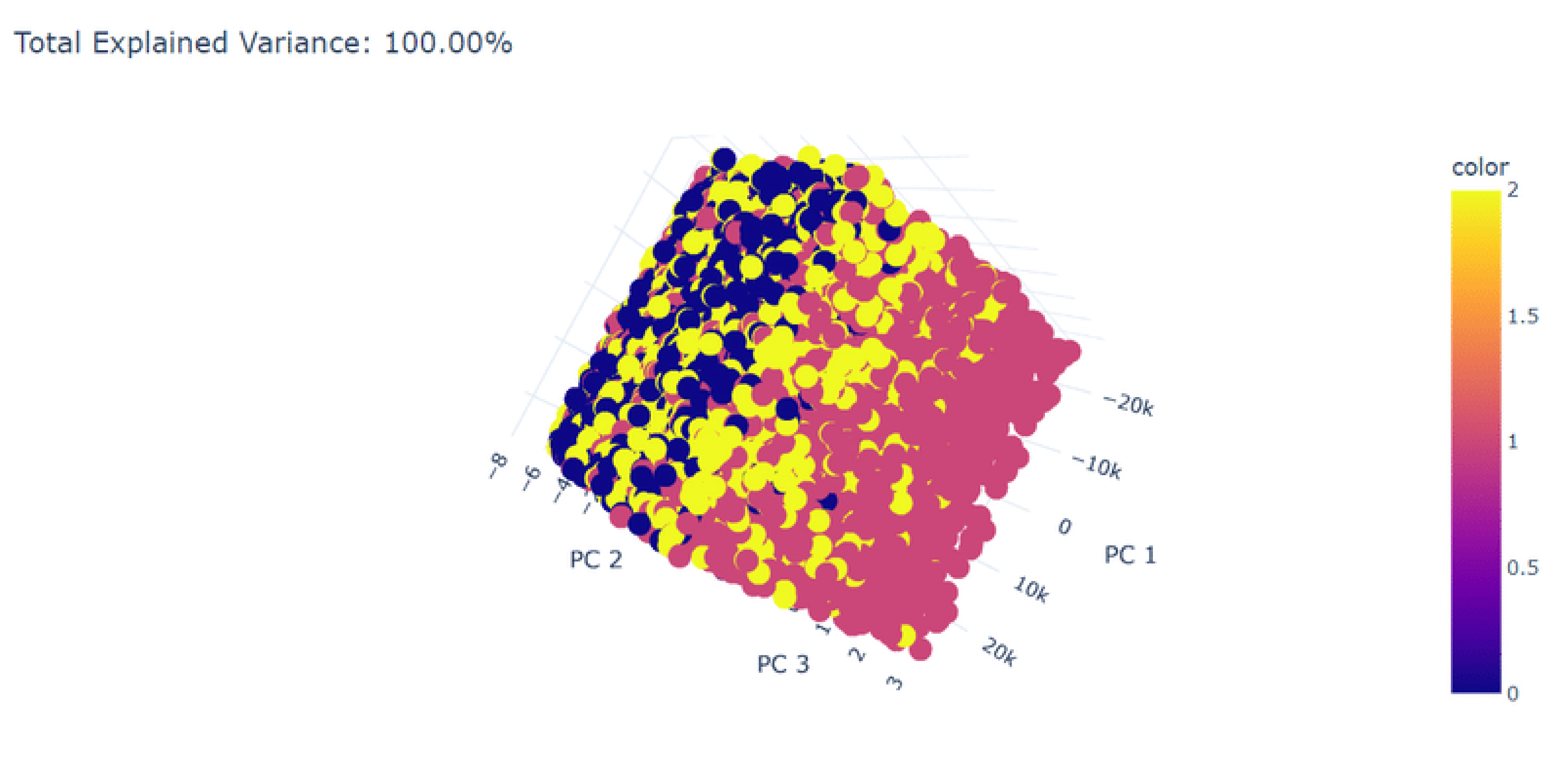
```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
pca.fit_transform(auto_pca)
```

```
rray([[ 2.56460001e+04,  3.85091894e+00, -1.17654940e+00],
       [ 2.56450001e+04,  3.85093560e+00, -1.17655897e+00],
       [ 2.56440001e+04,  5.80542371e+00, -1.30701998e+00],
       ...,
       [-2.56440000e+04, -6.73823509e-01, -1.45212727e+00],
       [-2.56450000e+04, -6.62071937e-01, -1.42101803e+00],
       [-2.56460000e+04, -6.71719297e-01, -1.48043853e+00]])
```

