RideCare Data Science Analysis

Presentation of statistical analysis of collected data

PEILoad

2nd February 2021

TABLE OF CONTENTS

Our methodology

Step 1

Concrete anomaly definition and collection scenario detailed

Step 2

Collect data in significant quantities and in the planned scenario

Step 3

Train ML models to spot new events
Discuss and conclusions

Step 4

Return to ‘Step 1’

Data Visualization

01

Insights about data metrics and data points distribution.

Data Visualization Roadmap

Dataset description

Studying mean, percentiles, min and max values,etc

Data Correlation

Understanding the correlation and importance of the various columns

Best columns Plots

Best columns analysis and plot

Initial Contextualization

Notes:

- At this stage, we are handling with 3 situations: **normal**, **smoke** and **stink**;
- The possibility of first distinguishing the odour in the particles, then distinguishing between bad and good odour.

```
Total Number of elements: 9918
--> Normal situation elements: 8777
--> Anomalous situation elements: 1141
```

At this moment:

- Almost 10 000 cases collected
  - 8777 for normal situation
  - 1141 for anomalies inside vehicle

Dataset Report – Normal Situation

	sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas	sensors.humidity	sensors.pressure	sensors.altitude
count	8777.000000	8777.000000	8777.000000	8777.000000	8777.000000	8777.000000	8777.000000
mean	5.152410	10.401527	19.732627	96936.973453	55.625551	995.612810	148.357316
std	3.979411	6.264248	3.902937	49005.702222	11.781887	11.691793	98.805466
min	0.000000	0.000000	10.512617	2971.000000	25.600421	955.262052	-81.183173
25%	2.400000	5.500000	16.013789	55702.000000	49.388772	991.632661	105.645401
50%	4.300000	9.000000	20.495430	99929.000000	52.022272	994.790689	154.832518
75%	6.400000	13.600000	22.717305	139272.000000	66.592563	1000.625071	181.553980
max	41.600000	52.700000	27.949531	404575.000000	84.871095	1023.038997	494.377666

Notes:

- Very **Low** pm's values;
- The values in the gas column are very **high**.

Dataset Report – Anomalous Situation

	sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas	sensors.humidity	sensors.pressure	sensors.altitude
count	1141.000000	1141.000000	1141.000000	1141.000000	1141.000000	1141.000000	1141.000000
mean	309.261700	626.679842	18.748447	42651.442594	60.381668	999.629140	114.554295
std	293.245202	645.917091	3.315124	21984.508079	7.756193	12.453751	104.938890
min	2.700000	8.700000	7.844306	3841.000000	38.051415	967.816309	-45.444814
25%	35.600000	56.100000	16.932344	24673.000000	55.181451	987.973221	16.188072
50%	213.100000	457.500000	18.610859	37799.000000	59.235854	992.727919	172.278727
75%	510.400000	1083.800000	20.132344	65632.000000	63.503448	1011.307156	212.604422
max	999.900000	1999.900000	32.225238	112731.000000	88.338576	1018.720306	385.325117

