EISTI - ALTRAN

# Final

FEATURE SELECTION FOR LANE CHANGING

ARTIFICIAL INTELLIGENCE - END OF STUDIES PROJECT

2019 - 2020

CHABNI Chabane - CISSE Babacar - NOREL Arnaud - RIGAUX David - SEBBAG Ashley

November 18, 2020

ACKNOWLEDGEMENTS

We want to use this section to express our gratitude to ALTRAN who gave us the opportunity to build this project. We would like to give our special thanks to Ms Rim El Ballouli and Mr Hussein Srour who divised this project and followed our every step to help us produce a project we can be proud of.

We also wish express our gratitude to our engineering school Eisti, especially to the professors Mr Julien Mercadal and Ms Souhila Arib who gave us time to oversee this project and advices to conclude it.

# Contents

# 1 Introduction

***Index Terms*—Autonomous vehicles, Feature selection, Lane changing**

The act of a driver changing lane comes after he calculates all the variables he sensed through his hearing and his seeing. The amount of data treated by the human brain is too voluminous for AI algorithms to process, it is necessary to scale these incoming data to be able to perform a lane change detection. A way to realize that task is to perform feature selection methods on the data used, allowing us to keep only the most relevant features since all features don't have the same importance on the data set. In this paper we use the following definition for lane change detection: "Lane change detection is fundamentally the act of detecting/predicting that a certain driver will change lane, at a certain moment in time, which direction, and when he arrives in the new lane."

Regarding lane change detection, being able to evaluate and select features is fundamental, giving place to a lot of improvement. Quantifying the contribution of each feature and keeping only the most important ones will allow a better prediction rate and less false positives by tuning the complexity of the method. Selecting the best features is also a way of preventing potential model overfitting in an environment with a multitude of configurations.

This project aims at reducing as much as possible the number of features used by our algorithm to assess whether or not an autonomous vehicle changes lane. To perform this feature reduction many paths have been explored; going from filter methods to wrapper methods after taking a glimpse at how embedding methods work.

## 1.1 State Of The Art

Past studies suggest that for the detection of lane changes by other drivers, only measurable information from the outside should be used. Thus, immeasurable features from the outside, such as steering angle, eye movement, and head motion [1, 3, 5, 7], should be excluded.

A detection method only using measurable information from the outside (e.g., lateral position, lateral velocity, and relative velocity) has been suggested by Schlechtriemen et al. [11]

Furthermore, Mandalia and Salvucci suggested the use of the variance of the lateral position within a constant window size as a feature [8].

Hanwool Woo et al. has proposed a method decreasing the number of false alarms, to predict a vehicle's trajectory and use it for lane change detection. Unlike previous methods, that only use past measurements until the current time, they focus on how human drivers predict a trajectory unconsciously. [12]

As of today, many studies have been conducted on simulators using CAN-bus signals. For example, Pentland and Liu [9] recognized driver lane change maneuver based on the recorded acceleration, braking, and steering data, and achieved an accuracy of 93.3% over the recorded 47 lane change episodes. Another example is the study led by Hou et al. [4], who adopted three signals (lateral acceleration, steering wheel angle rate) to classify lane-keeping and lane-change maneuvers. Their recognition rate for left lane change was 95.5%. However, in reality, the driving environment is more complex than simulated situations, which leads to more random driver behavior.

Guofa Li et al. [7] proposed an enhanced method to recognize lane change maneuvers in naturalistic highway driving situations via features exclusively extracted from vehicle state and driver operation signals. This method uses the Sequential Forward Floating Selection (SFFS) Algorithm as a baseline method for feature selection.

In the method proposed by Guofa Li et al. [7], when using the selected optimized feature set, the authors were able to obtain a recognition rate across all maneuver type of 90.8%, improved by 18.7% compared to the recognition rate when using the original CAN-bus signals. Digging deeper into the result, the recognition rate of lane-change maneuvers achieves 88.2%, improving by 7% compared to the result reported in [10].

The method proposed by Ürun Dogan et al. [2], based on evaluating models on different sets of features, allowed the authors to find the best combination of features and their importance. The most important features are Lane Offset, Steering Angle, and TTC, which decreases the number of false positives and false negatives. However, adding these features increases the prediction time. The method showed that it is possible to recognize the lane change maneuver at least 1s and up to 1.5s before the center of the ego vehicle passes the lane's borders.

The approach used by Puneet Kumar in [6] is based on the combination of a Support Vector Machine (SVM) to perform the classification between the three classes: left lane change, no lane change, and right lane change. Furthermore, a Bayesian Filter (BF) is employed to smooth the results, thus, improving the reliability of the predictions.

This is used after selecting the following features:

- the lateral position of the vehicle (l)
- the steering angle of the vehicle ($\phi$)
- the first derivative of l
- the first derivative of $\phi$

Those features being selected due to their different patterns for each of the three classes.

## 2 Dataset

The dataset provided to us by Altran is based on the Next Generation Simulation (NSGIM) from the U.S Department of Transportation (dataset source). Our dataset has been reworked, and some calculations have been done to extract the information we are interested in. containing a total of 8580 entries divided into 23 features. These features can be categorized into four different groups: Velocity, Lateral Distance, Longitudinal Distance, and Inverse Time to Collision (ITTC).

The information given by each entry does not only concern one car but a total of 8 cars. Indeed, the other cars are cars around the ego car, which is the main car. Here is a graphical representation of the situation making things easier to understand.

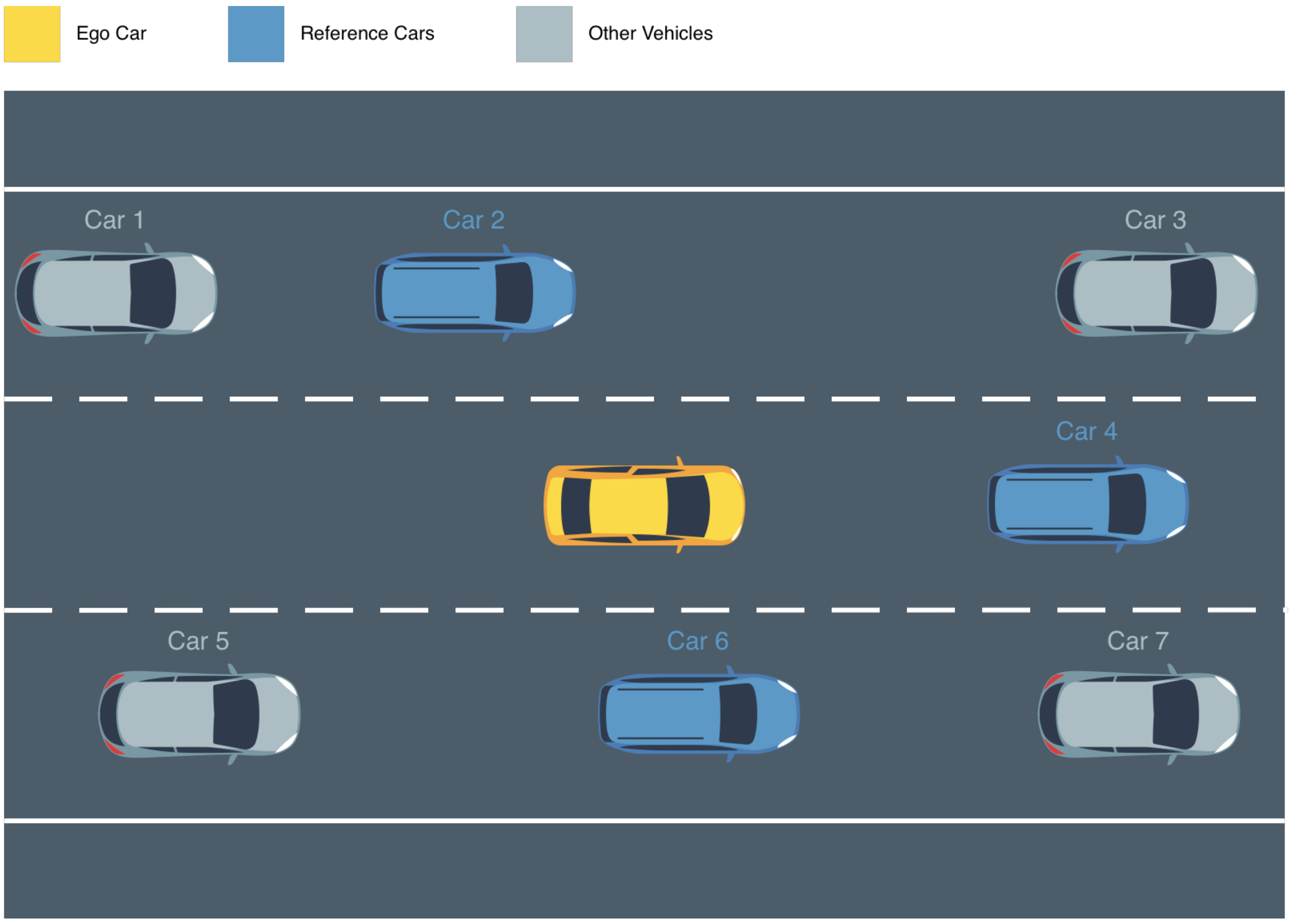

Figure 1: Graphical Representation of the situation

Now, let's dig deeper into the 23 features and explain them via tables.

## 2.1 Velocity Features

There are a total of 8 features describing the vehicles' velocity:

| Feature | Description |
|---|---|
| v_Vel | Velocity of ego car |
| v_Vel_Ref1 | Velocity of reference car 1 |
| v_Vel_Ref2 | Velocity of reference car 2 |
| v_Vel_Ref3 | Velocity of reference car 3 |
| v_Vel_preceed1 | Velocity of the preceeding car 1 |
| v_Vel_preceed2 | Velocity of the preceeding car 2 |
| v_Vel_follow1 | Velocity of the following car 1 |
| v_Vel_follow2 | Velocity of the following car 2 |

Table 1: Velocity Features

## 2.2 Lateral Distance Features

There are a total of 3 features describing the vehicles' lateral distance:

| Feature | Description |
|---|---|
| lateral_current_lane | Lateral distance between the center of the ego car and the lane center |
| lat_pos_vehicle1 | Lateral distance between the center of the reference car 1 and the ego car |
| lat_pos_vehicle2 | Lateral distance between the center of the reference car 2 and the ego car |

Table 2: Lateral Distance Features

## 2.3 Longitudinal Distance Features

There are a total of 7 features describing the vehicles' longitudinal distance:

| Feature | Description |
|---|---|
| longit_pos_vehicle1 | Longitudinal distance of the ego car |
| longit_pos_vehicle2 | Longitudinal distance of the reference car 1 |
| longit_pos_vehicle3 | Longitudinal distance of the reference car 2 |
| longit_pos_follow1 | Longitudinal distance of the preceding car 1 |
| longit_pos_follow2 | Longitudinal distance of the preceding car 2 |
| longit_pos_preced1 | Longitudinal distance of the following car 1 |
| longit_pos_preced2 | Longitudinal distance of the preceding car 2 |

Table 3: Longitudinal Distance Features

## 2.4 Inverse Time to Collision (ITTC) Features

Last but not least, there are a total of 5 features describing the vehicles' ITTC:

| Feature | Description |
|---|---|
| iTTC_preced1 | Inverse of time to collision for the preceding car 1 |
| iTTC_preced2 | Inverse of time to collision for the preceding car 2 |
| iTTC_follow1 | Inverse of time to collision for the reference car 1 with the follow car 1 |
| iTTC_follow2 | Inverse of time to collision for the reference car 2 with the follow car 2 |
| iTTC_ref3 | Inverse of time to collision for the ego car with the reference car 3 |

Table 4: ITTC Features

## 2.5 Data Exploration

To better understand the data we have, we are going to explore it. We are going to start by checking out the target data we have.