PCA (3D visualization)







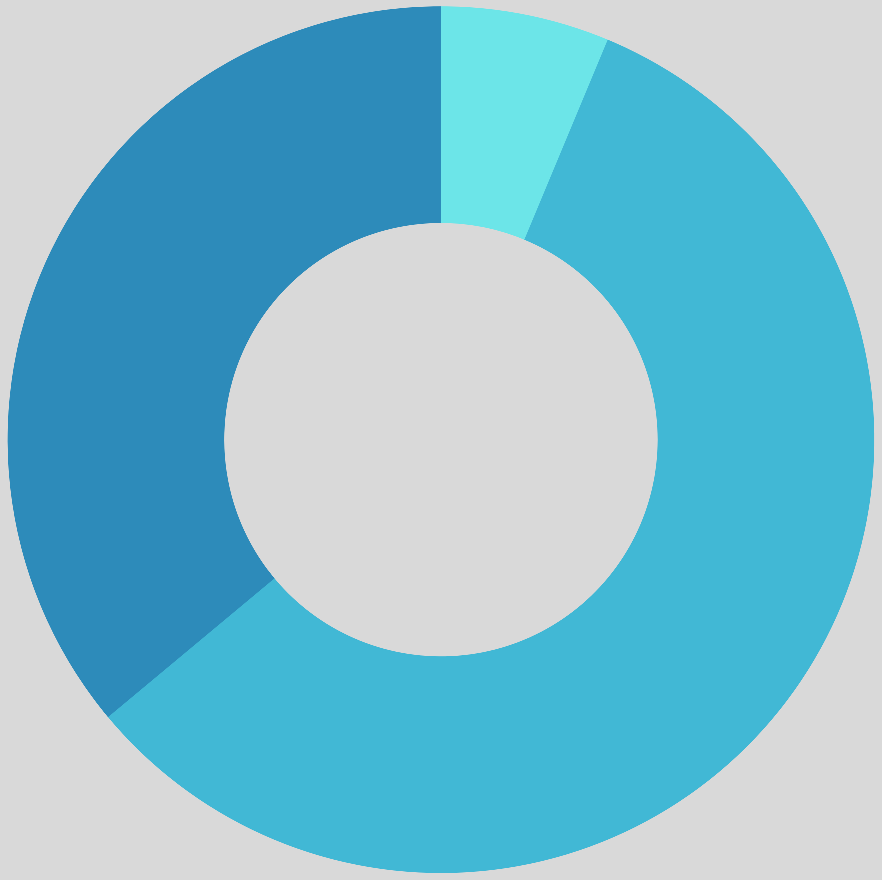
Cluster 0

Cluster 1

Cluster 2

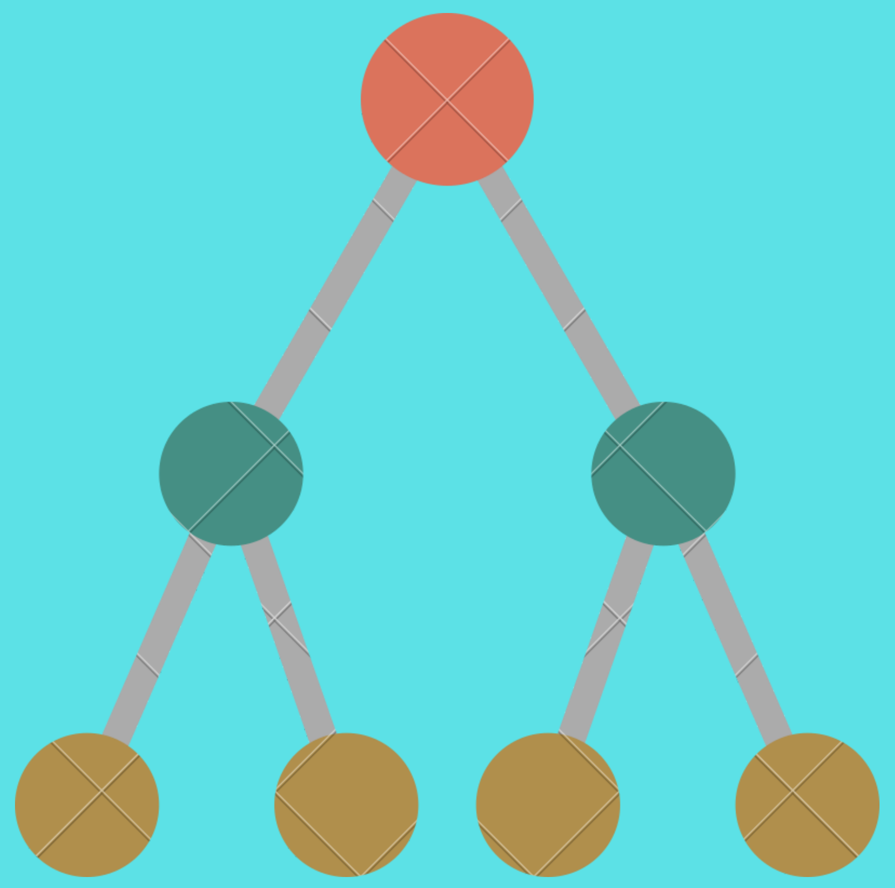
Clusters(Labels) visulization

clusters	Amount_paid	Limited_Milesage_option	Vehicle_use	Energy_type	Vehicle_age	Driver_age	License_issuance	Bonus_malus	Garage	Max_speed	Pri