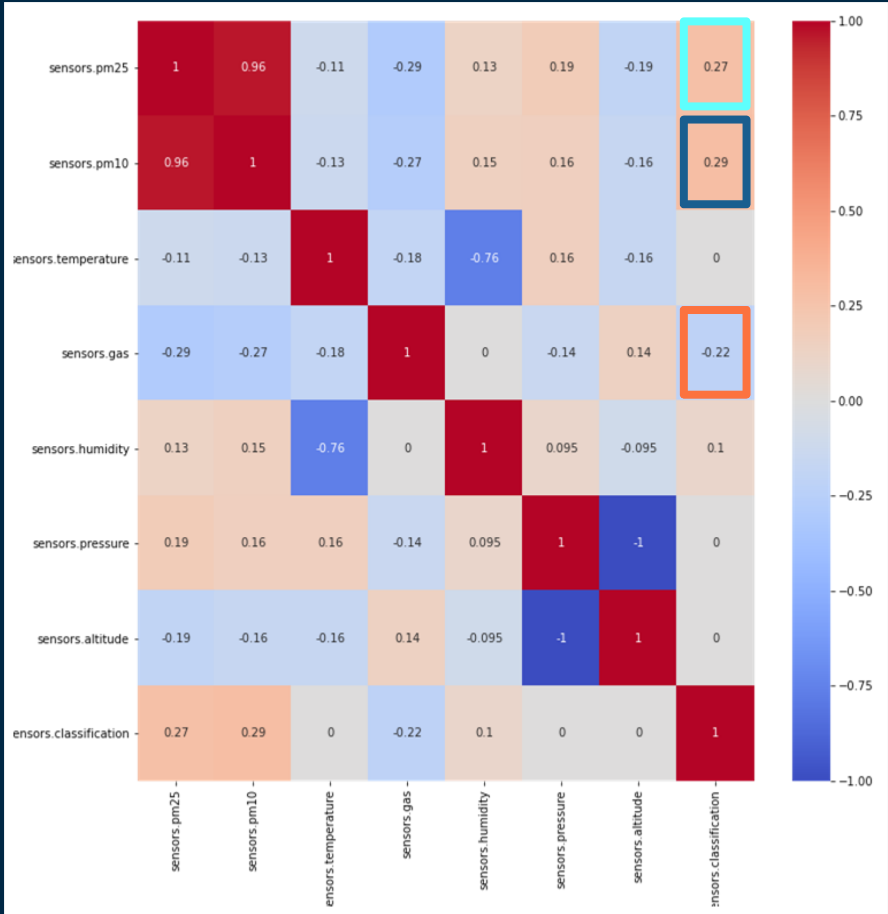
Notes:

- The first values we noticed was the **high** in pm25 and pm10 columns;
- The values in the gas column are **almost half** the values in normal situations.

Data Correlation

Comparing to the heatmap of stage 1, the correlation values **decreased**;
Even so, we can see that the most influential columns are the same as in the previous phase:

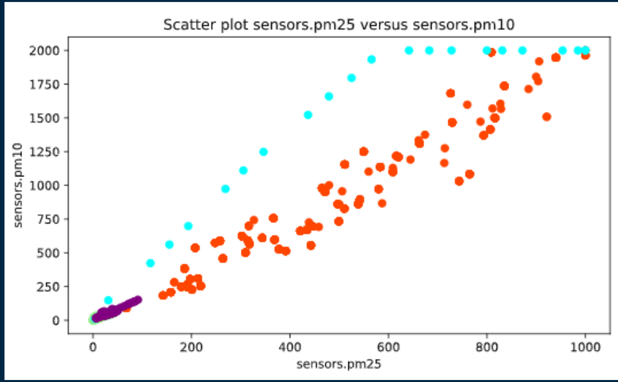
- pm25;
- pm10;
- gas;

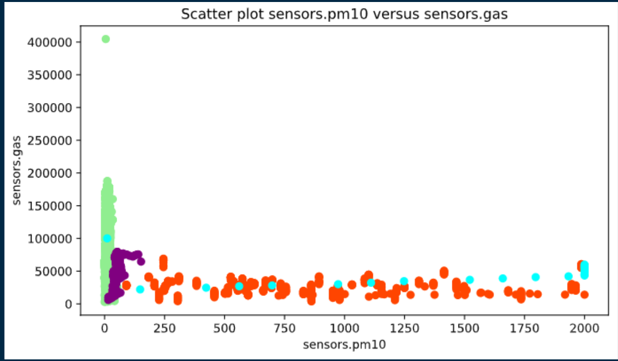


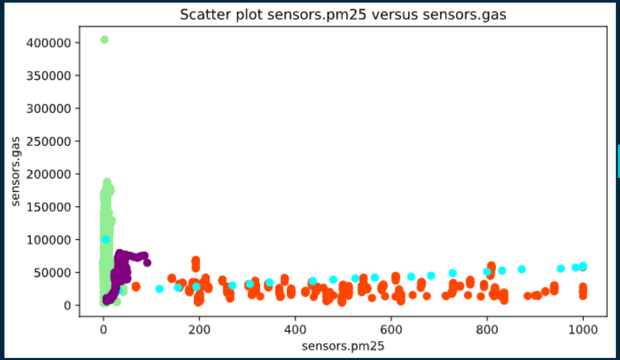
Best columns Plots

The best columns are as seen in presentation:

- pm25;
- pm10;
- gas;