Here is the distribution of the target time:

| Time | Occurrences |
|---|---|
| 5.0 | 572 |
| 4.0 | 2860 |
| 1.0 | 572 |
| 0.9 | 572 |
| 0.8 | 572 |
| 0.7 | 572 |
| 0.6 | 572 |
| 0.5 | 572 |
| 0.4 | 572 |
| 0.3 | 572 |
| 0.2 | 572 |

Table 5: Target Time Occurrences

We can see that there are a total of 11 different time values. Furthermore, the data is equally shared between all 11 values, for the exception of the value of 4.0 who has 5 times more entries than the other values.

Now let's have a look at the distribution for the target classes.

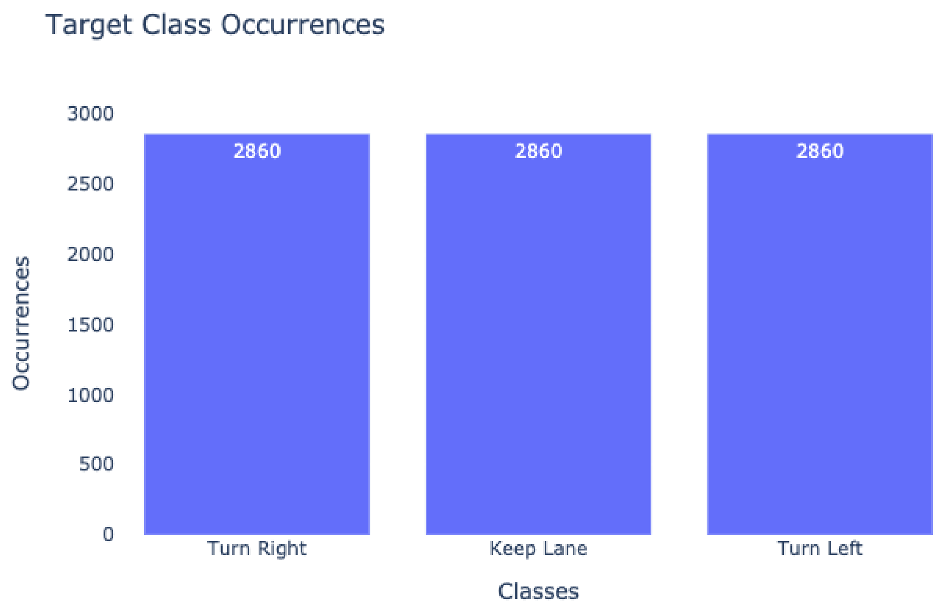

Figure 2: Target Class Occurrences

As we can see thanks to Table 5 and the figure 2, the data is in majority balanced, in exception for the Target Time value of 4.0 which has 5 times more occurrences than all the others. For the rest, they are evenly balanced.

## 3 Project Organisation

Before getting to the heart of the matter, it's important to understand how our project took shape. We had chance to receive rigorous monitoring from Altran. Indeed, each week we had an appointment with the company to take stock of the progress of the project and redefine the next step. Depending on the point to be treated, we had to provide the results of our work. These results could be a simple presentation, a report or even code. This organization allowed us to experiment with the research methodology. So the project was built around 5 assignments.

### 3.1 Assignment 1

For the first mission, we received a set of carefully selected scientific articles addressing the problem of trajectory prediction and lane change detection. The aim was to extract all the features used in each article and describe them.

### 3.2 Assignment 2

For the second mission, we had to explore the different strategies of feature selection in Machine Learning. Feature selection is divided into three different classes: filter, wrapper, and embedded methods.

### 3.3 Assignment 3

In this assignment, we had to implement some filtering methods, techniques used to select entities and is generally the first step in the feature selection process. We had to provide a python file containing methods that we will discuss later. As well as an Excel sheet containing the summary of the list of features selected and removed after having executed each filtering method separately.

### 3.4 Assignment 4

This assignment allowed us to conclude the study on filtering techniques. After having created several feature sets in assignment 3, it was now time to measure their performances. We were asked to create functions allowing us to conclude on which feature set was optimal. To obtain the metrics, we built random forest models. The first feature set on which we worked was the original set of features allowing us to have a base.

### 3.5 Assignment 5

The goal of this final assignment was to implement feature selection, based on the optimal feature set found in the previous assignment, using wrapper methods to try and find an even better feature set. We implemented the Sequential Feature Selection (forward and backward), and the Exhaustive Feature Selection.

## 4 Filter Techniques

Adequate selection of features may improve accuracy and efficiency of classifier methods. To start our feature selection, we applied the filter techniques which is one of the most successful approach for feature selection. Filter technique select features from a dataset independently for any machine learning algorithm. These methods rely only on the characteristics of these variables, so features are filtered out of the data before learning begins. These methods are powerful and simple and help to quickly remove features and they are generally the first step in any feature selection pipeline.

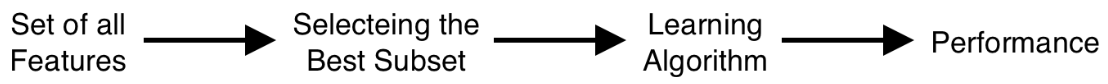

Figure 3: Filter Techniques Workflow

First we will talk about feature set and the creation of filter methods, then, with regard to performance, we will evaluate these feature sets with metrics.

## 4.1 Filter Methods

In this part we will talk about the six filter techniques that we have studied and then experiment to evaluate their results on our data set of 23 features.

### 4.1.1 Constant

**Theory** Constant features are the type of features that contain only one value for all the outputs in the dataset (zero variance). Constant features provide no information that can help in the classification of the record at hand. Therefore, it is advisable to remove all the constant features from the dataset.

**Method** To remove constant features we used VarianceThreshold function imported from sci-kit-learn. When passing a value of zero for the parameter we had filtered features with zero variance. After fitting the threshold function, the function enables us to see which features have been removed and which ones have been kept.

**Results** By using this method, 0 features were removed. Even though no features were removed when applying this technique, it is a must-do when performing feature selection.

### 4.1.2 Quasi-Constant

**Theory** Quasi Constant Removal is a filter method removing the almost constant features. In other words, a quasi-constant feature has a very large number of identical values. Such features are not very useful for making predictions. There is no rule as to what should be the threshold for the variance of quasi-constant features. However, as a rule of thumb, remove those quasi-constant features that have more than 99% similar values for the output observations.

**Method** This filter method works like the previous one. We used the same function named VarianceThreshold. This time, we tried different thresholds by applying them in a ”for” loop and we observed the results of each threshold.

**Results** Having no features removed after the constant removal filter, it was important for us to be sure no features were quasi-constant as well. We applied 4 thresholds: 85,90,95 and 99 to our features but for each of them, 0 features were removed.

### 4.1.3 Duplicates

**Theory** Duplicate features are features that have similar values. The duplicate removal technique is used to remove features that do not add any value to algorithm training, rather they add overhead and unnecessary delay to the training time.

**Method** To deal with this method and remove the duplicates features we just used pandas and NumPy. We processed our features data frame as a matrix by transposing it. After that, we used the "duplicated" function from pandas to detect any duplicates features as if they were rows from a data frame.

**Results** No features from the dataset have been removed in this step, but after those three methods, we were sure that our features were clean.

### 4.1.4 Information Gain

**Theory** In a decision tree, the information gain is theoretically the subtraction of the parent's entropy by the weighted average of the children's entropy. The information gain might be being used interchangeably with mutual information.

$$IG(X, Y) = H(X) - H(X|Y)$$

This filtering technique aims to measure the quality of one or one other features then to estimate the best subset of features.

**Method** What we did to estimate the features' quality and render a subset of features, is create the functions rmv_mutual_info_gain_classification and rmv_mutual_info_gain_regression which takes as input a parameter K (integer) and which returns as output the K best features according to the information gain for classification or regression.

In this part, we used the functions of mutual_info_regression and mutual_info_classif from Sci-Kit-Learn which measures the dependency between variables. These functions return an array with values between 0 and 1 corresponding to the quality of the feature joined with others. We then passed our data to the function (y_train for classification and y_train_time for regression) and sorted the results in the descending order of the mutual information gained values to keep the K best features.

**Results** For example, after having applied classification or regression with K=5 on the original dataset we can easily see on these charts which features will be kept and which one will be removed.

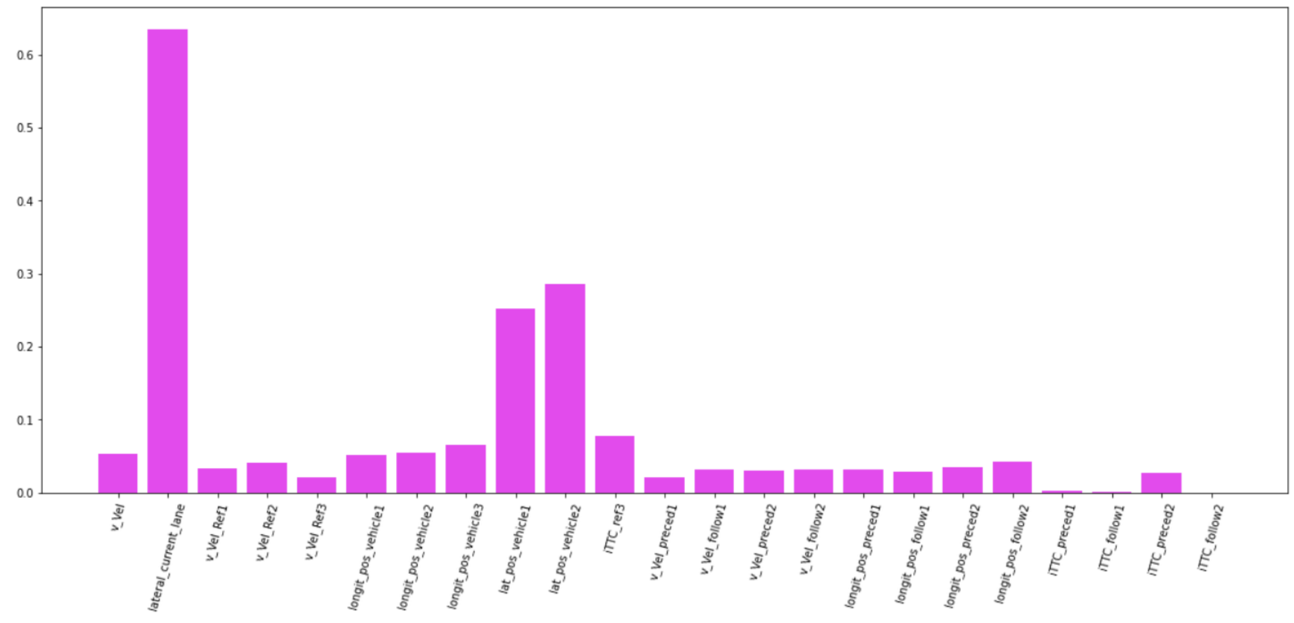

Figure 4: Information Gain with regression and K=5

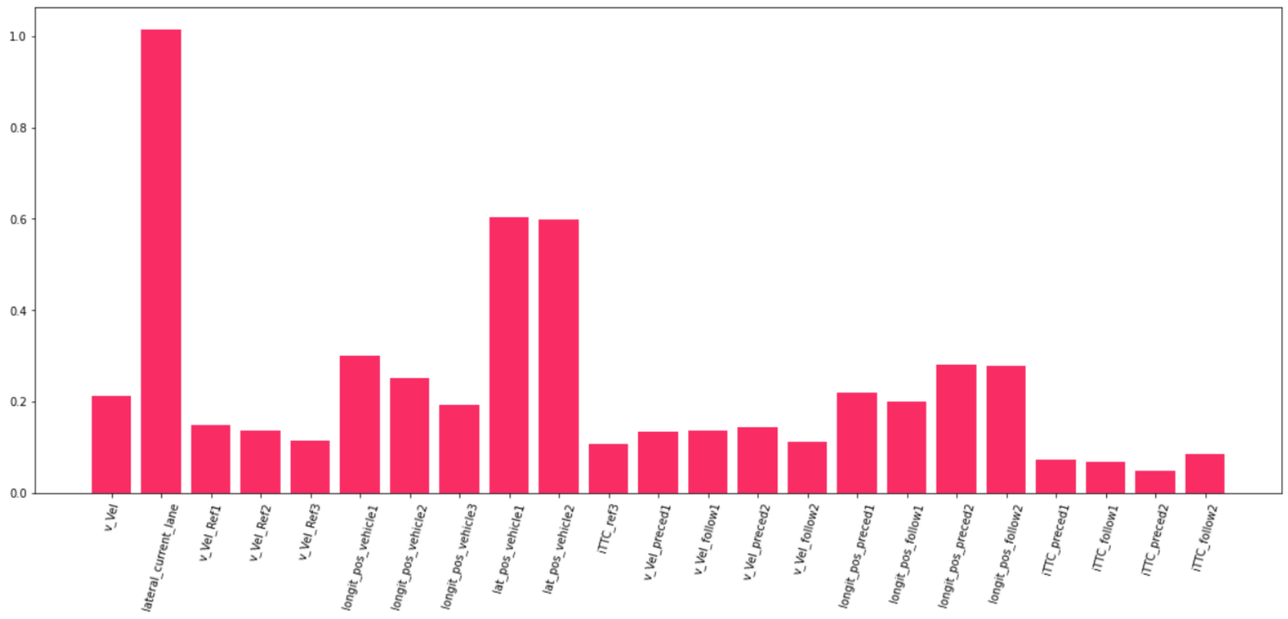

Figure 5: Information Gain with classification and K=5

We can see more in details these results by referring to the table 22

### 4.1.5 Anova

**Theory** The ANOVA is a statistical test analyzing the variance to ensure if the means of the different groups originated from the same population. To perform this test a few assumptions need to be made:

- Each group sample is drawn from a normally distributed population
- All populations have a common variance
- All samples are drawn independently of each other
- The observations are sampled randomly and independently of each other
- Factor effects are additive

**Method** To compute the anova we compute the ols() from the library statsmodels for each feature. The ols function will calculate the importance of the feature on the model.

**Results** Using this method to perform the features selection, we encounter a situation due to the high variance and the fact that the data are not equally distributed between the three classes. This implies that the ANOVA is not reliable in this situation for most of the features, deleting them. For the features having a good distribution as well as having a narrow variance between the three groups, a threshold of 0.01 is not enough to keep any of them. This brings to the conclusion that ANOVA doesn't consider any individual feature being significant in the model.

### 4.1.6 Correlation

Other than having duplicates, constants, quasi-constants, and so on, a dataset can also include correlated features.

Correlation is a measure of the linear relationship between two quantitative variables. In other terms, correlation is a measure of how strongly one variable depends on another.

In some cases, a high correlation can be a useful property. If two variables are highly correlated, we can predict one from the other. Therefore, we generally look for features that are highly correlated with the target, especially for linear machine learning models.

However, if two variables are highly correlated among themselves, they provide redundant information in regards to the target. Essentially, we can make an accurate prediction on the target with just one of the redundant variables.

In these cases, the second variable doesn't add additional information, so removing it can help to reduce the dimensionality and also the added noise.

Here is the original features' correlation matrix.

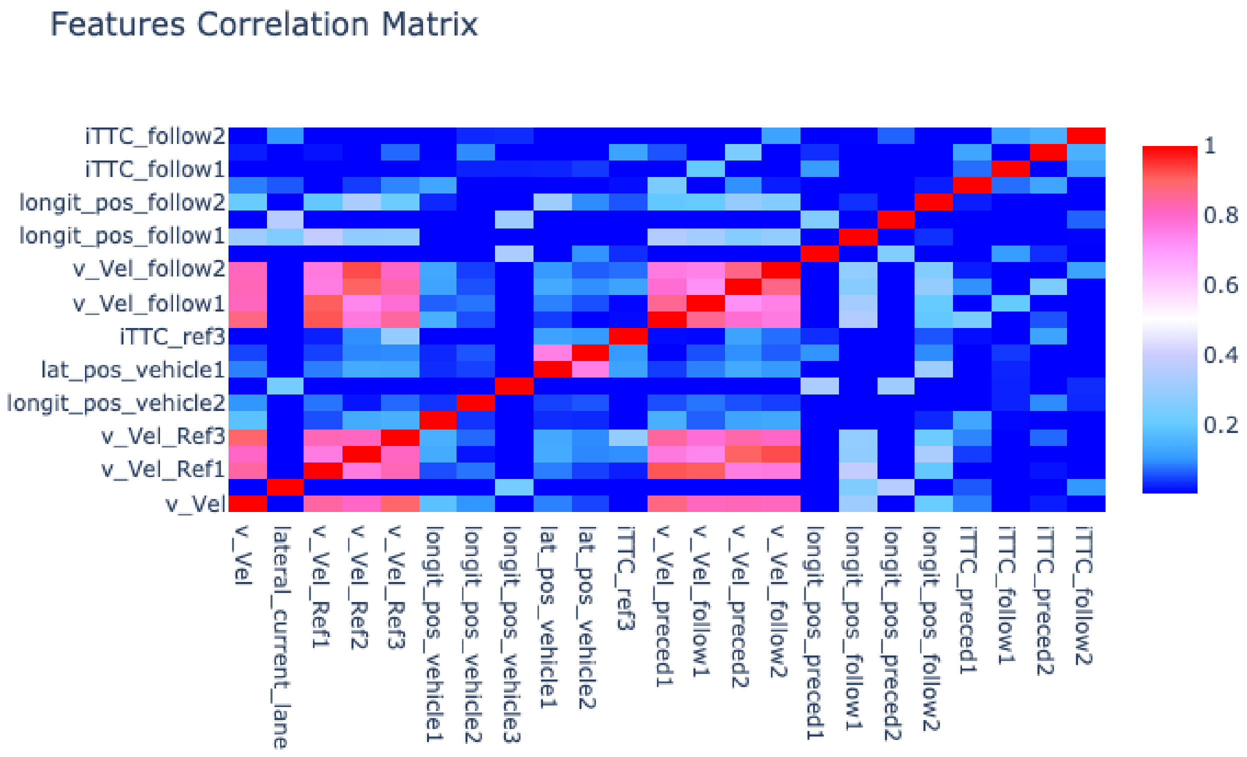

Figure 6: Features Correlation Matrix

We can notice some redish/pink rectangles, meaning that there are some features who are correlated with one another.

To apply the Correlated Features feature selection technique, we proceeded by doing the following four steps:

1. Create groups of features having a correlation coefficient of more than a given threshold
2. Create a Random Forest Classification model for each correlation groups
3. Extract the feature importance of these models
4. Only keep one feature from each group and remove all other features present in the correlated groups from the dataset

Regarding the selection of the threshold value, we decided to create an array containing six different thresholds (0.25, 0.50, 0.75, 0.85, 0.90, 0.95). The choice of having more thresholds closer than 1 to 0 is because we are interested in features that are more correlated with each other than not.

To give a full example of how the step four goes down, we will take the groups obtained from the threshold of 0.85:

| Groups | Features | Feature Importance |
|---|---|---|
| Group 1 | v_Vel_Ref3 | 0.325 |
| | v_Vel_preced1 | 0.321 |
| | v_Vel | **0.352** |
| Group 2 | v_Vel_preced1 | **0.341** |
| | v_Vel_follow1 | 0.328 |
| | v_Vel_Ref1 | 0.330 |
| Group 3 | v_Vel_preced2 | **0.339** |
| | v_Vel_follow2 | 0.333 |
| | v_Vel_Ref2 | 0.326 |

Table 6: Correlated Groups with 0.85 as threshold and the matching feature importance

The three most important features here are *v_vel* for group 1, *v_Vel_preceed1* for group 2, and *v_Vel_preceed2* for group 3.

Finally, after having removed the features present in the three correlated groups, other than the three most important features above, we have removed a total of 5 features and have 18 features remaining out of 23.

Please refer to Table 22 to see the new feature sets generated thanks to this filtering technique.

### 4.1.7 Results

After having applied a total of 6 different filter techniques upon the original 23 features, we have come up with 10 additional new feature sets (see table below).

| Feature Set | Description |
|---|---|
| A | Original Feature Set |
| B | Information Gain Classification, $k = 5$ |
| C | Information Gain Classification, $k = 10$ |
| D | Information Gain Regression, $k = 5$ |
| E | Information Gain Regression, $k = 10$ |
| F | Correlated Features, $threshold = 0.25$ |
| G | Correlated Features, $threshold = 0.50$ |
| H | Correlated Features, $threshold = 0.75$ |
| I | Correlated Features, $threshold = 0.85$ |
| J | Correlated Features, $threshold = 0.90$ |
| K | Correlated Features, $threshold = 0.95$ |

Table 7: Feature Sets

We will now start measuring their performances to find which one performs the best for our goal.

## 4.2 Filter Performances

After having applied different feature techniques upon the original feature sets, we will now proceed to measure the performance of each of them. To measure the performance of each feature set, metrics will be used to compare them with themselves and allow us to come up with the best feature set for classification, regression, and classification & regression obtained using feature techniques.

### 4.2.1 Second Batch Feature Selection

Before starting to talk about the different performance measures it is important for us to precise that we performed a second batch correlated features feature selection to create more combinations and to try them out. He will now explain the process.

In the next sections of this report, we will present to your the different classification and regression metrics used to compare the models' performances. We have used these very metrics to come up with the best feature set for classification and the best feature set for regression.

Once these two best feature sets have been selected comparing their metrics, they were merged (by adding the two sets together and removing the duplicates) to create a new feature set. From that new feature set was applied, like said earlier, the correlated features feature selection using the same thresholds.

This second batch feature selection created a total of 7 feature sets and will be used to find the best feature sets. To compare the difference obtained with performing this second batch, we will find the best feature sets only using the first 11 feature sets, and then incorporate the 7 new ones.

| Feature Set | Description |
|---|---|
| L | FB BFS for classification + FB BFS for regression |
| M | Feature Set L Correlated Features, $threshold = 0.25$ |
| N | Feature Set L Correlated Features, $threshold = 0.50$ |
| O | Feature Set L Correlated Features, $threshold = 0.75$ |
| P | Feature Set L Correlated Features, $threshold = 0.85$ |
| Q | Feature Set L Correlated Features, $threshold = 0.90$ |
| R | Feature Set L Correlated Features, $threshold = 0.95$ |

Table 8: New feature sets obtained during the second batch feature selection. (*FB*: First batch, *BFS*: Best Feature Set).

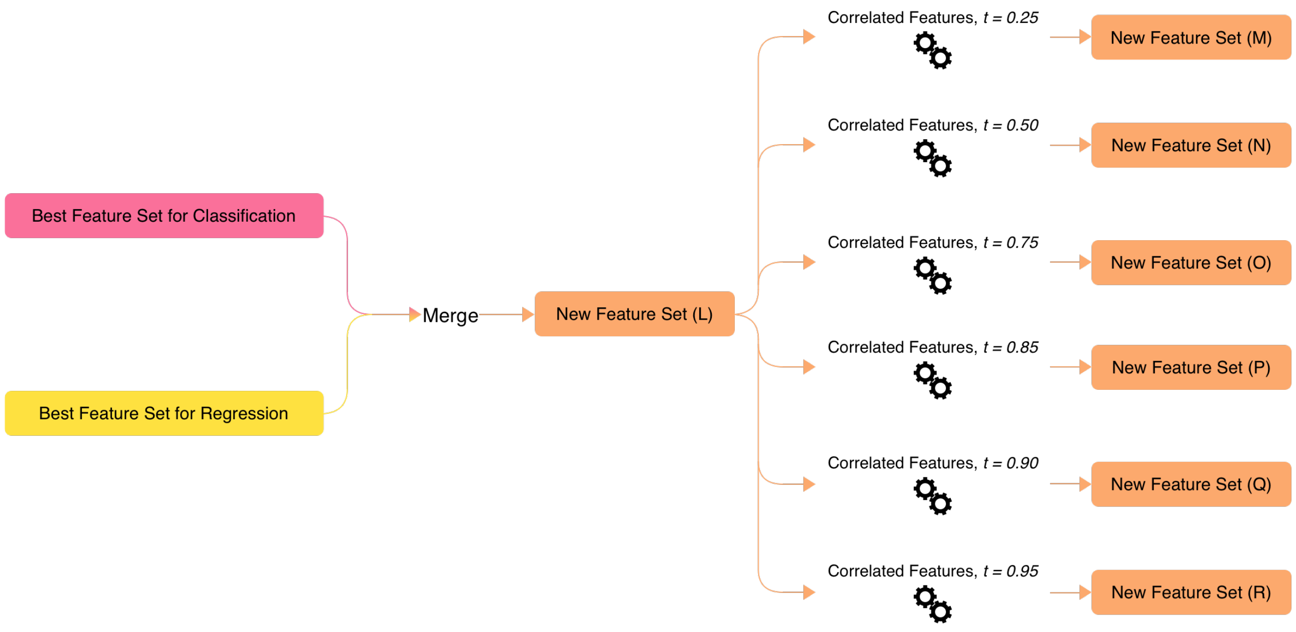
Best Feature Set for Classification
Best Feature Set for Regression
Merge
New Feature Set (L)
Correlated Features, t = 0.25
New Feature Set (M)
Correlated Features, t = 0.50
New Feature Set (N)
Correlated Features, t = 0.75
New Feature Set (O)
Correlated Features, t = 0.85
New Feature Set (P)
Correlated Features, t = 0.90
New Feature Set (Q)
Correlated Features, t = 0.95
New Feature Set (R)

Figure 7: Second Batch Feature Selection diagram

### 4.2.2 Classification

#### Metrics

**Accuracy** Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- $TP = TruePositives$,
- $TN = TrueNegatives$,
- $FP = FalsePositives$,
- $FN = FalseNegatives$.