0	48.52	N	PRIVE	ESSENCE	11.0	41.0	30.0	0.0	AUTRES	140.0	
1	1780.00	N	PRIVE	DIESEL	0.0	31.0	30.0	0.0	AUTRES	181.0	
2	1.14	N	PRIVE	DIESEL	11.0	21.0	7.0	9.0	AUTRES	161.0	

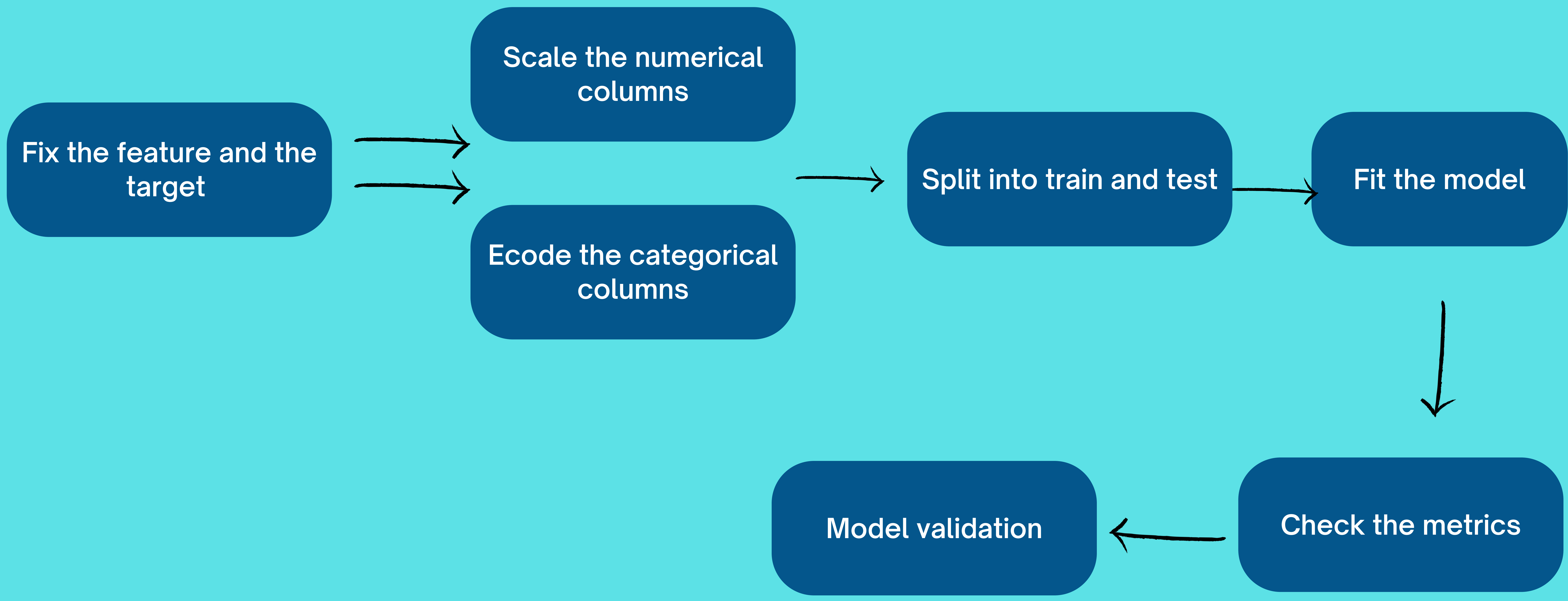


Insurers Classification

[Supervised Learning]