Data Preprocessing

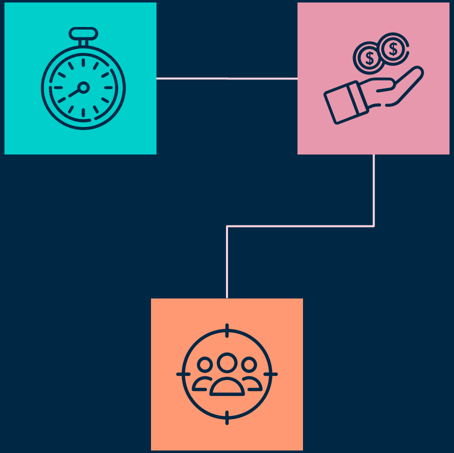
02

Data processing/transformation
and analysis techniques

Roadmap Data Preprocessing

Data Structure

Organize data in order to prepare it for models training



Data quality assurance

Drop *missing values*, NaN, *Outliers Analysis*,etc

Handling Miscaptures

Hardware limitations and removal of invalid captures

Data Preprocessing – *Columns Selection*

	sensors.id	sensors.carId	sensors.carLocation	sensors.timeValue	sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas
0	553	66-ZZ-66	41.5608 -8.3968	2020-12-02 23:54:53	9.5	13.0	21.639570	21515
1	552	66-ZZ-66	41.5608 -8.3968	2020-12-02 22:24:04	16.5	24.7	20.131758	34165
2	551	66-ZZ-66	41.5608 -8.3968	2020-12-02 22:23:53	16.1	25.5	20.107344	33695
3	550	66-ZZ-66	41.5608 -8.3968	2020-12-02 22:23:43	16.0	26.4	20.089375	33113
4	549	66-ZZ-66	41.5608 -8.3968	2020-12-02 22:23:32	15.9	24.4	20.071992	32696

sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas	sensors.humidity	sensors.pressure	sensors.altitude
9.5	13.0	21.639570	21515	43.570138	982.296484	260.956467
16.5	24.7	20.131758	34165	46.910041	982.307568	260.861841
16.1	25.5	20.107344	33695	46.782896	982.325957	260.704846
16.0	26.4	20.089375	33113	46.737320	982.321309	260.744524
15.9	24.4	20.071992	32696	46.847113	982.322707	260.732590

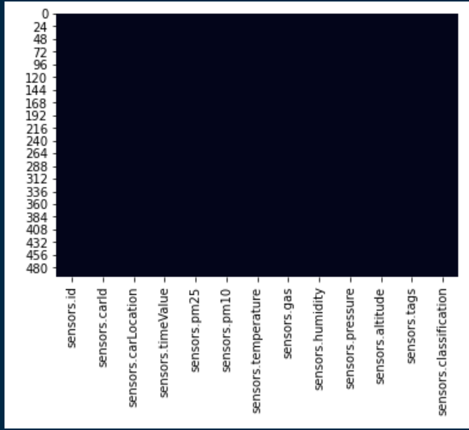
Notes:

- Discard unnecessary columns such as id, cardID, carLocation, timeValue
- Model training preparation ->just important columns selected

- PM25, PM10, Gas and Humidity are the most important *columns*, as shown previously

Data Preprocessing – Wrong Data Removal

```
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   sensors.id            498 non-null    int64
 1   sensors.carId         498 non-null    object
 2   sensors.carLocation   498 non-null    object
 3   sensors.timeValue     498 non-null    object
 4   sensors.pm25          498 non-null    float64
 5   sensors.pm10          498 non-null    float64
 6   sensors.temperature   498 non-null    float64
 7   sensors.gas           498 non-null    int64
 8   sensors.humidity      498 non-null    float64
 9   sensors.pressure      498 non-null    float64
 10  sensors.altitude      498 non-null    float64
 11  sensors.tags          498 non-null    object
 12  sensors.classification  498 non-null    int64
dtypes: float64(6), int64(3), object(4)
memory usage: 54.5+ KB
```

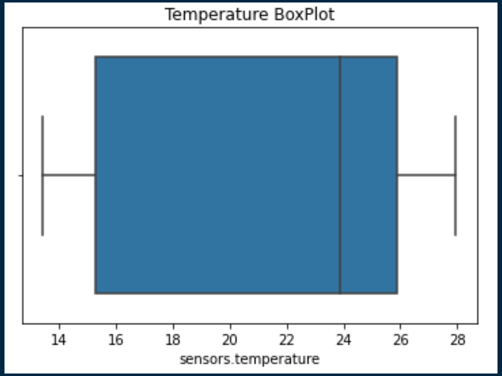


Notes:

Dataset informations

- Removal of unwanted intruders, such as NaN, *missing values, etc*
- Data types are mostly float or integer, which will improve the training process
- Final data does not show any bad value or missing value

Analysis and Outliers



Additionally, two methods were used: **Z-score** e **IQR**.