**Precision** Precision attempts to answer the following question:

*What proportion of identifications were correct?*
Precision is defined as follows:

$$\begin{aligned} Precision &= \frac{TruePositive}{TruePositive + FalsePositive} \\ &= \frac{TruePositive}{TotalPredictedPositive} \end{aligned}$$

**Recall** Recall attempts to answer the following question:

*What proportion of actual positives was identified correctly?*
Mathematically, recall is defined as follows:

$$\begin{aligned} Recall &= \frac{TruePositive}{TruePositive + FalseNegative} \\ &= \frac{TruePositive}{TotalActualPositive} \end{aligned}$$

**Original Set**

**Method** We worked on the original dataset to build the basis of a comparison between the different sets created. As for the classification attempt on the original set, we used a RandomForestClassifier from sklearn.

**Results** We obtained the following results regarding its scoring.

| | |
|---|---|
| Accuracy | 0.9687 |
| Precision | 0.9694 |
| Recall | 0.9687 |

Table 9: Random Forest Classification measure result on original set of features

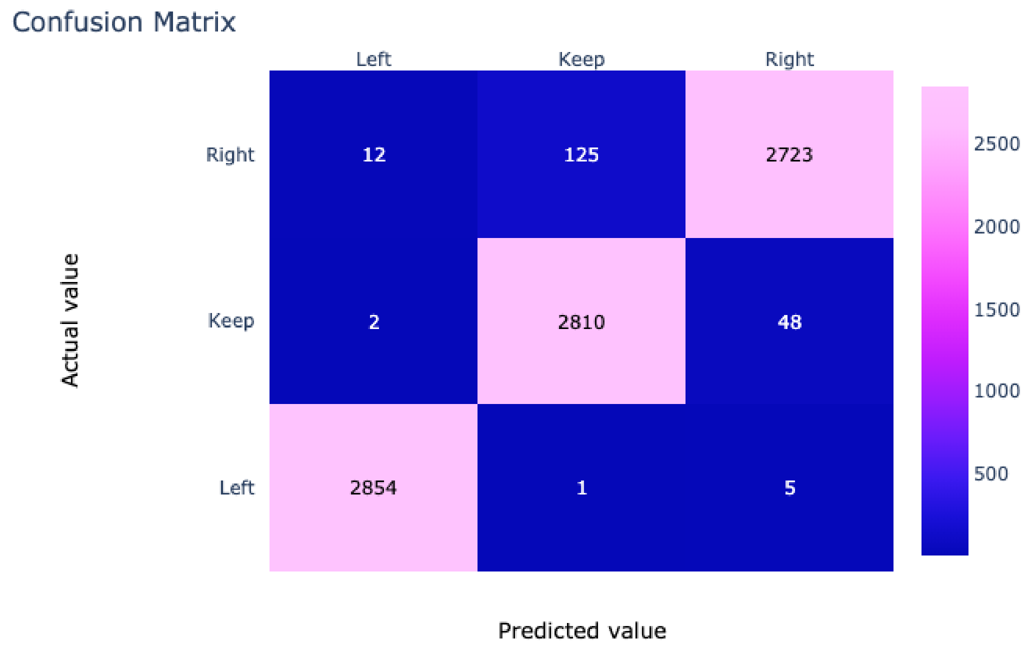

Figure 8: Original Feature Set Classification Confusion Matrix

### Best Set

**Method** After applying different filter methods with different parameters on our original dataset, we had to determine which one was the best subset of features for the classification problem.

Subsets obtained from the first feature selection batch (cf Table 7) :

1. Feature set B, C from Information Gain Classification
2. Feature set D, E from Information Gain Regression

3. Feature set F,G,H,I,J,K from Correlated Features

To compare them we ran a Random Forest classifier on our different subsets and used cross-validation to get the three metrics presented before. After that, we compare every metric of every subset to each other to obtain a positive difference of each metrics and get the best filter subsets for the first batch.

**Results** We compared the different metrics for each subset and have found the best set was the feature set H (cf Table 19) filtered by the correlated features feature selection with a threshold of 0,75. This feature set including 15 features has the following metrics results :

| | |
|---|---|
| Accuracy | 0.976 |
| Precision | 0.976 |
| Recall | 0.976 |

Table 10: Random Forest Classification measure result on the correlated features feature selection 0.75 set

**First Batch** Below, we can visualize the confusion matrix of the feature set H (describe in the results) which the best feature set of the first batch for classification. By analyzing the confusion matrix we can see that they are less False Positives and Negatives than the other methods used in the first batch.

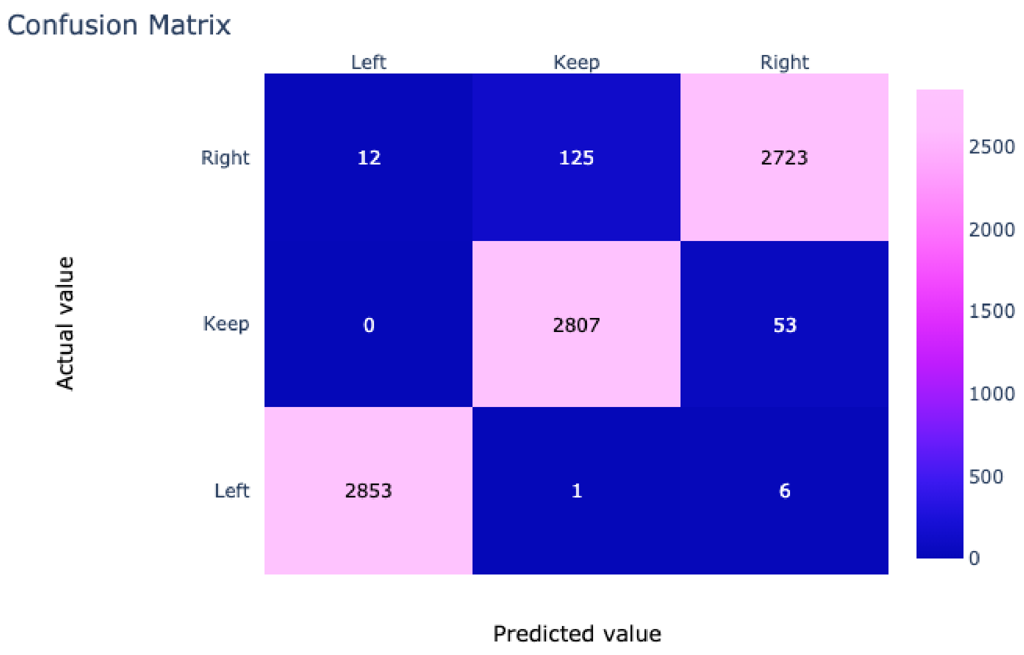

Figure 9: Best Feature Set from first feature selection batch for Classification Confusion Matrix

**Second Batch** Below, we can visualize the confusion matrix of the best feature set from the second batch for classification which has 19 features. We can see, by watching the False positive, that this feature set is as efficient in performance than the feature set of the first batch except some false positive less.

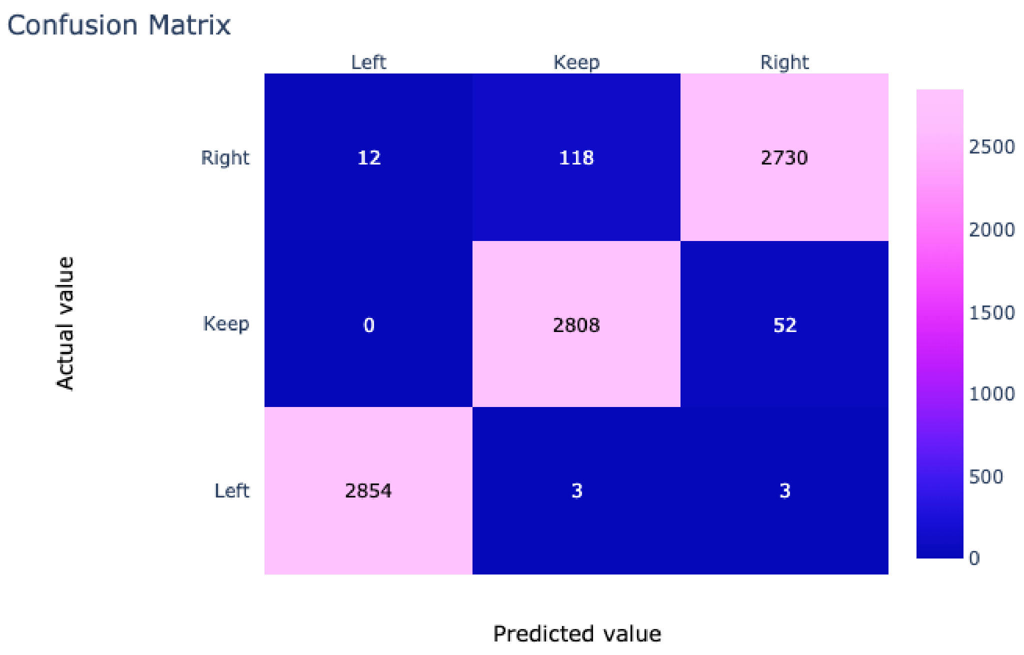

Figure 10: Best Feature Set from second feature selection batch for Classification Confusion Matrix

### 4.2.3 Regression

#### Metrics

**Mean absolute error** Mean absolute error, otherwise called MAE, is the estimation of the error according to pairs of values. We calculate the absolute difference between the predicted value and true value than get the MAE as following:

$$MAE = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

Where $e_i$ is the difference between $x_i$ (predicted) and $y_i$ (true) values.

**R2 Score** R2 Score corresponds to the coefficient of determination. It allows us to quantify the quality of the prediction using regression. The way to calculate the R squared is the following:

$$R^2 = 1 - \frac{SST}{SSR}$$

SST corresponds to the total sum of squares and SSR to the residual sum of squares.

**Max Error** Then we calculate the Maximum Error, otherwise called Maximum Residual Error. It evaluates how much the results deviate from the theoretical value.

$$ME = max(|x - \hat{x}|)$$

Where x is the true and x̂ the predicted value

**Original Set** Thanks to sckitlearn.metrics we could estimate these metrics with regression. We did it on the original feature set to start, to get a base of comparison with the other sets. We can notice the following results :

| | |
|---|---|
| MAE | 0.275 |
| R2 | -0.394 |
| Max Error | 3.110 |

Table 11: Random Forest Regression Results on original set of features

*About R2: that negative result means that our SST is superior to our SSR.