Classification technique



Encoding

	Amount_paid	Limited_Milesage_option	Vehicle_use	Energy_type	Vehicle_age	Driver_age	License_issuance	Garage	Max_speed	Price_class_vehic
0	61.99	N	PRIVE	ESSENCE	11.0	21.0	3.0	AUTRES	151.0	
1	4.10	N	PRIVE	ESSENCE	11.0	21.0	3.0	AUTRES	151.0	
2	27.37	N	PRIVE	ESSENCE	11.0	20.0	0.0	AUTRES	151.0	
3	28.09	N	PRIVE	DIESEL	11.0	26.0	7.0	AUTRES	151.0	
4	158.71	N	PRIVE	DIESEL	11.0	26.0	7.0	AUTRES	151.0	
...	...	...	...	...	...	...	...	...	...	
51288	49.10	O	PRIVE	DIESEL	11.0	51.0	30.0	AUTRES	161.0	
51289	39.94	O	PRIVE	DIESEL	11.0	51.0	30.0	AUTRES	161.0	
51290	294.56	O	PRIVE	DIESEL	11.0	26.0	7.0	INDIVIDUEL	171.0	

```python
from sklearn.preprocessing import LabelEncoder
```

```python
b=x[categorical_features].apply(LabelEncoder().fit_transform)
b
```

	Limited_Milesage_option	Vehicle_use	Energy_type	Garage	Price_class_vehicle
0	0	0	2	0	4
1	0	0	2	0	4
2	0	0	2	0	2
3	0	0	0	0	7
4	0	0	0	0	7
...	...	...	...	...	...

Standardization

	Amount_paid	Limited_Milesage_option	Vehicle_use	Energy_type	Vehicle_age	Driver_age	License_issuance	Garage	Max_speed	Price_class_vehic
0	61.99	N	PRIVE	ESSENCE	11.0	21.0	3.0	AUTRES	151.0	
1	4.10	N	PRIVE	ESSENCE	11.0	21.0	3.0	AUTRES	151.0	
2	27.37	N	PRIVE	ESSENCE	11.0	20.0	0.0	AUTRES	151.0	
3	28.09	N	PRIVE	DIESEL	11.0	26.0	7.0	AUTRES	151.0	

```python
from sklearn.preprocessing import StandardScaler
from sklearn.compose import make_column_transformer
```

```python
transformer1=make_column_transformer((StandardScaler(), ['Vehicle_age', 'Driver_age',  'License_issu
a=transformer1.fit_transform(x)
```

	Vehicle_age	Driver_age	License_issuance	Max_speed
0	0.90479	-1.124454	-1.288579	-0.722202
1	0.90479	-1.124454	-1.288579	-0.722202
2	0.90479	-1.198737	-1.577373	-0.722202
3	0.90479	-0.753040	-0.903522	-0.722202
4	0.90479	-0.753040	-0.903522	-0.722202

Define the features and the target

Features

```
x=auto[[ 'Limited_Milesage_option', 'Vehicle_use', 'Energy_type',
       'Vehicle_age', 'Driver_age', 'License_issuance', 'Garage', 'Max_speed',
       'Price_class_vehicle']]
```

	Vehicle_age	Driver_age	License_issuance	Max_speed	Limited_Milesage_option	Vehicle_use	Energy_type	Garage	Price_class_vehi
0	0.90479	-1.124454	-1.288579	-0.722202	0	0	2	0	
1	0.90479	-1.124454	-1.288579	-0.722202	0	0	2	0	
2	0.90479	-1.198737	-1.577373	-0.722202	0	0	2	0	
3	0.90479	-0.753040	-0.903522	-0.722202	0	0	0	0	
4	0.90479	-0.753040	-0.903522	-0.722202	0	0	0	0	
...	...	...	...	...	...	...	...	...	
38	0.90479	1.104031	1.310560	-0.238569	1	0	0	0	
39	0.90479	1.104031	1.310560	-0.238569	1	0	0	0	
90	0.90479	-0.753040	-0.903522	0.245065	1	0	0	2	
91	0.90479	-0.753040	-0.903522	0.245065	1	0	0	2	