Notes:

- Z-score method with threshold = 3 (reference value)
- IQR will remove data points which are beyond Q25 and Q75

However....

- **Regular outliers detection methods** can **influence performance** since:
  - Data is collected in a **known** and **controlled** environment
  - **Outliers** are supposed to be **found** by **algorithms** (anomaly detection models)

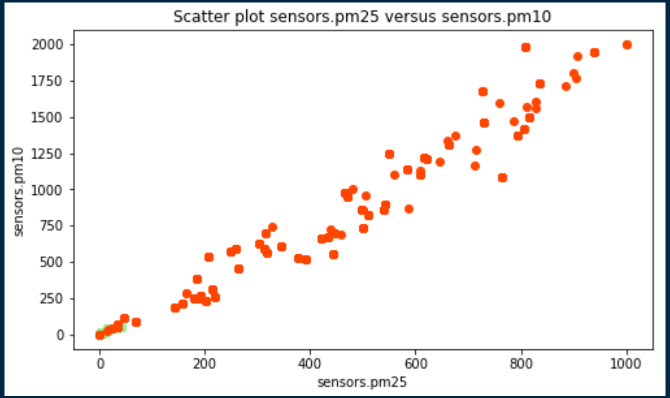
So....

- **Keep data variability**, **not** proceeding to **remove** outliers in a typical way

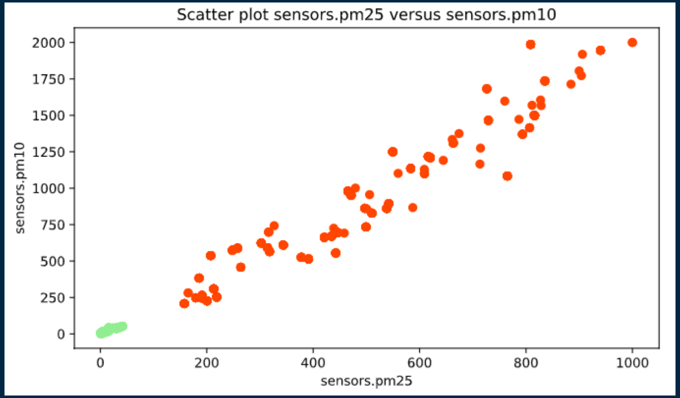
Bad Captures Corrections

There are anomaly values around "**0**", due to bad captures from sensor. The solution is to remove them, in order to avoid models bias

- Bad caputes are deleted
- Gap between "Normal scenario" and "Smoking scenario" is increased, looking for better classification results and keeping data variability



Before



After

Algorithms 03

Implemented Algorithms

- Supervised learning

1. SVM (Support Vector Machine)
2. Neural Network
3. Random Forest
4. Naive Bayes

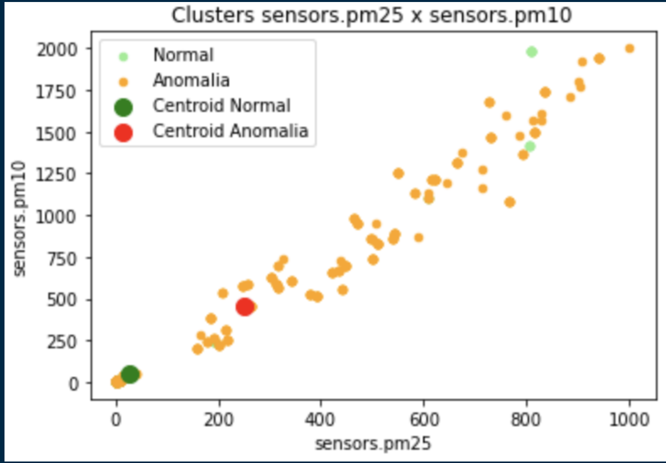
- Unsupervised learning

1. K-means clustering
2. Local Outlier Factor
3. Isolation Forest
4. One-Class SVM

- 70% training data, 30% test data

K-means clustering

- Implementation using the library scikit-learn



K-means Clusters: pm25 & pm10

```
[[328 123]
 [  9 132]]
              precision    recall  f1-score   support

           0       0.97      0.73      0.83       451
           1       0.52      0.94      0.67       141

    accuracy                           0.78       592
   macro avg       0.75      0.83      0.75       592
weighted avg       0.86      0.78      0.79       592
```