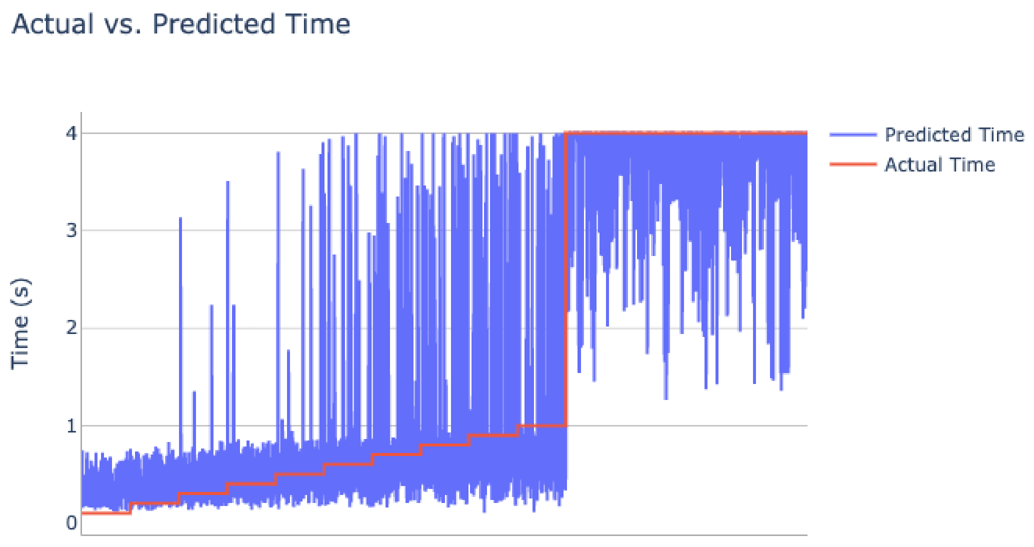

Figure 11: Original Feature Set Regression Target vs. Prediction

**Best Set** To find the best feature set for regression we also used a random forest regressor to know which set was best suited for the regression problem. To know which feature set had the best performances we used the same regression metrics as previously (mean absolute error, R2 Score, and max error).

**First Batch** The best feature set found for the regression problem in the first feature selection batch was the feature set J, which resulted from the correlated features having a threshold of 0.90. This feature set has a total of 19 features remaining, and 4 removed from the original set. The 4 removed features were all features concerning velocity. Here is a graph showing us the Target vs. Predicted Curves:

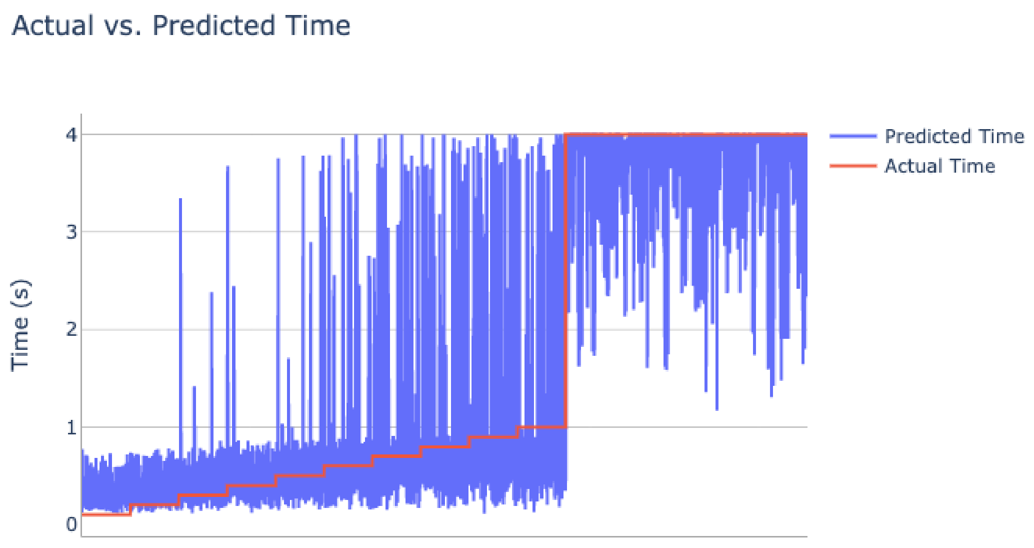

Figure 12: Best Feature Set from first feature selection batch for Regression Target vs. Prediction

Finally, here are it's metrics:

| | |
|---|---|
| Mean Absolute Error | 0.2684 |
| R2 Score | −0.3262 |
| Max Error | 3.0879 |

Table 12: Best feature set for regression from batch 1's metrics

**Second Batch** Now that we also have the second batch's feature sets to compare, we found that the best feature set for the regression problem in the second feature selection batch was the feature set Q, which resulted from the correlated features having a threshold of 0.90 on the feature set L. Here is a graph showing us the Target vs. Predicted Curves:

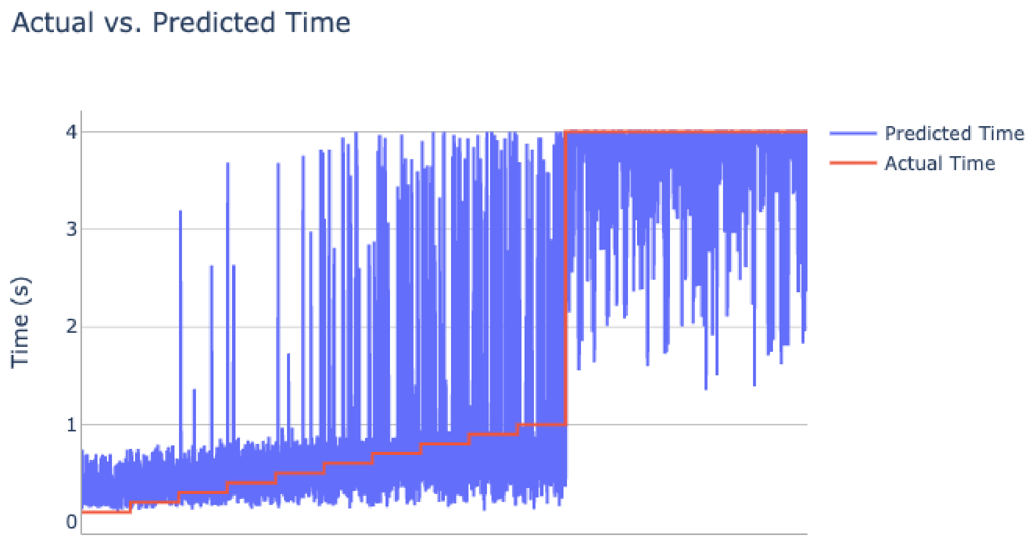

Figure 13: Best Feature Set from second feature selection batch for Regression Target vs. Prediction

Finally, here are it's metrics:

| | |
|---|---|
| Mean Absolute Error | 0.2661 |
| R2 Score | −0.3097 |
| Max Error | 3.1260 |

Table 13: Best feature set for regression from batch 2's metrics

We can see that with the second batch there is a slight improvement with the mean absolute error and the R2 score. However, we observe a slight decrease in performance when it comes to the Max Error.

### 4.2.4 Classification & Regression

Now that we have seen the best feature sets for classification only and regression only, let's find out what the best feature set is for both classification and regression.

To avoid having to create manually a classification and a regression Random Forest model for every feature set, we used a loop to create new models for each set. The same metrics have been used as earlier (classification: accuracy, precision, and recall; regression: mean absolute error, R2 score, max error). We then proceeded to cross-validation on each feature sets for regression and classification containing 10 folds.

After having obtained the results all we need to do is compare them with one another. To compare the results, we compared each feature set's metrics with every other feature set's metrics, one by one. Thus, giving us the difference (positive or negative) for each set.

Having to find the best feature set for classification and regression at the same time, to compare the results was somewhat subtle since we had to find the feature set giving us an equally good performance in classification as in regression. Since we did not seek to find the best feature set for classification & regression with only the first batch, the result will be based on the results from the second batch (from the feature set A to feature set R).

The best feature set for classification and regression is the feature set which resulted from the correlated features with a threshold of 0.85 (Feature Set I). Here are the confusion matrix and the Target vs. Predicted Curves obtained from this feature set:

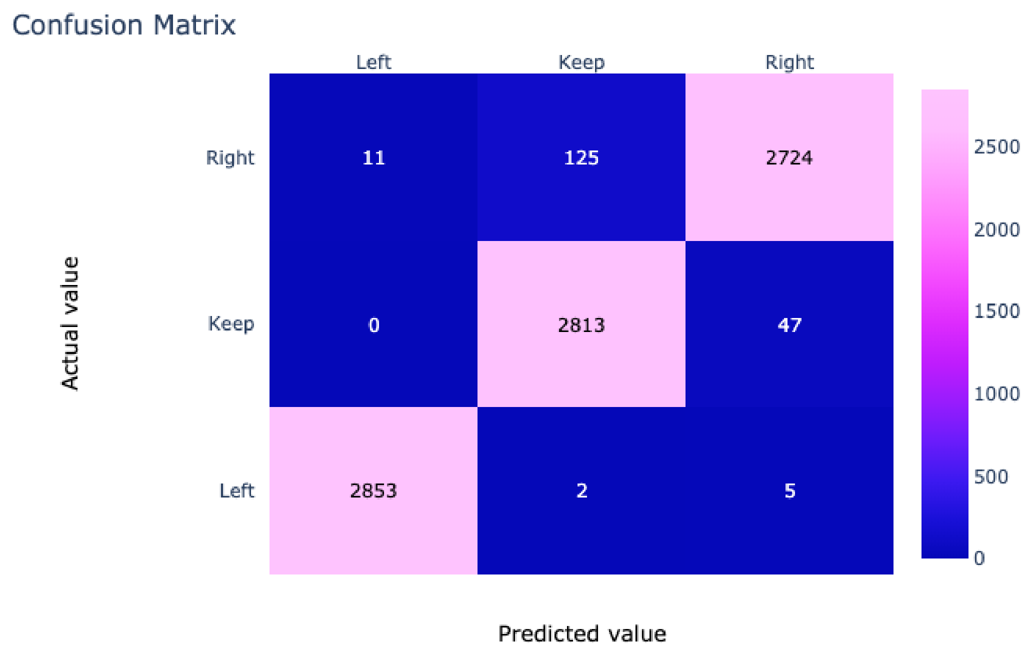

Figure 14: Best Feature Set for Classification & Regression Confusion Matrix

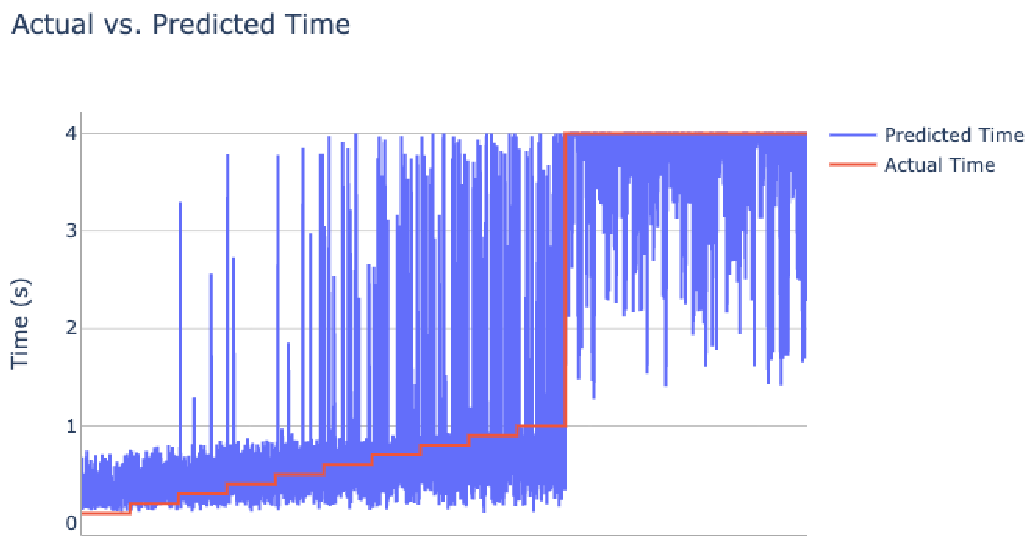

Figure 15: Best Feature Set for Classification & Regression Target vs. Prediction

And finally, here are it's metrics:

| | | |
|---|---|---|
| Classification | Accuracy | 0.9757 |
| | Precision | 0.9762 |
| | Recall | 0.9757 |
| Regression | Mean Absolute Error | 0.2711 |
| | R2 Score | −0.3221 |
| | Max Error | 3.1480 |

Table 14: Random Forest Classification and Regression Results on Correlated Features Feature Selection Set

## 4.3 Results

After having used six different filtering techniques we came up with 11 different feature sets. Adding these with the feature sets issued after a second batch feature selection (Correlated feature selection on the addition of the best feature set for classification and the best one for regression), we had a total of 18 feature sets.