Target

```
y=auto[['clusters']]
```

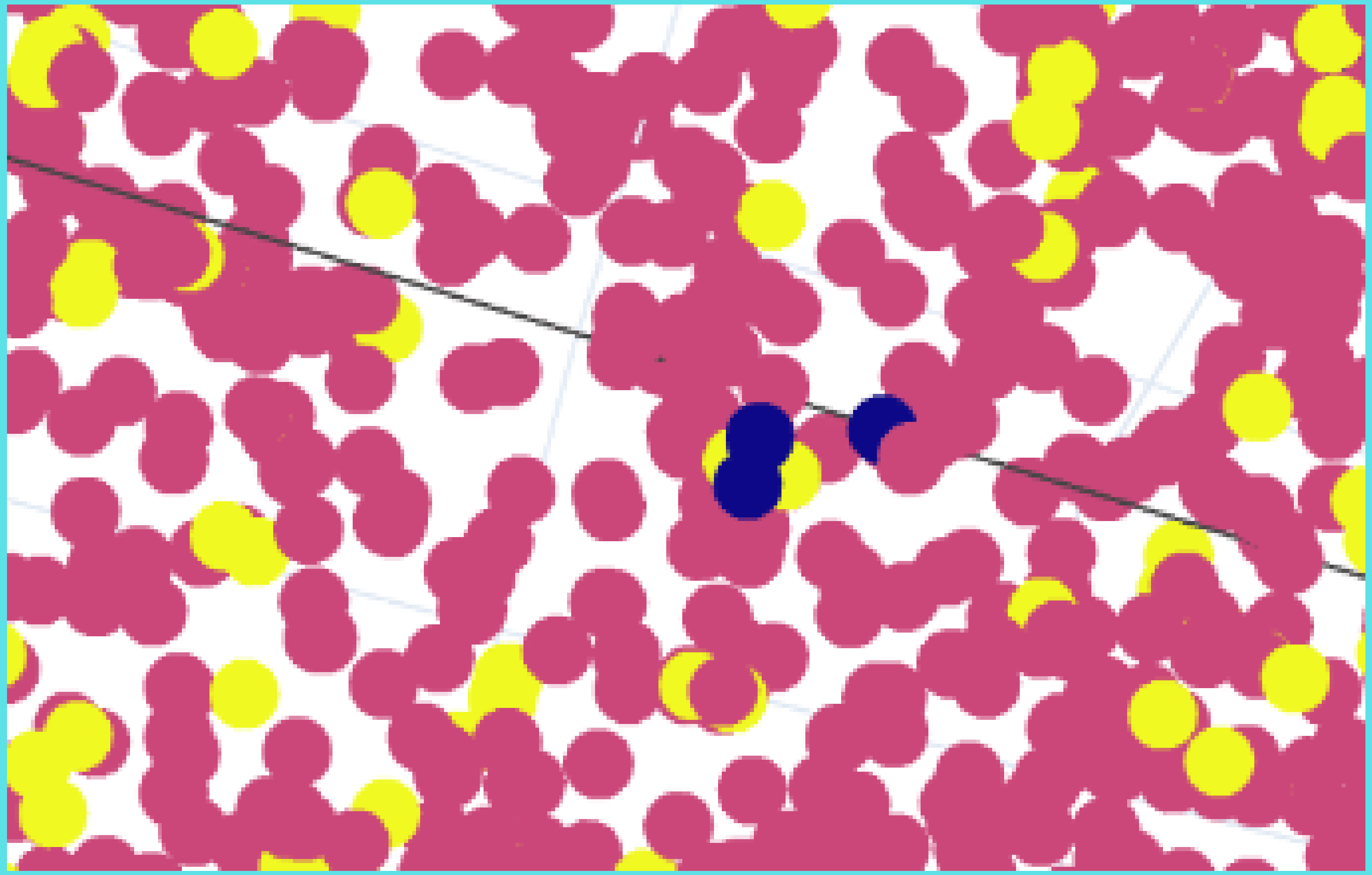
```
y
```

	clusters
0	1
1	1
2	1
3	1
4	1
...	...
51288	1

Slpit the data into train and test

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x,y, test_size=0.2)
```

Imbalanced data



BalancedRandomForestClassifier

Fit the model

```
from imblearn.ensemble import BalancedRandomForestClassifier
clf = BalancedRandomForestClassifier(max_depth=3, random_state=0)
clf.fit(x_train, y_train)
BalancedRandomForestClassifier(...)
```

Check the metrics

