DATA UPSCALING PROGRAM
REPORT STEP 3 - MODELLING

PROJECT MENTOR		
Antoine TARDIVON	Data Scientist	DataScientest
COHORT		
Clement ARNAUD	Process Engineer	CFT TEN - PARIS
Diego GOMEZ-OCHOA	Process Engineer	REFINING TEN - PARIS
Presheet DESHPANDE	Technical Safety & Risk Engineer	GENESIS - LONDON
Reginaldo MARINHO	Process Engineer	CFT TEN - PARIS
Simran MASOOD	Process Engineer	CFT TEN - PARIS
NAME	***POSITION***	***DEPARTMENT – CENTER***

Contents

1. INTRODUCTION

1.1. Context

Global transportation sector is a major contributor to greenhouse gas emissions, with passenger cars and vans responsible for around 10% of global energy-related CO_2 emissions in 2022 according to International Energy Agency (IEA). This substantial emission rate significantly affects air quality and contributes to climate change. Therefore, identifying the vehicles emitting the most CO_2 and other pollutants is crucial for devising effective strategies to mitigate environmental impact. As automotive technology evolves, understanding the role of technical characteristics with respect to emissions is vital for promoting the development and adoption of cleaner and more efficient vehicles ultimately contributing to the realization of the Net Zero Emissions goals by 2050.

This project explores two datasets (given below) encompassing a wide array of technical specifications of vehicles, alongside their fuel consumption, CO_2 emissions, and pollutant emissions, marketed both in France and Europe. Through the application of Data Science and Machine Learning techniques, our objective is to explore the relationship between vehicle specifications and emissions. By doing so, we aim to provide valuable insights that can inform decision-making processes in environmental policy and drive advancements in automotive industry practices towards sustainable transportation solutions.

The following datasets are provided for reference:

- data.gouv.fr
- European Environment Agency

This project employs a combination of data analysis, statistical modelling, and machine learning techniques to extract actionable insights from the dataset. Exploratory data analysis (EDA) will uncover patterns and relationships within the data, providing a foundational understanding of the variables at play. Feature engineering will involve transforming or selecting relevant variables to enhance model performance. Lastly, statistical modelling techniques, such as linear regression, will help quantify the impact of technical characteristics of vehicles on CO_2 emissions. Additionally, machine learning algorithms, such as decision trees or random forests, or ensemble learning algorithms such as Bagging and Boosting may be utilized for better predictive performance of the model.

1.2. Objectives

The main objective is estimating the CO2 emissions of vehicles based on their characteristics. This can be done by approaching the problem in two ways:

- **Regression Problem**: Estimating the values of the CO2 emissions via Linear Regression algorithms to find a value as precise as possible.
- **Classification Problem**: Grouping the CO2 emissions by ranges and trying to predict, based on the vehicle characteristics, what group it belongs to.

This report is then divided in tow main sections corresponding to the ways to approach the problem as described above.

1.3. Initial Data Set

The data set used in this report is the one obtained in the Step 2 of this project (Pre-processing and Feature Engineering). Refer to the corresponding report for more details.