We performed metric comparisons to come up with the best feature sets for classification, regression, and classification & regression.

Here is a table summing which feature set was the best:

| | |
|---|---|
| Best for Classification | Feature Set L |
| Best for Regression | Feature Set Q |
| Best for Classification & Regression | Feature Set I |

Table 15: Summary of the best feature sets

Knowing that the bigger problem is a classification and regression problem, we are going to continue the rest of our study using the best feature set for classification & regression (feature set I)

## 5 Wrapper Techniques

After having selected some features with the Filtered methods it was interesting to use Wrapper Methods because some features independently useless could be useful include in a subset of features.

Wrapper Methods work like greedy methods. For each feature it will use Machine Learning algorithms to create the best subset of features with the following criteria: Which one has the best performance?

Wrapper Methods allow to detect interaction between variables then to find the optimal final subset. This way this method offers a better predictive accuracy.

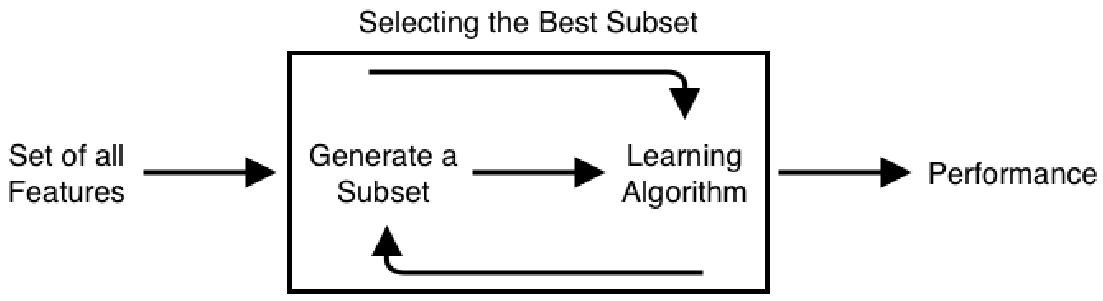

Figure 16: Wrapper Techniques Workflow

## 5.1 Used Metrics

To measure the performance of the wrapper techniques that we used we introduced two new metrics for classification and regression while removing one for regression.

### 5.1.1 Classification

As said earlier, we have added two new metrics for the classification which are the *F1 Score*, and the *Area Under the Receiver Operating Characteristic curve* (*ROC AUC*)

**F1 Score** The F1 Score is another well-known metric for classification, it is a function of Precision and Recall. Its formula is as follow:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

This metric is useful when you want to seek a balance between Precision and Recall. It is different from the accuracy, which can be largely contributed by a large number of True Negatives which in some cases has less value than False Negatives and False Positives. Indeed, the F1 Score can be used if we seek a balance between Precision and Recall and when there is an uneven class distribution.

**ROC AUC** The ROC AUC is a very effective metric to compare the performance of two classification machine learning model. ROC stands for Receiver Operating Curve and is the curve between the False Positive Rate and the True Positive Rate by different threshold and so different confusion matrix. AUC stands for Area Under Curve and more the AUC increase better is the classification model. As the AUC will only increase if and only if ROC has points on the top left as it conveys that for some threshold the model has high true positive rate and low false positve rate.

So we applied this metric by computing Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores to each class against the rest.

### 5.1.2 Regression

For the regression, we have also added two new metrics, as well as removing one. The two added metrics are the *Mean Squared Error*, and the *Median Absolute Error*. Concerning the removed metric, it is the *Max Error*.

**Mean Squared Error** The Mean Squared Error (MSE) is the average of the squares of the errors. It allows us to measure the quality of an estimator.

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where n is the size of the sample, y the true values, and ŷ the predicted values.

**Median Absolute Error** The Median Absolute Error otherwise called Median Absolute Deviation (MAD) measures the variability of a univariate sample of quantitative data.

$$MAD = \frac{1}{n} \times \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

## 5.2 Sequential Feature Selection

Sequential feature selection consists of starting from a set of features and adding or removing them one by one until you get the best subset according to the performance. To evaluate this performance we used metrics specific to classification or regression.

In our implementation of the Sequential Feature Selection, the number of features to be found was inputted as a range. Meaning that the method should return a list of features within the feature range. To facilitate the tests and avoid having too many biased range creation, we generated 10 random ranges, with the minimum and maximum having a difference of at least 5. This allows the ranges to not be too small.

### 5.2.1 Forward

The forward feature selection consists of starting with an empty feature set, then test the features one by one and adding them if the performance is higher with this feature.

**Classification** After having run the Sequential Forward Feature Selection for classification we have come up with a feature set containing 7 features. This feature set will be named *Feature Set S*

Here is the confusion matrix of these features when using a Random Forest Classifier with 10 folds cross-validation.

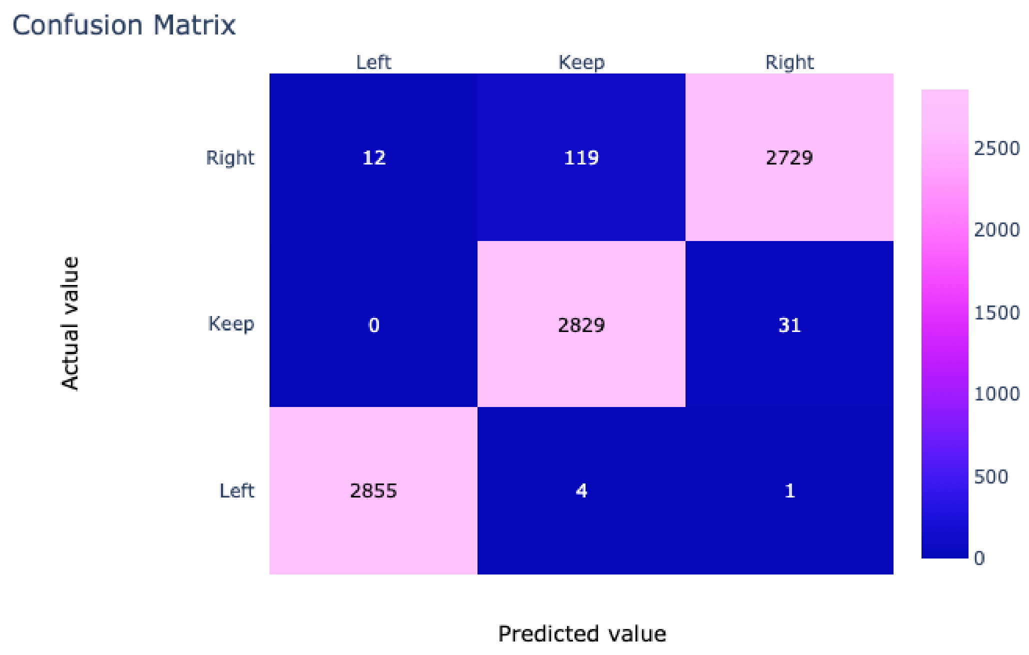

Figure 17: Best Feature Set for Classification using Sequential Forward Feature Selection Confusion Matrix

Now that we the confusion matrix let's have a look at the metrics and more information concerning the results.

| | |
|---|---|
| Range | (6,15) |
| Accuracy | 0.9808 |
| Precision | 0.9813 |
| Recall | 0.9808 |
| F1 Score | 0.9808 |
| ROC AUC | 0.9947 |

Table 16: Sequential Forward Feature Selection Classification Best Feature Set's Metrics

**Regression** Then we have applied the same strategy for regression and have got the following results. The feature set that we got will be named *Feature Set T*, it contains 7 features.

| | |
|---|---|
| Range | (2,8) |
| Mean Absolute Error | 0.2760 |
| Mean Squared Error | 0.2716 |
| Median Absolute Error | 0.1540 |
| R2 Score | −0.2666 |

Table 17: Sequential Forward Feature Selection Regression Best Feature Set's Metrics

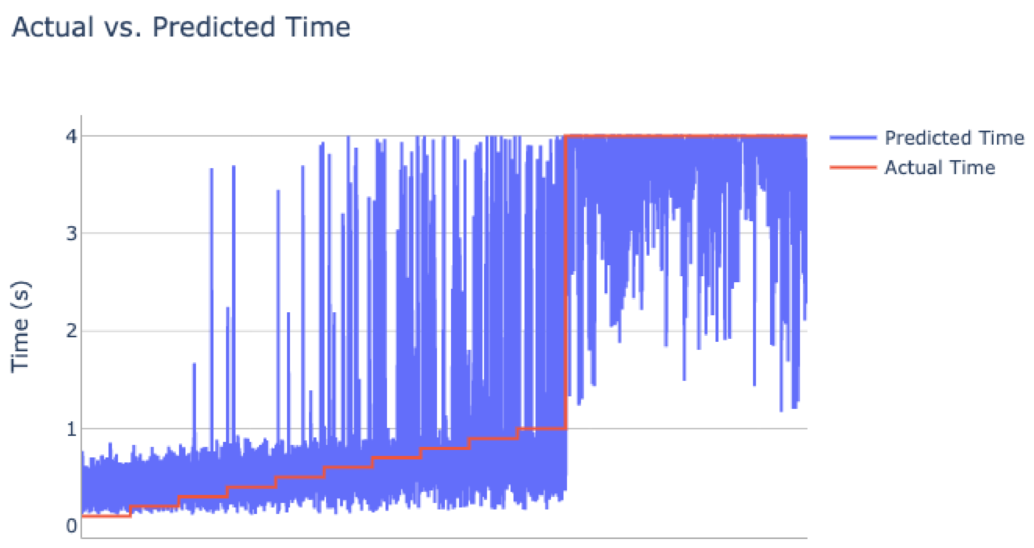

Figure 18: Best Feature Set for Regression using Sequential Forward Feature Selection Target vs. Prediction

### 5.2.2 Backward

The Backward Feature Selection otherwise called Backward Feature Elimination consists of starting from a full set and removing features one by one until getting the best performance. As well as the Forward Feature Selection, we used evaluation metrics to measure the performance of a subset.

**Classification** For the Sequential Backward Feature Selection, the best feature set has 10 features. This feature set will be named *Feature Set U*. Here is the confusion matrix of these features when using a Random Forest Classifier with 10 folds.