```
from imblearn.metrics import classification_report_imbalanced
```

```
y_true=y_test
y_pred=clf.predict(x_test)
```

```
print(classification_report_imbalanced(y_true, y_pred))
```

Results

	pre	rec	spe	f1	geo	iba	sup
0	0.89	0.92	0.95	0.90	0.94	0.88	2918
1	0.91	0.93	0.96	0.92	0.95	0.90	2968
2	0.96	0.92	0.97	0.94	0.94	0.89	4373
avg / total	0.92	0.92	0.96	0.92	0.94	0.89	10259

Metrics formula

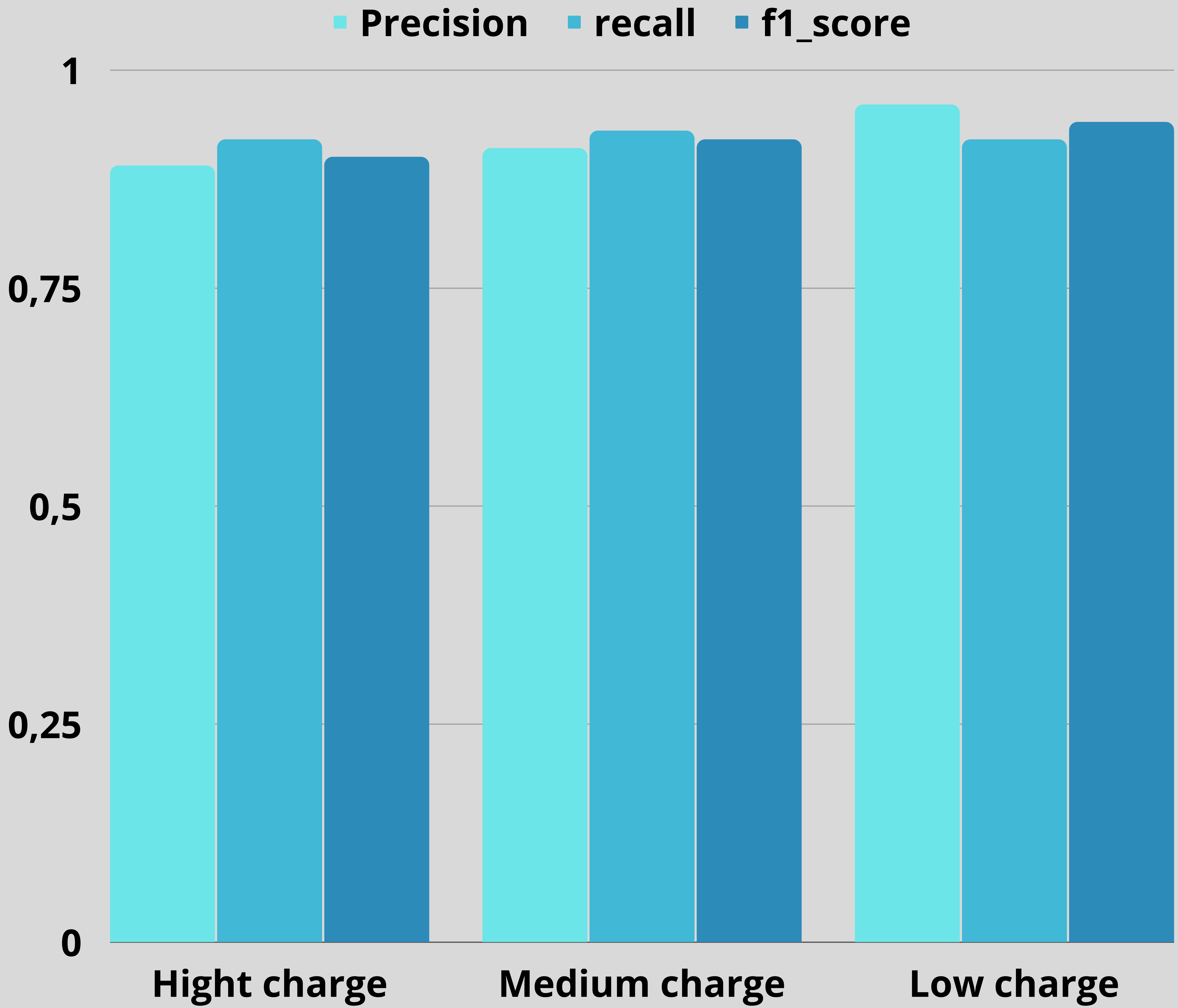
$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}$$

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}$$

$$F_1 \text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Metrics accuracy



Results