Best result obtained - K-means

Summary and Conclusions

- All supervised algorithms have excellent results.
- Unsupervised algorithms get worse results, mainly due to the variability of the data.

Implemented Algorithms

- Supervised learning

1. SVM (Support Vector Machine)
2. Neural Network
3. Naive Bayes
4. Random Forest

- 70% training data, 30% test data

SVM (Support Vector Machine)

- Implementation using the library scikit-learn
- Use of the Grid Search tuning technique to optimize hyperparameters

```
Best parameters =>  {'C': 0.1, 'gamma': 1, 'kernel': 'linear'}
```

Best result obtained - SVM

```
Classification Matrix :
[[138   0   2]
 [  0 130   0]
 [  0   0 117]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       140
           1       1.00      1.00      1.00       130
           2       0.98      1.00      0.99       117

    accuracy                           0.99       387
   macro avg       0.99      1.00      0.99       387
weighted avg       0.99      0.99      0.99       387
```

Best parameters obtained - SVM

Neural Network

- Implementation using the library scikit-learn - MLP

```
{'activation': 'tanh',
 'alpha': 0.05,
 'hidden_layer_sizes': (128, 64, 16, 3),
 'learning_rate': 'constant',
 'solver': 'adam'}
```

Best parameters obtained - NN

```
Classification Matrix :
[[ 77   3  60]
 [  0 110  20]
 [  0   3 114]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.55      0.71       140
           1       0.95      0.85      0.89       130
           2       0.59      0.97      0.73       117

    accuracy                           0.78       387
   macro avg       0.85      0.79      0.78       387
weighted avg       0.86      0.78      0.78       387
```

Best result obtained - NN

Random Forest

- Implementation using the library scikit-learn

```
{'bootstrap': True,
 'max_depth': 10,
 'max_features': 'auto',
 'n_estimators': 40}
```

Best parameters obtained - RF

```
Classification Matrix :
[[138   0   2]
 [  0 130   0]
 [  0   0 117]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       140
           1       1.00      1.00      1.00       130
           2       0.98      1.00      0.99       117

    accuracy                           0.99       387
   macro avg       0.99      1.00      0.99       387
weighted avg       0.99      0.99      0.99       387
```

Best result obtained - RF

Naive Bayes

- Implementation using the library scikit-learn

```
{'var_smoothing': 1e-07}
```

Best parameters obtained - NB

```
Classification Matrix :
[[135   0   5]
 [  0 130   0]
 [  3   0 114]]
Classification Report :
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       140
           1       1.00      1.00      1.00       130
           2       0.96      0.97      0.97       117

    accuracy                           0.98       387
   macro avg       0.98      0.98      0.98       387
weighted avg       0.98      0.98      0.98       387
```

Best result obtained - NB

Summary and Conclusions

- With the excellent results obtained in supervised models, it's concluded that it's possible to distinguish and detect anomalies from smoke and/or bad odor.

Model	Accuracy	Precision
SVM	0.99	0.99
Neural Network	0.78	0.86
Random Forest	0.99	0.99
Naive Bayes	0.98	0.98

AlertAI Cloud

04

AlertAI Cloud - Details

- Flask based REST application, hosted by Cloud, in order to **provide alternative classification** for raw data

- Designed and built for further research work and system progress

- Includes extra supervised:
  - Gaussian Naive Bayes
  - Neural Network - Sklearn version and Customized version
  - Support Vector Machine

- Exposes two different endpoints:
  - /captures -> receive raw data for new classifications
  - /models -> give alternative classification for specific record

RideCare Data Science Analysis

Presentation of statistical analysis of collected data

PEILoad

2nd February 2021