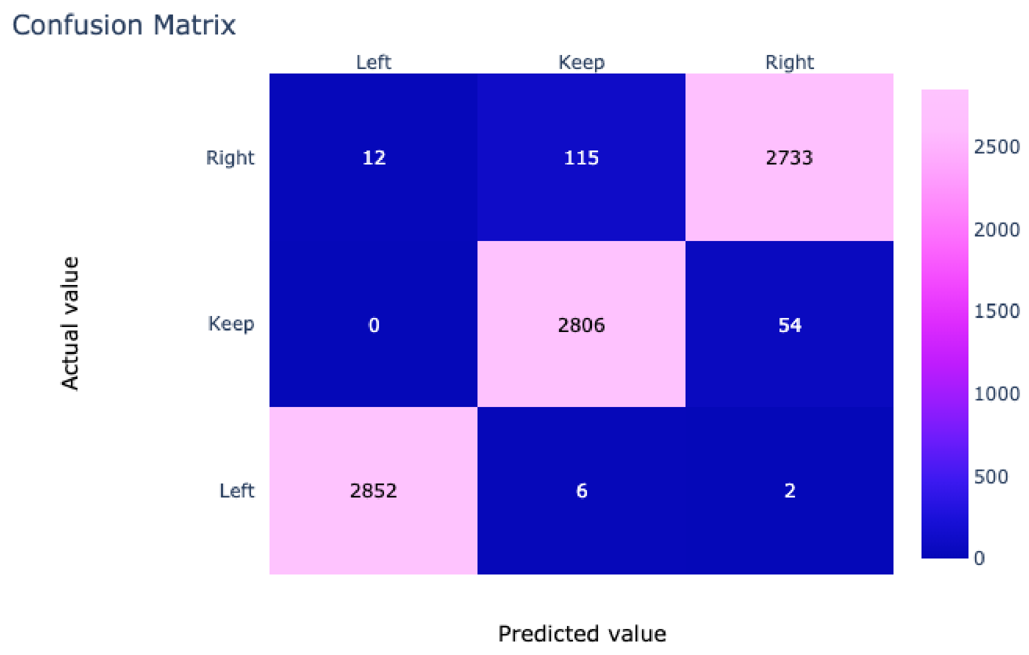

Figure 19: Best Feature Set for Classification using Sequential Backward Feature Selection Confusion Matrix

Now that we the confusion matrix let's have a look at the metrics and more information concerning the results.

| | |
|---|---|
| Range | (6,12) |
| Accuracy | 0.9805 |
| Precision | 0.9808 |
| Recall | 0.9805 |
| F1 Score | 0.9805 |
| ROC AUC | 0.9959 |

Table 18: Sequential Backward Feature Selection Classification Best Feature Set's Metrics

**Regression** As well as with classification, we tested different ranges and got the best results for (7,16). Here there are the metrics of the *Feature Set V* which contains 12 features.

| | |
|---|---|
| Range | (7,16) |
| Mean Absolute Error | 0.2702 |
| Mean Squared Error | 0.2728 |
| Median Absolute Error | 0.1450 |
| R2 Score | −0.2470 |

Table 19: Sequential Backward Feature Selection Regression Best Feature Set's Metrics

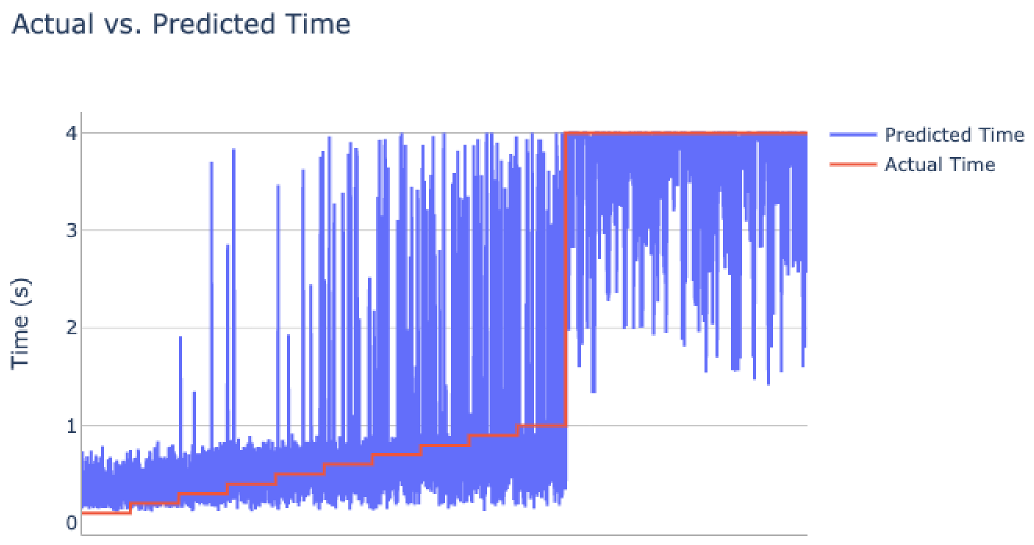

Figure 20: Best Feature Set for Regression using Sequential Backward Feature Selection Target vs. Prediction

## 5.3 Exhaustive Feature Selection

The Exhaustive Feature Selection (EFS) consists of trying all the possible features combinations. To apply Exhaustive Feature Selection you need to precise a minimum and a maximum number of features to avoid taking so much time, (EFS is heavier than both precedent algorithms). The aim stays the same: getting the best subset of features according to the performance, so to the evaluation of metrics.

### 5.3.1 Classification

We applied the EFS to our classification problem by computing the selector with a Random Forest Classifier for different min-max values. After that we were able to get the best feature combination evaluate by a metric. To get other metrics we ran a cross-validation with a Random Forest Classifier on the subset obtained and printed all the metrics needed.

We computed the selector for a min-max of 7-18, after having trying others combinations under 1 and 7. The execution time was extremely long (8 hours) because of the number of combination to test. For example for the 7-18 range, 230964 different combinations of features were tried. Finally this method enable us to get a new subset of 8 features the set was reduces of 10 features and had the best accuracy of the 230964 combinations.

We get the other metrics with a cross validation, which are printed in the table below :

| | |
|---|---|
| Accuracy | 0.98 |
| Precision | 0.979 |
| Recall | 0.978 |
| F1 | 0.978 |
| Roc Auc | 0.993 |

Table 20: Metrics result on the best combination of features from the classification exhaustive feature selection

After the metrics, we can also see the correlation matrix of this set of features and especially the low rate of False Positive.

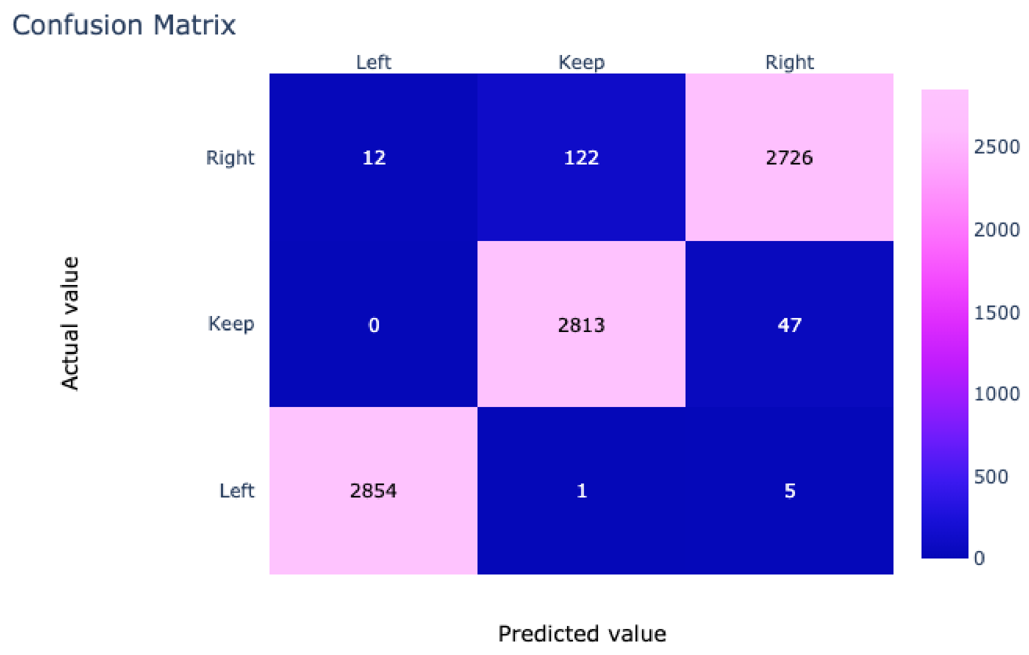

Figure 21: Best Feature Set for Classification using Exhaustive Feature Selection Confusion Matrix

### 5.3.2 Regression

Following the same logic used for the exhaustive feature selection for classification exept this time we will be using a Random Forest Regressor when computing the selector. As for calculating the other metric we used a Random Forest Regressor on the subset obtained in the first phase.

Due to computation power issues we constructed the selector on a min-max range of 3-5. We get the other metric with a cross validation and here are the results:

| | |
|---|---|
| Mean absolute error | 0.2705 |
| Mean Squared Error | 0.2841 |
| Median Absolute Error | 0.1474 |
| R2 Score | -0.3986 |

Table 21: Metrics result on the best combination of features from the classification exhaustive feature selection

### 5.4 Results

Feature sets avec moins de features pour la dimension mais avec un resultat convenable

Best classif Best regression

## 6 Improvements

In every project, there is always room for improvement. Throughout the project, we have noticed improvements that could be made.

### 6.1 Embedded Techniques

We have seen two classes of feature selection, filter, and wrapper methods. However, there is another class of feature selections which widely known: the embedded methods.

What differentiates the embedded methods compared to the others is that they complete the feature selection process within the construction of the machine learning algorithm itself. In other words, they perform feature selection during the model training, which is why we call them embedded methods.

But what is so different between all three of them? The wrapper methods are used to measure the "usefulness" of features based on the classifier/regressor performance. However, the filter methods pick up the intrinsic properties of the features (i.e., the "relevance" of the features) measured via univariate statistics instead of cross-validation performance. Therefore, wrapper methods are essentially solving the "real" problem (optimizing the classifier/regressor performance), but they are at the same time computationally more expensive than the filter methods due to the repeated learning steps and cross-validation.

Finally, the embedded methods, are quite similar to wrapper methods since they are also used to optimize the objective function or the performance of a learning algorithm or model. The notable difference to wrapper methods is that an intrinsic model building metric is used during learning. Here is the embedded methods workflow.

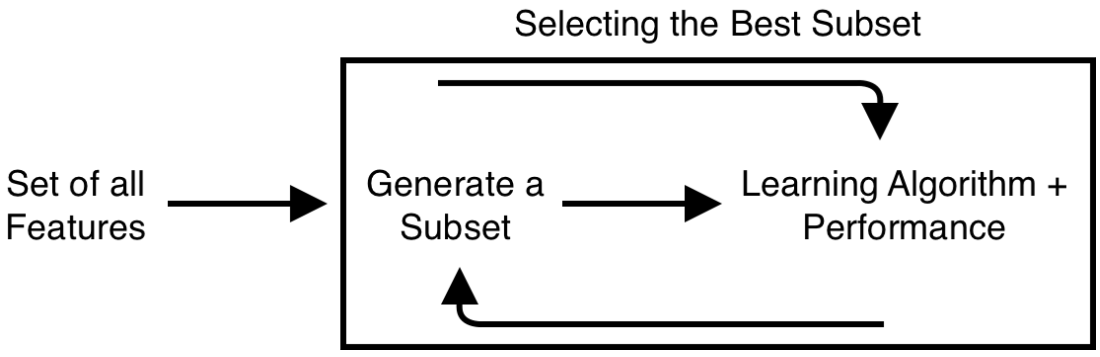

Figure 22: Embedded Techniques Workflow

With having only feature selected using the two first methods, it would be interesting to see if the performances can be improved by finding the optimal feature set using the embedded methods.

## 6.2 Computing Power

It is well known that machine learning algorithms can be very demanding in terms of computing power. During the implementation of some methods we have been limited by the power of our computers. Some members of the team have different machines and it can possibly make a difference in term of calculation. But the gap was mostly in term of execution time, in fact our machines were really limited and overloaded to do some tasks. Especially for the exhaustive wrapper which needed all the power of our computers and took around 8 to 9 hours to run 100% of the processor's power clocked at 4.75GHz.

Furthermore, the difference between the different metrics calculated to compare performances between feature sets were so little that from one execution to another the best feature set always changed. The best feature set during one execution could become one of the worst in the next one.

To try and overcome this problem we could have calculated for each feature set their metrics several times (around 10 for example) and compare the mean of their metrics. Unfortunately, having experienced some difficulties for calculating only one the metrics, this task was impossible for us.

## 6.3 Deep Learning

Using Deep Learning techniques to select our features would have been a very efficient alternative to the other methods.