- The insurers risk can be segmented in 3 groups according to the charge that they could generate.

- To do prediction for the future risks we can use BalancedRandomForestCalssifier, which provided significant metrics.

Improvements

Lack of information

Measurable Factors

- **Geographical situation**
- **Weather**

Unpredictable Factors

- **Psychological state of the driver**
- **Drunk driven state**
- **Lack of visibility**

Data bases

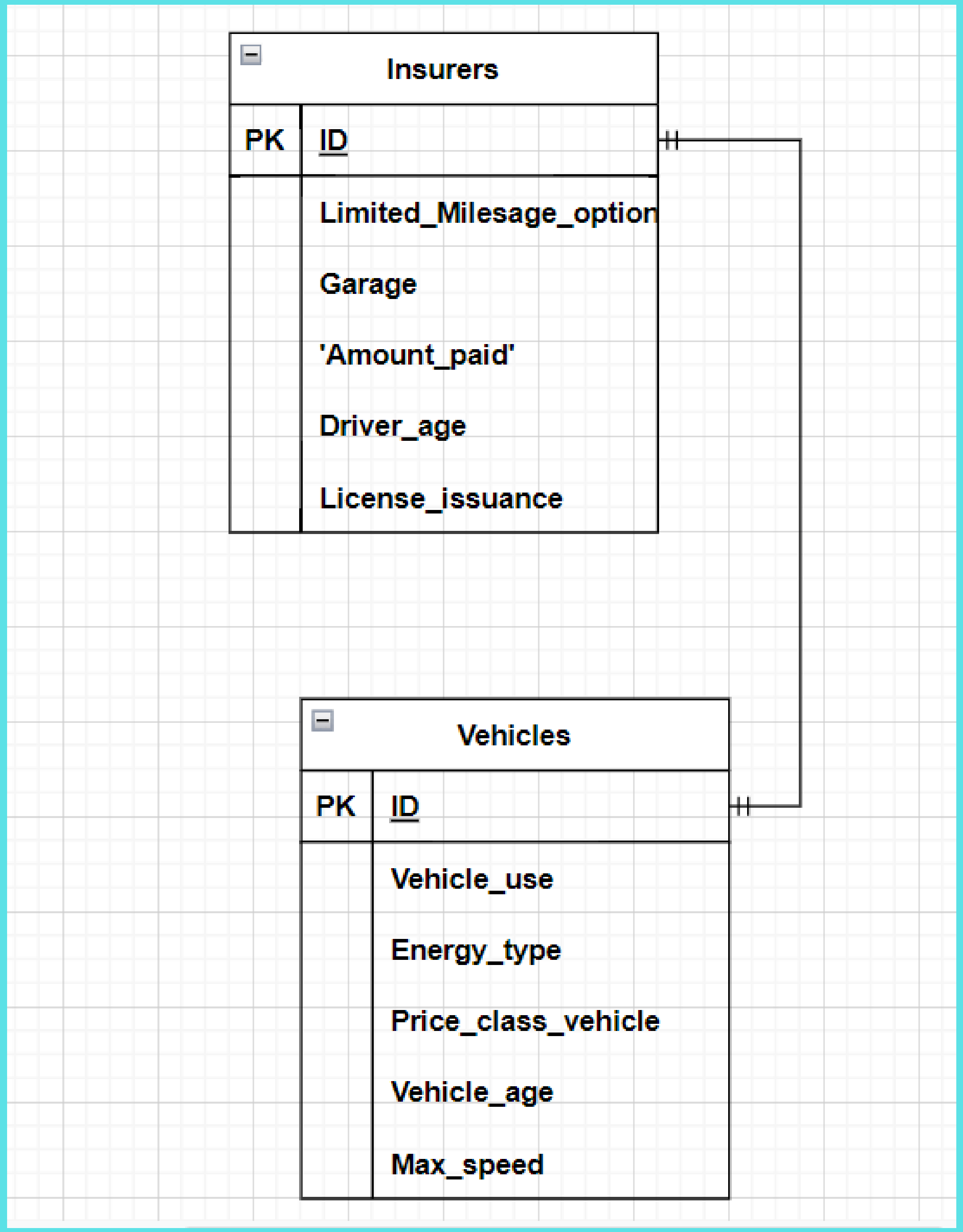
SQL vs NoSQL

Let's define the concepts!

SQL----> **S**tructured **Q**uery **L**angage

NoSQL----> **N**ot **O**nly **S**QL

ERD using draw.io



MySQL

Queries and stored procedures