In fact, techniques like applying an auto-encoder to the features (AEFS) would have given us another vision of the problem and the result. This method is quite different of the others, the ANN compress and encode the data, learn them and reconstruct them. By reconstructing them, it will reconstruct the feature space and ignore the ”noise” which are the useless features. Other efficient methods like the Teacher-Student architecture could be interesting to try in the future.

### 6.4 Hyperparameter Optimization

The sklearn library give us access to RandomizeSearchCV(). This class' purpose is to optimize the RandomForest parameters by setting the number of trees in the forest, the maximum number of leaves or even the maximum value of deepness of trees. The random search is an estimator which trains and evaluate different version of the model and selects the best one. Another way to optimize our model would be to use a grid search which studies the same space of possibility but is more precise because RandomSearch is subject to noise . But one advantage that can be taken from the RandomSearch is its speed of execution.

## 7 Feedback

### 7.1 CHABNI Chabane

I come out a lot of points from this project, firstly I appreciated the fact that this work was complete for us, we have done the research, found methods, understood and applied them to the problem. We get around all the main feature selection techniques and I have learn a lot from those methods. Secondly, the context of the project enable me to use new team working method via Teams.

### 7.2 CISSÉ Babacar

I am grateful to have the chance to participate in that project which helped me keep growing. On the technical aspect learning how to reduce computation time by removing less important features will for sure help me in the future. But this project was also the occasion to deepen the knowledge shared with me by my teachers. On the personal side working with this team was an amazing experience. the professionalism, the fun, and all the other moment shared together are valuable for me and I would be more than happy to work with them again in the future.

### 7.3 NOREL Arnaud

It was a really rewarding experience. Feature Selection was for me an unknown topic. This subject was very interesting. Furthermore I really appreciated the fact to handle data with the aim to develop smart vehicles. This ending studies project was great because it brought me the feeling of participation to a future world with objects more connected, particularly to a future world where less people would die in the road.

On a technical point of view I really appreciated the teamwork during this project. We dispatched the tasks every weeks and talked really often about the problems we met in our code. Working on collaborate IDE was very efficient to share our bugs or results.

### 7.4 RIGAUX David

I have learned a lot through this project. In my opinion, feature selection is a step in every machine learning or deep learning project, which is more often than not overlooked. Besides, I have never had a class about feature selection, only learning from blog articles, forums, and so on.

To learn about the different techniques and the order in which it is suggested to apply them is going to be very helpful for future projects.

The small and brief glance this project allowed me to have upon the research world is very interesting and has changed my opinion on the matter.

Working with my teammates was very fun, and we have learned a lot while working as a group.

### 7.5 SEBBAG Ashley

As with every project we had to carry out within the EISTI, this PFE has been a source of learning for me. It also allowed us to put into practice our knowledge acquired during this year in AI option. This collaboration with Altran opened a door to the world of research and brought us a new perspective on teamwork. In addition, the coaching we received from Altran on the progress and organization of the project was very beneficial as it allowed us to better focus our efforts and our work on the subject. I am very happy to have carried out my PFE on feature selection because it is a subject at the heart of machine learning and artificial intelligence.

## 8 Conclusion

### 8.1 Chabane ORIGINAL

The aim of this work was to reduce as much as possible the number of features used by our algorithm to assess whether or not an autonomous vehicle changes lane. Consequently, through this project, we were able to browse much of the feature selection techniques. Starting by the filter method going to the wrapper method we were able to get results whether for the classification or for the regression problem. In consequence, as things progress, we have optimized the performance of our feature sets by handling different metrics and comparing each of them. By computing those different techniques we obtained a best feature set for classification and another for regression with a reduce number of features. Moreover, we found a compromise between a the classification and the regression problem and got a optimized feature set which strives to answer the two kinds of problem. We were able to adapt to the computing power, especially for the wrapper method which asked a lot of time. Improvements are possible, depending on the computing power and other methods that deserve to be tried. In a nutshell, we were able to get some conclusive result about the feature importance and the feature space reduction through a well organized project dealing with an innovative subject.

## 8.2 Improved

With the popularity rise of autonomous cars (Tesla, BMW, Ford, Mercedes-Benz, ...) and intelligent cars, the global researcher and engineer community is constantly trying to improve the performances of their systems. The act of a driver changing lane is filled with dangers including the surrounding cars (speed and position). The amount of data treated by the human brain is too voluminous for AI algorithms to process and to predict the drivers' actions. The aim of our work was to reduce as much as possible the numbers of features used by machine learning algorithms to assess whether or not a vehicle changes lane. To attain our goal, we applied several feature selection techniques and compared the performances of the outputted feature sets. The problem we had was a classification and regression problem. Indeed, the classification task was to predict whether the current vehicle was going to change lane and go on the left one or the right one, or going to keep lane. Regarding the regression task, it had to predict the time until the lane change (or lane keeping) was finalised.

The data we had at our disposition was a dataset based on calculations made from the NGSIM dataset made by the U.S Department of Transportation. The number of features was of 23, and described the velocity, lateral distance, longitudinal distance, and ITTC of the current vehicle and its surrounding vehicles.

To achieve our goal, our project was divided in five assignments which represented the different steps to complete our study.

We worked on two of the main types of feature selection: filter techniques and wrapper techniques. Using six filter techniques (constant, quasi-constant, duplicates, anova, information gain). During our filter techniques study, we decided to perform two batches. After the filter techniques we had a total of 18 feature sets. Regarding the wrapper techniques, we used 2 techniques (sequential feature selection and exhaustive feature selection). The sequential feature selection having a forward and a backward mode, the techniques outputted a total of 6 new feature sets (3 classification optimized and 3 regression optimized).

After having compared the 24 feature sets metrics', we were able to find the best feature sets for classification and regression.

- Best feature set for classification: S - Best feature set for regression: X

These two feature sets are only optimised for their respective tasks. We have tried to find a balanced feature set for classification and regression at the same time during our filter techniques study. This best balanced feature set for the classification and regression task is the feature set I. It is however important to note that without a joint learning model, we cannot find the optimal feature set for these two tasks at the same time.

While working and studying the different results obtained, we realised that there was a certain number of improvements which could be done to improve the correctness and the optimal feature set. These improvements include having more computing power, using the embedded feature selection techniques, applying deep learning, performing hyperparameter optimization upon the Random Forest models used, and having access to a joint learning model.

In a nutshell, we were able to get some conclusive result about the feature importance and the feature space reduction through a well organized project dealing with an innovative subject.

## References

[1] Ruina Dang, Fang Zhang, Jianqiang Wang, Shichun Yi, and Keqiang Li. Analysis of chinese driver's lane change characteristic based on real vehicle tests in highway. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 1917–1922. IEEE, 2013.

[2] Ürün Dogan, Johann Edelbrunner, and Ioannis Iossifidis. Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 1837–1843. IEEE, 2011.

[3] Anup Doshi and Mohan Manubhai Trivedi. On the roles of eye gaze and head dynamics in predicting driver's intent to change lanes. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):453–462, 2009.

[4] Haijing Hou, Lisheng Jin, Qingning Niu, Yuqin Sun, and Meng Lu. Driver intention recognition method using continuous hidden markov model. *International Journal of Computational Intelligence Systems*, 4(3):386–393, 2011.

[5] Nobuyuki Kuge, Tomohiro Yamamura, Osamu Shimoyama, and Andrew Liu. A driver behavior recognition method based on a driver model framework. *SAE transactions*, pages 469–476, 2000.

[6] Puneet Kumar, Mathias Perrollaz, Stéphanie Lefevre, and Christian Laugier. Learning-based approach for online lane change intention prediction. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 797–802. IEEE, 2013.

[7] Guofa Li, Shengbo Eben Li, Yuan Liao, Wenjun Wang, Bo Cheng, and Fang Chen. Lane change maneuver recognition via vehicle state and driver operation signals—results from naturalistic driving data. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 865–870. IEEE, 2015.

[8] Hiren M Mandalia and Mandalia Dario D Salvucci. Using support vector machines for lane-change detection. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 49, pages 1965–1969. SAGE Publications Sage CA: Los Angeles, CA, 2005.

[9] Alex Pentland and Andrew Liu. Modeling and prediction of human behavior. *Neural computation*, 11(1):229–242, 1999.

[10] Amardeep Sathyanarayana, Pinar Boyraz, and John HL Hansen. Driver behavior analysis and route recognition by hidden markov models. In *2008 IEEE International Conference on Vehicular Electronics and Safety*, pages 276–281. IEEE, 2008.

[11] Julian Schlechtriemen, Andreas Wedel, Joerg Hillenbrand, Gabi Breuel, and Klaus-Dieter Kuhnert. A lane change detection approach using feature ranking with maximized predictive power. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 108–114. IEEE, 2014.

[12] Hanwool Woo, Yonghoon Ji, Hitoshi Kono, Yusuke Tamura, Yasuhide Kuroda, Takashi Sugano, Yasunori Yamamoto, Atsushi Yamashita, and Hajime Asama. Lane-change detection based on vehicle-trajectory prediction. *IEEE Robotics and Automation Letters*, 2(2):1109–1116, 2017.

# Appendices

| Feature Set | Description |
|---|---|
| A | Original Feature Set |
| B | Information Gain Classification, $k = 5$ |
| C | Information Gain Classification, $k = 10$ |
| D | Information Gain Regression, $k = 5$ |
| E | Information Gain Regression, $k = 10$ |
| F | Correlated Features, $threshold = 0.25$ |
| G | Correlated Features, $threshold = 0.50$ |
| H | Correlated Features, $threshold = 0.75$ |
| I | Correlated Features, $threshold = 0.85$ |
| J | Correlated Features, $threshold = 0.90$ |
| K | Correlated Features, $threshold = 0.95$ |
| L | FB BFS for classification + FB BFS for regression |
| M | Feature Set L Correlated Features, $threshold = 0.25$ |
| N | Feature Set L Correlated Features, $threshold = 0.50$ |
| O | Feature Set L Correlated Features, $threshold = 0.75$ |
| P | Feature Set L Correlated Features, $threshold = 0.85$ |
| Q | Feature Set L Correlated Features, $threshold = 0.90$ |
| R | Feature Set L Correlated Features, $threshold = 0.95$ |
| S | Sequential Forward Feature Selection for Classification |
| T | Sequential Forward Feature Selection for Regression |
| U | Sequential Backward Feature Selection for Classification |
| V | Sequential Backward Feature Selection for Regression |
| W | Exhaustive Feature Selection for Classification |
| X | Exhaustive Feature Selection for Regression |

Table 22: Complete Feature Sets

| Feature Set | Features |
|---|---|
| A | Original Feature Set |
| B | Information Gain Classification, $k = 5$ |
| C | Information Gain Classification, $k = 10$ |
| D | Information Gain Regression, $k = 5$ |
| E | Information Gain Regression, $k = 10$ |
| F | Correlated Features, $threshold = 0.25$ |
| G | Correlated Features, $threshold = 0.50$ |
| H | Correlated Features, $threshold = 0.75$ |
| I | Correlated Features, $threshold = 0.85$ |
| J | Correlated Features, $threshold = 0.90$ |
| K | Correlated Features, $threshold = 0.95$ |
| L | FB BFS for classification + FB BFS for regression |
| M | Feature Set L Correlated Features, $threshold = 0.25$ |
| N | Feature Set L Correlated Features, $threshold = 0.50$ |
| O | Feature Set L Correlated Features, $threshold = 0.75$ |
| P | Feature Set L Correlated Features, $threshold = 0.85$ |
| Q | Feature Set L Correlated Features, $threshold = 0.90$ |
| R | Feature Set L Correlated Features, $threshold = 0.95$ |
| S | Sequential Forward Feature Selection for Classification |
| T | Sequential Forward Feature Selection for Regression |
| U | Sequential Backward Feature Selection for Classification |
| V | Sequential Backward Feature Selection for Regression |
| W | Exhaustive Feature Selection for Classification |
| X | Exhaustive Feature Selection for Regression |

Table 23: Complete Feature Sets' Features