```sql
SELECT * FROM final_project.vehicle;
select count(Energy_type)  as total_Energy_type_2016,  Energy_type as  Energy_type,  Price_class_vehicle from vehicle
where Year=  2016
group by Energy_type
order by count(Energy_type) ;
```

Result Grid | Filter Rows: | Export:

total_Energy_type_2016	Energy_type	Price_class_vehicle
1	ELECTRIQUE	O A V
6923	ESSENCE	C
7664	DIESEL	H

```sql
select insurer.Sex , vehicle.Garage
from insurer
left join vehicle
on insurer.ID=vehicle.ID
group by Sex;
```

Result Grid | Filter Row

Sex	Garage
HOMME	AUTRES
FEMME	AUTRES

```sql
select insurer.Driver_age, vehicle.Vehicle_use
from insurer
left join vehicle
on insurer.ID=vehicle.ID
group by Vehicle_use
order by Driver_age desc;
```

Result Grid | Filter Rows:

Driver_age	Vehicle_use
31	TOURNEES
21	PRIVE
21	PROFESSIONNEL

```sql
CREATE PROCEDURE `get_insurer_info`()
BEGIN
select * from vehicle;
END
```

```sql
call get_insurer_info;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

ID	Year	NUMCNT	Limited_Milesage_option	Vehicle_use	Energy_type	Vehicle_age	License_issuance	Bonus_malus	Garage	Max_speed	Car_type	Price_class_vehicle
1	2015	2846378304	N	PRIVE	ESSENCE	11	3	8	AUTRES	151	BERLINE	E
2	2015	2846378304	N	PRIVE	ESSENCE	11	3	8	AUTRES	151	BERLINE	E
3	2015	2846378604	N	PRIVE	ESSENCE	11	0	9	AUTRES	151	BERLINE	C
4	2015	2846380204	N	PRIVE	DIESEL	11	7	8	AUTRES	151	BERLINE	H
5	2015	2846380204	N	PRIVE	DIESEL	11	7	8	AUTRES	151	BERLINE	H
6	2015	2846381304	N	PRIVE	ESSENCE	11	0	9	AUTRES	141	BERLINE	D
7	2015	2846381504	N	PRIVE	DIESEL	11	0	9	AUTRES	151	CAMIONNETTE	E

```sql
CREATE PROCEDURE `get_energy_type_info`(in Energy_type char)
BEGIN
select * from vehicle
where vehicle.Energy_type= Energy_type;

END
```

```sql
call get_energy_type_info('ESSENCE');
```

ID	Year	NUMCNT	Limited_Milesage_option	Vehicle_use	Energy_type	Vehicle_age	License_issuance	Bonus_malus	Garage	Max_speed	Car_type	Price_class_vehicle
1	2015	2846378304	N	PRIVE	ESSENCE	11	3	8	AUTRES	151	BERLINE	E
2	2015	2846378304	N	PRIVE	ESSENCE	11	3	8	AUTRES	151	BERLINE	E
3	2015	2846378604	N	PRIVE	ESSENCE	11	0	9	AUTRES	151	BERLINE	C
5	2015	2846381304	N	PRIVE	ESSENCE	11	0	9	AUTRES	141	BERLINE	D
8	2015	2846381604	N	PRIVE	ESSENCE	5	30	3	AUTRES	141	BERLINE	C
9	2015	2846382104	N	PRIVE	ESSENCE	11	25	0	AUTRES	151	COUPÃ´	F
10	2015	2846383004	N	PRIVE	ESSENCE	11	7	2	AUTRES	171	BERLINE	G
11	2015	2846383004	N	PRIVE	ESSENCE	11	7	2	AUTRES	171	BERLINE	G

```sql
CREATE PROCEDURE `insurance_charge_year`( in Year int)
BEGIN
select * from vehicle
where vehicle.Year = Year
group by Garage;
END
```

```sql
call  insurance_charge_year(2015);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	Year	NUMCNT	Limited_Milesage_option	Vehicle_use	Energy_type	Vehicle_age	License_issuance	Bonus_malus	Garage	Max_speed	Car_type	Price_class_vehicle
1	2015	2846378304	N	PRIVE	ESSENCE	11	3	8	AUTRES	151	BERLINE	E
18	2015	2846385904	O	PRIVE	ESSENCE	6	30	0	INDIVIDUEL CLOS	220	CABRIOLET	W A Z5
26	2015	2846390804	O	PRIVE	ESSENCE	0	7	5	CLOS COLLECTIF	171	CABRIOLET	K

END