The initial data set information is shown here-under:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103248 entries, 0 to 103247
Data columns (total 14 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   lib_mrq_utac     103248 non-null  object 
 1   cod_cbr          103248 non-null  object 
 2   hybride          103248 non-null  object 
 3   puiss_max        103248 non-null  float64
 4   conso_mixte      103248 non-null  float64
 5   co2              103248 non-null  float64
 6   co_typ_1         103248 non-null  float64
 7   hc               103248 non-null  float64
 8   nox              103248 non-null  float64
 9   hcnox            103248 non-null  float64
 10  masse_ordma_max  103248 non-null  float64
 11  year             103248 non-null  int64  
 12  type_of_gearbox  103248 non-null  object 
 13  nbr_reports      103248 non-null  object 
dtypes: float64(8), int64(1), object(5)
memory usage: 11.0+ MB
```

2. REGRESSION

Regression is a supervised learning technique that is used to analyse the relationship between a target variable (X) and one or more explanatory variables (y). It models the relationship between explanatory and target variable enabling us to predict or estimate the target variable. In a regression problem, the target variable takes continuous values. These values are numerical: price of a house, quantity of oxygen in the air of a city, etc. The target variable can therefore take an infinity of values.

There are many differed types of regression algorithms, including:

- Linear Regression: it is the most widely used method that assumes a linear relationship between the target variable (X) and the explanatory variable/s (Y). It is used when the data is normally distributed.
- Regularized Regression: This is a more complex method that is used to prevent overfitting in machine learning models. Regularized regression is used when the data is noisy or there is a high number of variables or when the explanatory variables are highly correlated. It includes techniques such as Lasso, Ridge, and Elastic Net regression, which add a penalty term to the regression equation to prevent overfitting.

2.1. Evaluation of Regression Models

The evaluation of the regression model can be done using several metrics:

2.1.1. Mean Squared Error (MSE)

This function consists in measuring the average squared difference between the predicted and actual values.

$$MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

This value could be very high and therefore difficult to interpret which is why, there is another metric called Mean Absolute Error (MAE) which measures the average absolute difference between the predicted and actual values of the target variable.

$$MAE = \frac{1}{n}\sum|y_i - x_i|$$

2.1.2. R-squared (R^2) Value

R-squared is a statistical measure that represents the proportion of the variance in the target variable. It can take values between 0 and 1, whereby 1 indicates the model fits the data perfectly without any difference between the predicted and actual value. It applies only to linear regression models.

2.1.3. Root Mean Square (RMSE)

RMSE is a universal evaluation metric that allows us to compare the predictive performance of different types of models. It is the square root of the average of the squared residues.

$$RMSE = \sum \mathbb{E}(\hat{y} - y)^2$$

2.1.4. Mean Absolute Error (MAE)

MAE is another well-known metric to analysis the performance of the model. It is characterized by the following formula:

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

2.2. Preparation of the dataset for Linear Regression

2.2.1. Creation of the dummy variables

To be able to use the different models of Linear Regression, some preliminary tasks must be performed. First of all, the numerical and the qualitative variables are separated from each other's. The objective of this separation is to create dummy variables from categorical variables to be able to use it.

The `get_dummies` function is applied to fourth variables:

- `cod_cbr` (type of fuel)
- `Hybrid` (Yes/No)
- Type of Gearbox
- Number of reports on the gearbox

This transformation allows to go from 4 categorical variables to 31 dummy variables. After removing the target variable (CO_2 emissions), the dataset contains 38 variables (dummy variables and quantitative variables).

2.2.2. Creation of the training and test set

The objective of this part is to separate the dataset into a training and test sets. Two options are possible:

- Apply the function `train_test_split` with a test size between 20% and 30%
- Use a complete year of the dataset as the test dataset if the split is acceptable

The variables are splitted by year as the following:

Year	Number of Values (%)
2012	2.6%
2013	32.1%
2014	46.2%
2015	19.0%

As we can observe, the last year of the dataset represents almost 20% of the overall dataset. Therefore, the test set will be composed only of the values from 2015 and the training set will be composed of the values from the years 2012 to 2014.

Finally, the training set and test set are standardized using the function `StandardScaler`. The scaler is fitted on the training set and then transform the training set and the test set. It is now possible to start training the model.

2.3. Multiple Linear Regression

The multiple Linear Regression consists in modelling a relation between multiple explanatory variables and the target variable such as:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{i,j} + \epsilon_i$$

2.3.1. First attempt using StandardScaler

The first attempt has been done using the function StandardScaler as explained in section 2.2.2. The results are the following:

Figure 2-1: Main results of the Linear Regression using StandardScaler

	R^2
Train	0.9953
Test	-111308174133481734930432

We can see in the table above that there is a clear overfitting on the Test set for not a clear reason.

There is two possibilities to solve this problem:

- Considering a new scaler to normalize our data (i.e. RobustScaler in our case, see in the next section)
- Use of Regularized Linear Regression to remove some variables (refer to section 2.4)

2.3.2. Second attempt using RobustScaler

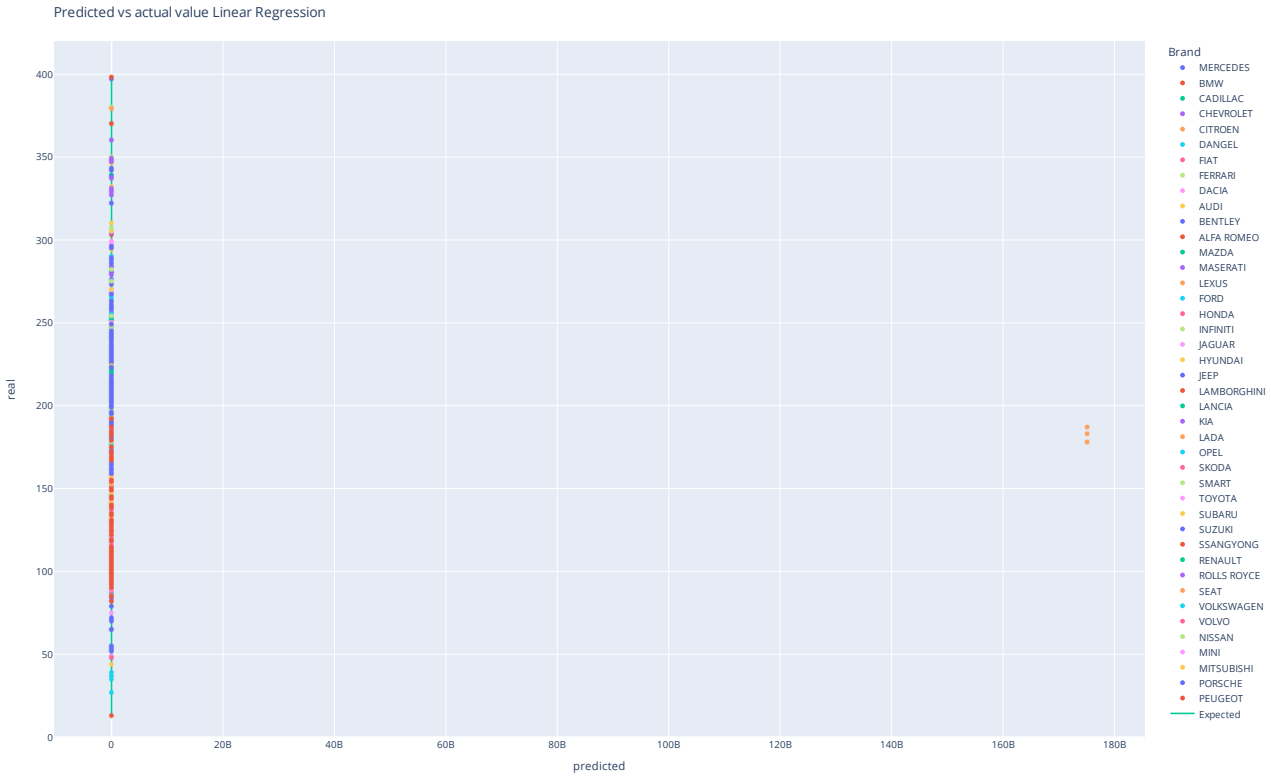
The second attempt has been done using this time the function RobustScaler. The main results of the Linear Regression Model is presented in the following table:

Figure 2-2: Main results of the Linear Regression using StandardScaler

	R^2
Train	0.9953
Test	-29816200138904276

As we can see in the results above and in the graph below the problem of overfitting persists. We will then try to solve it by using regularized Linear Regression.

Figure 2-3: Predicted vs Predicted value from the Linear Regression Model



Further investigation is done in Appendix I to verify with statistical tests if linear models are a good option to predict the behavior of our data set.

2.4. Regularized Linear Regression[1]

The objective of regularization is to avoid overfitting, i.e. learning from the sample of training data, but not too much (no overdependence of the model). To do this, the principle is to accept a slight increase in bias to achieve a proportional reduction in variance. The idea is to regulate the modelling a little more firmly by imposing constraints on the estimated regression coefficients (constraints on the values that the βi as a whole may take).

This is called shrinkage: the ranges of values that can be taken by the estimated parameters are narrowed.

Unlike conventional regression, where variables are generally kept as measured, it is common practice to center and reduce the explanatory variables, to avoid that variables with high variance have too much influence. In general, the value of βi depends on the scale of the associated explanatory variable. When calculating the norm, in order not to penalize or favor a coefficient, it is desirable that each coefficient be affected in a "similar" way. One way to achieve this balance is to center and reduce all variables.

The target variable must be centered to remove the regression constant (which should not be penalized), and it can also be reduced.

2.4.1. Ridge Regression

The Ridge regression consists in adding a constraint on the coefficients during modeling to control the amplitude of their values. This constraint is expressed by adding a penalty function - in the form of a L2 norm of the coefficients - to the sum of the squares of the residuals that we are trying to minimize. This penalty function is accompanied by a penalty coefficient (α) to be set, which allows to control the impact of the penalty.

Thus, the Ridge regression can be written:

$$\min_{\beta}\left(||Y - X\beta||^2 + \alpha||\beta||^2\right)$$

Or

$$\min_{\beta_1,\dots\beta_p}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p}\beta_j x_{i,j}\right)^2 + \alpha\sum_{j=1}^{p}\beta_j^2$$

The choice of the α parameter, or the penalty coefficient, is important. As a general rule, it is determined on a predictive performance criterion basis.

The principle the following: we set a range of values of α over which we evaluate the model's performance, and we choose the α that minimizes an error criterion.

The class `Ridge` of the module `sklearn.linear` model allows to create a Ridge regression model in the same way as a classic linear regression, to which a parameter α is provided in addition, by default =1.

To obtain the best possible predictive model, it is preferable to use the class `RidgeCV` to which we pass a list of values for α via the parameter alphas. The function will go through the list to create and evaluate several models by cross-validation, then it will select α by the best performance.

For this study the `RidgeCV` is created with the following parameters:

```
k_folds = KFold(n_splits = 5)
alphas = [
    1e-5, 1e-4, 1e-3, 1e-2, 0.1, 0.3,
    0.5, 0.7, 1, 5 , 10, 20, 30, 40,
    75, 100, 200, 500, 700, 1000,
    10000, 50000]
```

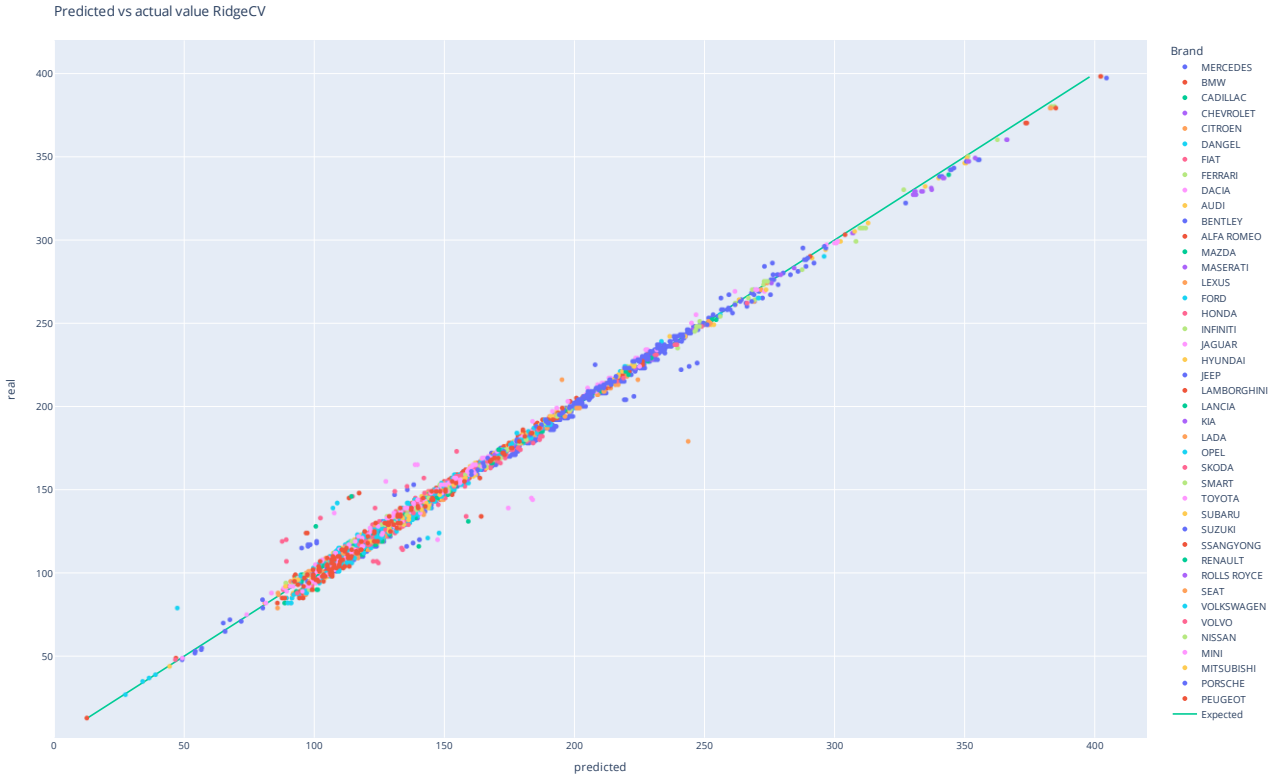
[1] Ref. Module 124 – Regression – section 3

The table below shows the results for the RidgeCV model:

Table 2-1: Main results of the RidgeCV model

	R^2	MAE	MSE	RMSE
Train	0.9953	1.41	6.43	2.54
Test	0.9929	2.68	13.44	3.67

As we can observe with the main metrics, the model already presents very good results on the training and the test set by considering all the variables. The scatterplot shown below confirms the very good predictions of the model.



Nevertheless, the model seems to have some difficulties predicting some parts of the dataset, especially some cars between with a CO_2 emission of 100 and 200 g/km. One of the reasons may be because of the very large quantities of data in this zone. It seems to affect all brands.

The second part where the model have some difficulties to predict is between 200 and 250 g/km and it concerns mainly the brand Mercedes.

2.4.2. Lasso Regression

Lasso regression is similar to Ridge regression, the only difference between the two being that with Lasso the constraint is put on the L1 norm, rather than on the L2 norm.

The Lasso regression can therefore be written as follows:

$$\min_{\beta_1,\dots\beta_p} \sum_{i=1}^{n}\left(y_i - \sum_{i=j}^{p}\beta_j x_{i,j}\right)^2 + \alpha\sum_{i=j}^{p}|\beta_j|$$

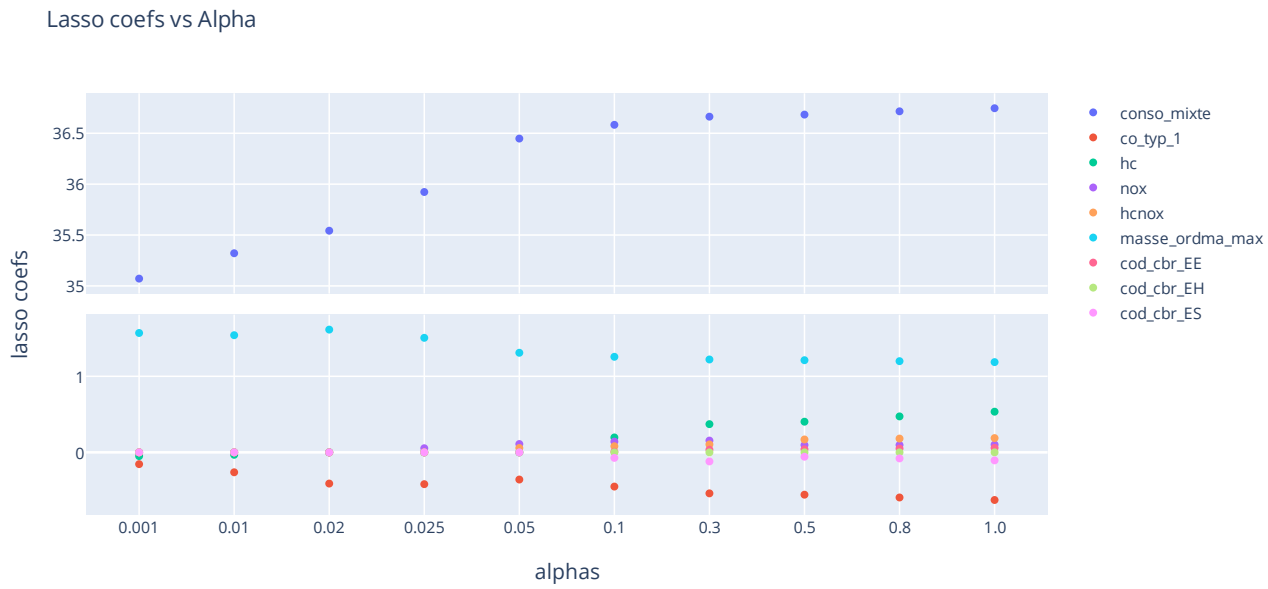
The main interest of Lasso regression, compared to Ridge regression, is that it can serve as a tool for selecting variables by canceling certain coefficients βj. Indeed, some variables can have their coefficient estimated to be zero and be de facto excluded from the predictive model. Obviously, the higher the penalty coefficient, the higher the number of coefficients estimated to be 0.

The function `lasso_path` allows to produce the estimated coefficients corresponding to the different α that it receives as arguments.

The Lasso path for this study is set with the following values for α:

```
1   alpha_list = (0.001, 0.01, 0.02, 0.025, 0.05, 0.1, 0.3, 0.5, 0.8, 1.0)
```

The image below shows the selected coefficient for each alpha:



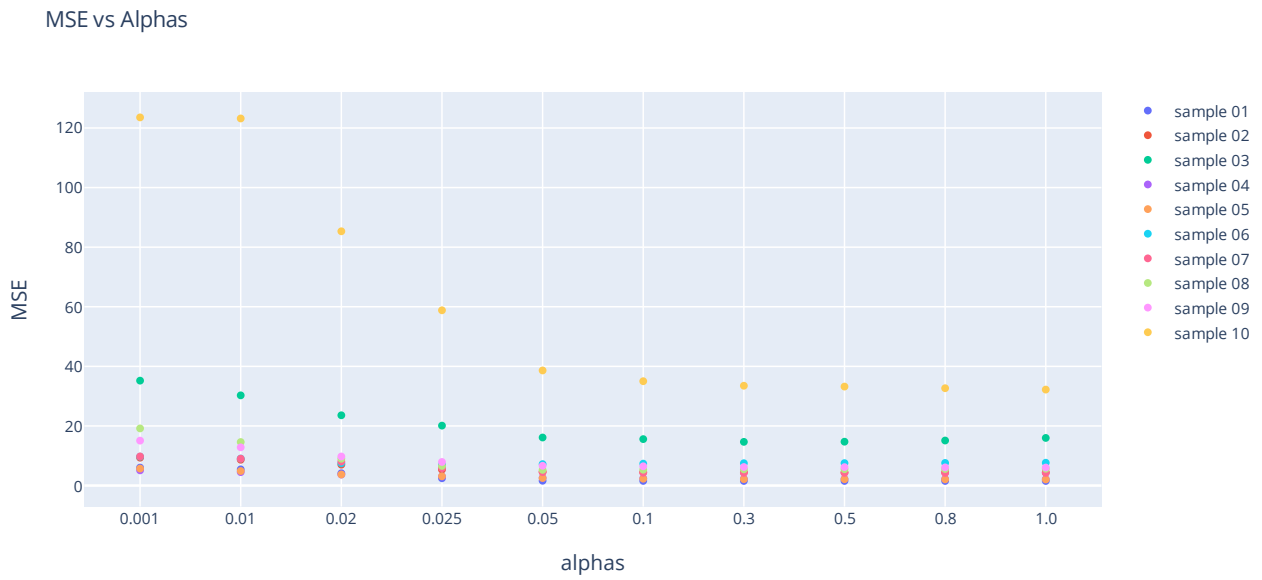
We can see that the variable conso_mixte has a much higher coefficient compared to the other variables (upper subplot). More features start being selected for $\alpha > 0.1$.

As for the Ridge regression, there is a class LassoCV which allows to find the optimal α by cross-validation based on predictive performance.

For this study the `LassoCV` is created with the following parameters:

```
1   alpha_list = (0.001, 0.01, 0.02, 0.025, 0.05, 0.1, 0.3, 0.5, 0.8, 1.0)
2   model_lasso = LassoCV(cv=10, alphas=alpha_list)
```

The graph below shows the MSE as a function of each alpha for the LassoCV model:



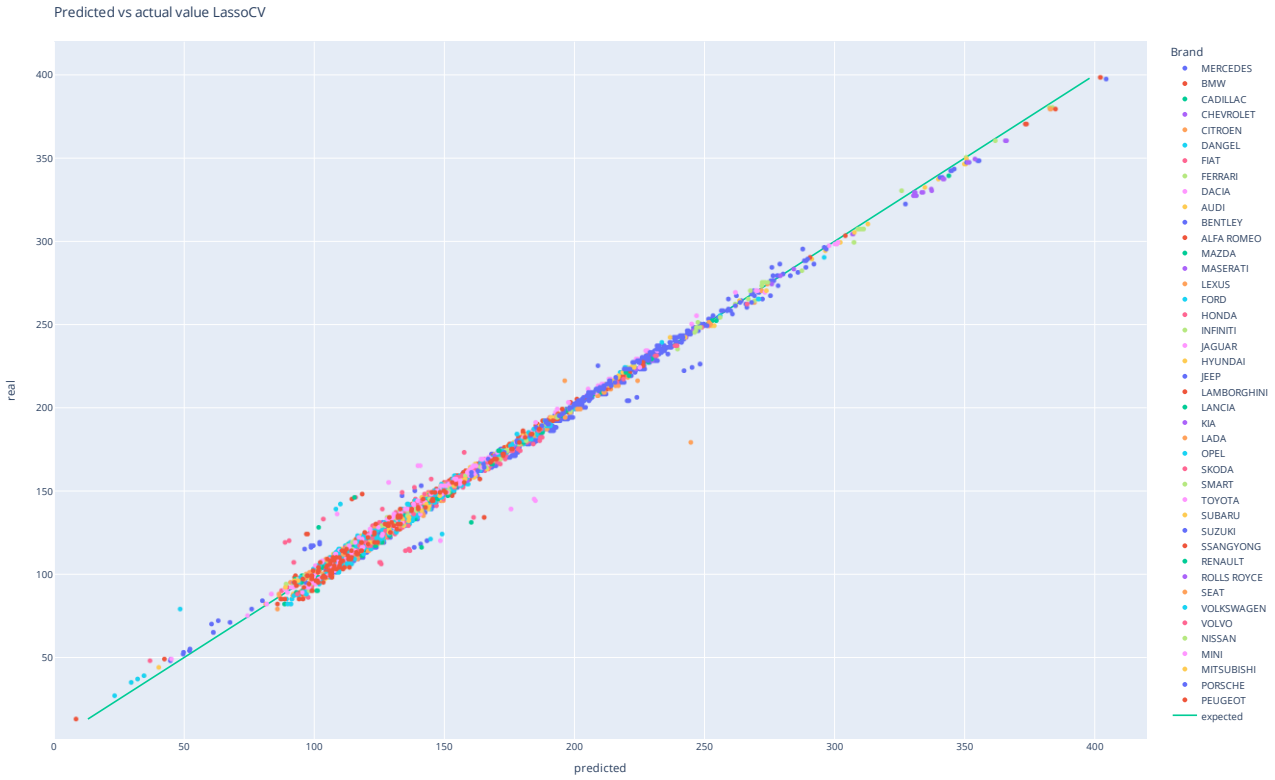
We can see that for all the Cross Validation samples, the MSE is minimised for α greater than or equal to 0.1.

The table below shows the results for the LassoCV model:

Table 2-2: Main results of the LassoCV model

	R^2	MAE	MSE	RMSE
Train	0.9953	1.42	6.44	2.54
Test	0.9929	2.70	13.51	3.67

The results are very close to the ones obtained by the LassoCV model. The scatterplot shown below confirms the very good predictions of the model.



2.4.3. Elastic Network[2]

Elastic Net combines Lasso and Ridge Regression and penalty coefficient is a linear combination of L1 and L2 penalties. In other words, the Elastic Net regression is written:

$$\min_{\beta_1,\dots\beta_p} \sum_{i=1}^{n} \left(y_i - \sum_{i=j}^{p} \beta_j x_{i,j} \right)^2 + \alpha \left(\lambda \sum_{i=j}^{p} |\beta_j| + (1-\lambda) \sum_{i=j}^{p} \beta_j^2 \right)$$

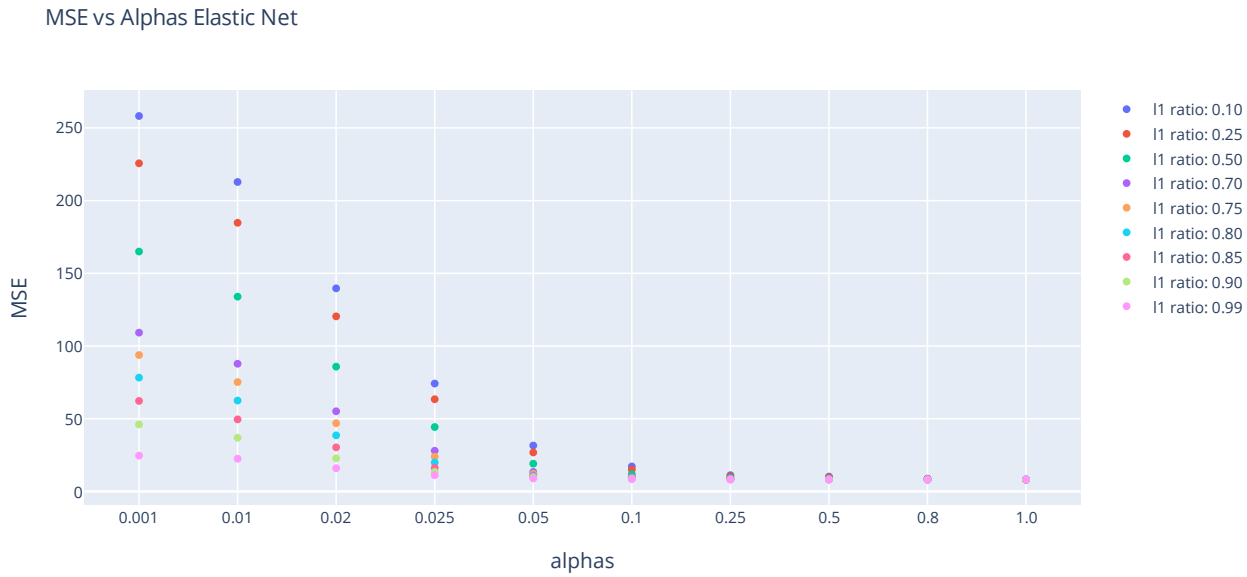
The advantages of this combination are mainly: Maintaining LASSO's capability to exclude irrelevant variables (variable selection) and sharing weights between correlated variables with no arbitrary selection. In scikit-learn, the term λ is represented by the parameter `l1_ratio`. `l1_ratio=1` therefore corresponds to a Lasso regression, while `l1_ratio=0` corresponds to a Ridge regression. The other values between 0 and 1 correspond to a combination of the two regularizations.

The function `ElasticNetCV` allows to find the best parameters λ and α by cross validation.

For this study the `ElasticNetCV` is created with the following parameters:

```
model_en = ElasticNetCV (
    l1_ratio = (0.1, 0.25, 0.5, 0.7, 0.75, 0.8, 0.85, 0.9, 0.99),
    alphas= (0.001, 0.01, 0.02, 0.025, 0.05, 0.1, 0.25, 0.5, 0.8, 1.0),
    cv = 8
)
```

The image below shows the average value of MSE (all CV samples) for each alpha and for each L1 ratio.



We can see that the MSE is very low for alphas greater than 0.05 and that for an that the minimum value is reached for alpha around 1.

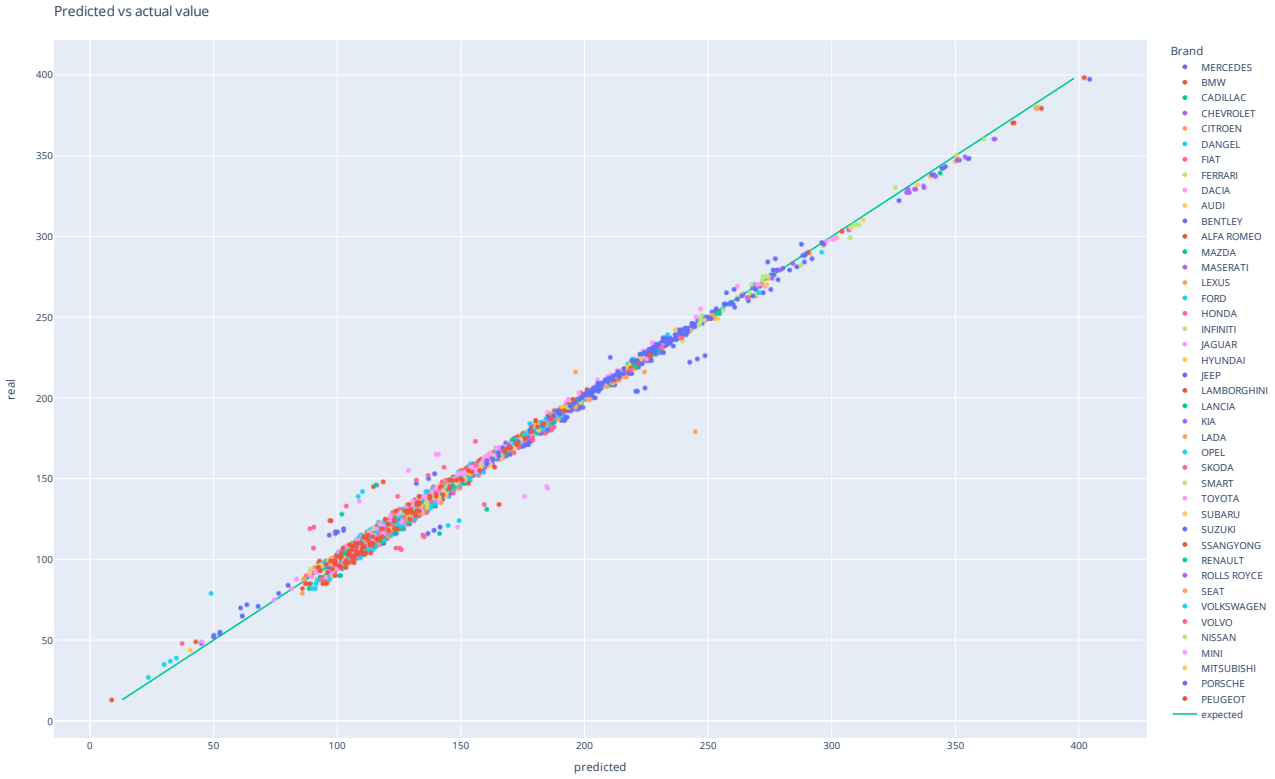
[2] Ref. Module 124 – Regression – section 4

The table below shows the results for the `ElasticNetCV` model:

Table 2-3: Main results of the `ElasticNetCV` model

	R^2	MAE	MSE	RMSE
Train	0.9953	1.42	6.43	2.54
Test	0.9928	2.70	13.52	3.68

The results are very close to the ones obtained by the `ElasticNetCV` model. The scatterplot shown below confirms the very good predictions of the model.



In Appendix II plots for different years and for some brands are displayed individually to better visualize the performance of the models. This way we can better identify the points out of the curve above. They come mainly from brands with low representation in the dataset like DACIA meaning that this can come from an unbalance problem in the dataset for certain brands. The linear regressions nevertheless still do a very good job predicting the CO2 emissions overall.

2.4.4. Summary

The graph below shows the summary of all the Regularized Linear Regression models:

We can see that all of them perform very well. We managed to fix the problem of overfitting and get a score above 99% for our test dataset. Although the models have slightly different MSE for the test set, their differences are negligible, and we can say that their performance are the same. Any of the 3 regularized models developed in this study can then be used as a good estimator for CO_2 emissions prediction.

3. CLASSIFICATION

Classification is a supervised learning method where a model predicts the label or class of input data fed. There are certain classification types used:

- Binary Classification: The key objective is to classify input data into one of two classes or categories. For example, determining whether email is spam or not, or whether a person has a certain disease based on health conditions.
- Multiclass Classification: Here, input is classified into several classes or categories. For instance, identifying the species of a flower based on its features.

To apply classification algorithms, CO_2 emissions are divided into four classes based on quartiles of CO_2 emissions distribution:

- Low CO_2 emissions <= Q1
- Medium CO_2 emissions > Q1 and <= Q2
- High CO_2 emissions > Q2 and <= Q3
- Very High CO_2 emissions >Q3

Note that for all models below train data corresponds to all data available except for year 2015 and test data corresponds to the year 2015.

One important hypothesis for assessing the performance and the pertinence of the model consists in considering that cars classified in a lower class are less acceptable than cars classified in a higher emissions class. For example, classifying a high emissions car in medium emissions class is not acceptable while medium emissions cars classified as high emissions class is tolerated.

3.1. Performance Criteria

The performance of the classification models is evaluated based on the following criteria:

3.1.1. Confusion Matrix

A confusion matrix represents the predictive performance of a model on a dataset. For a binary class dataset (which consists of, suppose, 'positive' and 'negative' classes), a confusion matrix has four essential components:

- True Positives (TP): Number of samples correctly predicted as 'positive'.
- False Positives (FP): Number of samples wrongly predicted as 'positive'.
- True Negatives (TN): Number of samples correctly predicted as 'negative'.
- False Negatives (FN): Number of samples wrongly predicted as 'negative'.

		Actual	
		+ve	-ve
Predicted	+ve	TP	FP
	-ve	FN	TN

3.1.2. Accuracy

The accuracy metric computes how many times a model made a correct prediction across the entire dataset. This can be a reliable metric only if the dataset is class-balanced; that is, each class of the dataset has the same number of samples.

3.1.3. Precision

Precision measures how many of the "positive" predictions made by the model were correct.

$$Precision = \frac{TP}{TP + FP}$$

3.1.4. Recall

Recall measures how many of the positive class samples present in the dataset were correctly identified by the model.

$$Recall = \frac{TP}{TP + FN}$$

3.1.5. F1 Score

The F1 score is calculated as the harmonic mean of the precision and recall scores, as shown below. It ranges from 0-100%, and a higher F1 score denotes a better-quality classifier.

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

The harmonic mean encourages similar values for precision and recall. That is, the more the precision and recall scores deviate from each other, the worse the harmonic mean.

3.2. Simple Models

This section covers the various simple classification models used to make predictions for CO_2 emissions based on the input data given. These models are namely Logistic Regression, Support Vector Machines (SVM), K-Nearest Neighbour (KNN) and Decision Tree Classifier.

3.2.1. Logistic Regression

Logistic Regression is one of the techniques used for classification.

This statistical method studies the relationship between the explanatory and target variable by predicting the probability that an event occurs or not from the optimization of the regression coefficients.

A graphical representation of the Classification Report for the Logistic Regression model trained considering default model parameters (e.g. penalty = L2, C=1…) is show in the graph here-under.

Confusion Matrix:

Real \ Predicted	low	medium	high	very high
low	**12 723**	59	0	0
medium	55	**4 276**	289	1
high	0	10	**537**	17
Very high	0	0	0	**1 638**

The model using default parameters give good precision, recall and `f1_score` for Low, Medium and Very High CO_2 emissions. However, the model is not that good for predicting High emissions category as its precision is around 65%. 35% of High emissions cars are classified as medium which is not on the conservative side from an environmental point of view.

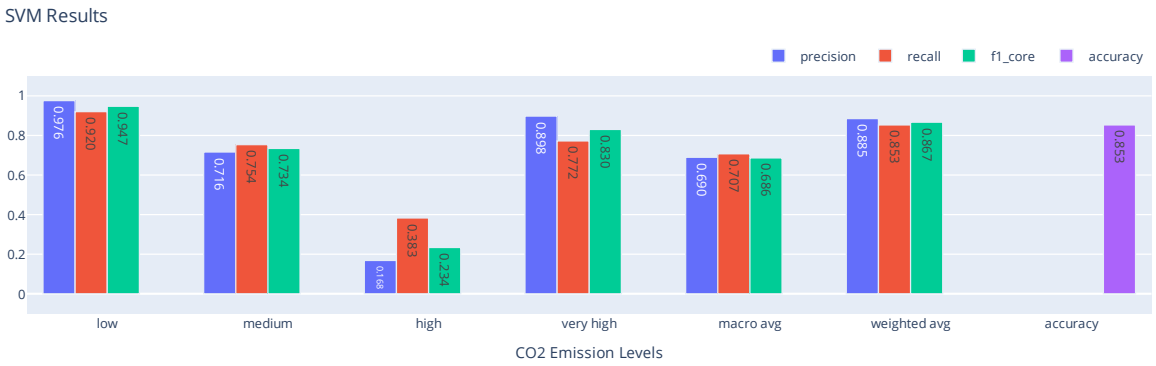
3.2.2. Support Vector Machine (SVM)

SVM technique allows to deal with non-linear discrimination problems and reformulate the problem as a quadratic optimization problem thanks to the Kernel trick.

Results below correspond to a SVM model trained considering the following parameters:

```
gamma = 0.01
kernel = 'poly'
```

Other parameters are by default, in particular the degree of the polynomial kernel function = 3. A graphical representation of the Classification Report for the SVM trained model is shown below:



Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**11 764**	993	25	0
	medium	283	**3 482**	856	0
	high	4	200	**216**	144
	Very high	0	186	187	**1 265**

Accuracy of the model is 85% and performance is lower compared to Logistic regression for classifying Medium and High CO_2 emitters. This means that the relationship between the explanatory and target variable is closer to a linear behaviour than a polynomial 3rd degree. The model is not good at all for classifying High emitters which is not acceptable for cars classified as Medium instead of high emitters.

3.2.3. K-Nearest Neighbour (KNN)

KNN (K-Nearest Neighbors) is a non-parametric algorithm, meaning it does not make any assumptions about the underlying data distribution.

In KNN, the algorithm works by finding the K closest data points in the training set to a new data point, based on a distance metric such as Euclidean distance. The algorithm then assigns the new data point to the class or value that is most common among its K nearest neighbors.

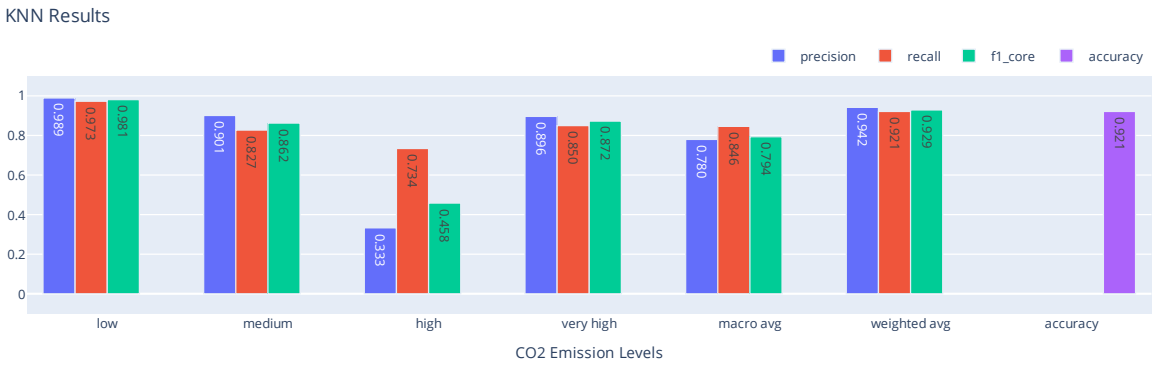
For example, in a classification task, if K=3 and two of the three nearest neighbors belong to class A and one belongs to class B, the algorithm would classify the new data point as belonging to class A.

KNN is a simple and intuitive algorithm that can work well for small datasets and low-dimensional feature spaces. However, it can be computationally expensive for large datasets and high-dimensional feature spaces, and it may not perform well if the data is not well-structured or if there are irrelevant features.

Results below correspond to a KNN model trained with the following parameters:

```
n_neighbors = 3
metric = 'minkowski'
```

Other parameters were left to their default values. A graphical representation of the Classification Report for the KNN trained model is shown below:



Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**12 434**	348	0	0
	medium	112	**3 822**	625	62
	high	1	50	**414**	99
	Very high	20	22	204	**1 392**

The results show that the KNN model is more accurate than the SVM model, producing a model that is 92% accurate. However, once again, we see that the model poorly predicts for the high emissions class and incorrectly places a lot of predictions in the medium class category, which is not acceptable for our study.

3.2.4. Decision Tree Classifier

Decision Trees work by recursively partitioning the feature space into smaller and smaller regions, based on the values of the input features, until a stopping criterion is met.

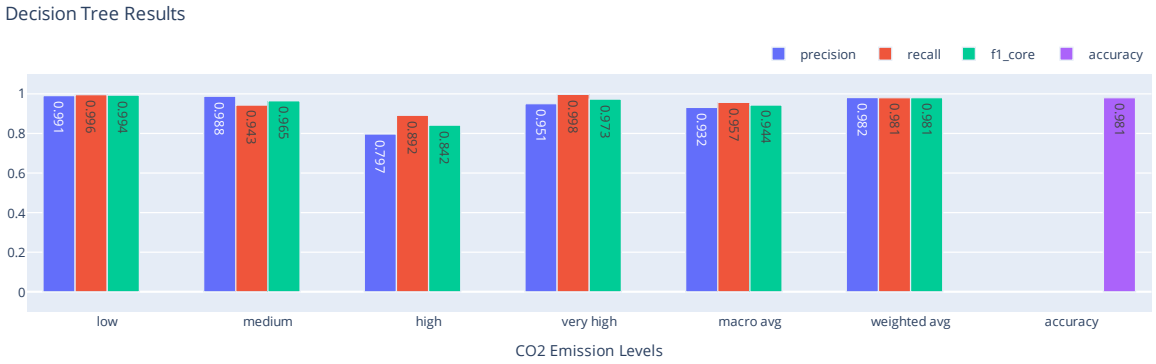
At each step of the partitioning process, the algorithm selects the feature that best separates the data into different classes or values, based on a measure of impurity such as Gini impurity or entropy. The feature is then used to split the data into two or more subsets, and the process is repeated on each subset until the stopping criterion is met.

Decision Trees are easy to interpret and visualize, and they can handle both categorical and numerical data. However, they can be prone to overfitting if the tree is too deep or if the data is noisy, and they may not perform well if the feature space is too large or if there are irrelevant features.

Results below correspond to a Decision Tree model trained with the following parameters:

```
1  Max_depth = 5
```

Other parameters were left to their default values. A graphical representation of the Classification Report for the Decision Tree trained model is shown below:

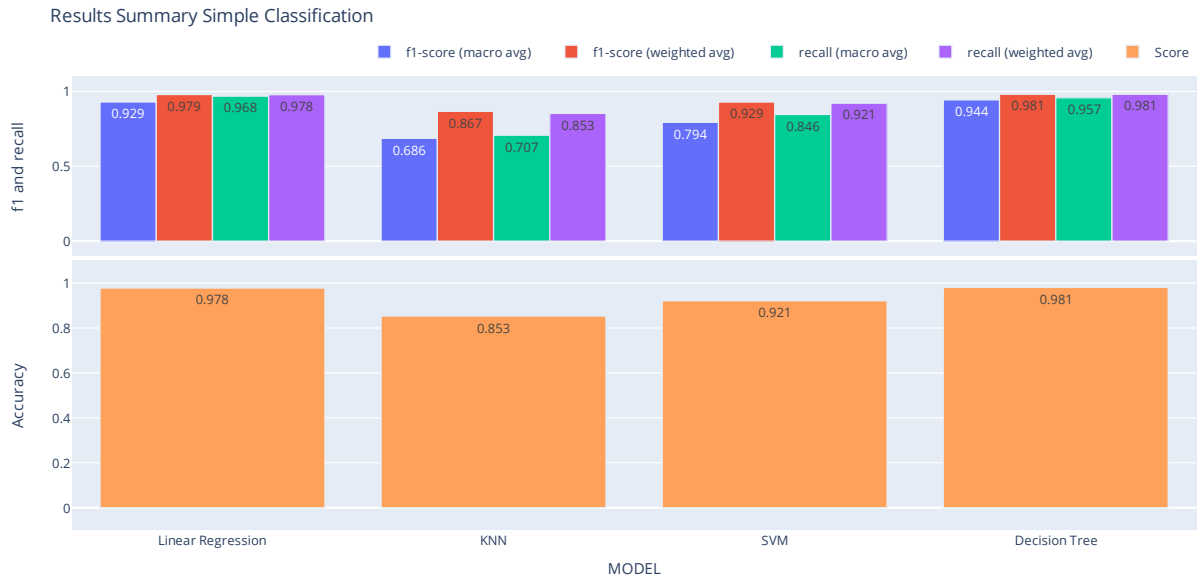


Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**12 731**	51	0	0
	medium	110	**4 359**	124	28
	high	2	2	**503**	57
	Very high	0	0	4	**1 634**

The results show that the Decision Tree model is 98% accurate. However, once again, we see that the model performs the worst for the high emissions class and incorrectly places a lot of predictions in the medium class category, which is not acceptable for our study. Although, this occurrence is not as bad as the KNN method.

3.2.5. Summary



Overall, these analysis shows that the Decision Tree model gives the most accurate results at 98% and thereby is the most performant. The next section will explore methods in which the results can be improved even further.

3.3. Boosting and Bagging[3]

Boosting and Bagging are two ensemble learning techniques used to improve the performance of models by combining the predictions of multiple models. This approach reduces variance or bias, thereby improving the final performance.

Both methods operate in a similar way and comprise of 2 main steps:

- Build different simple Machine Learning models on subsets of the original data
- Produce a new model from the assembly of the previous ones

3.3.1. Boosting

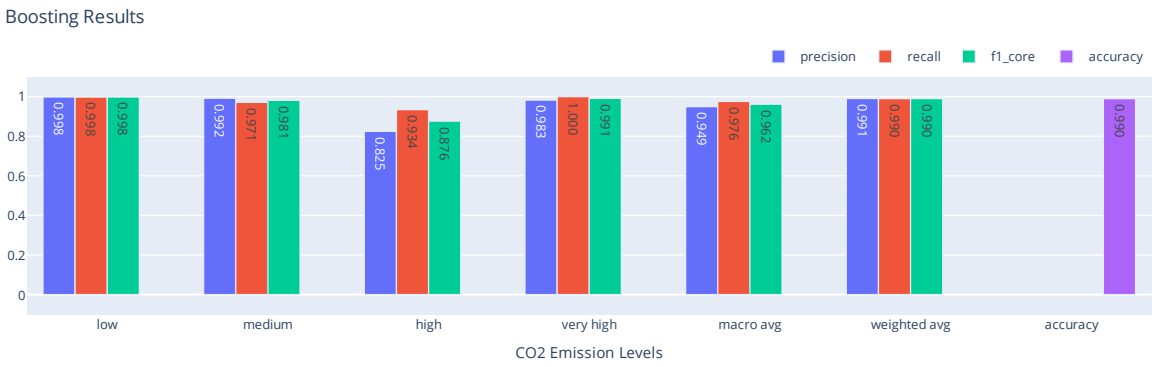
Boosting is a method that aims to reduce bias of simple and weak Machine Learning models and convert them into a stable and powerful model. The general principle of boosting is to build a family of recursively constructed "weak" estimators, which are then aggregated by a weighted average of the estimates (in regression) or a majority vote (in classification). By weak, we mean a decision rule whose error rate is slightly better than that of a purely random rule.

Each estimator is an improved version of the previous one, which aims at giving more weight to the poorly fitted or poorly predicted observations. Thus, at each iteration, the evaluation of the estimator allows a resampling of the data, with a greater weight given to the poorly predicted observations. The estimator built at step i will therefore concentrate its efforts on the observations poorly fitted by the estimator at step i - 1. Finally, the classifiers are combined and weighted by coefficients associated with their respective predictive performances.

The AdaboostClassifier is trained with the following parameters:

```
base_estimator = DecisionTreeClassifier(max_depth = 5)
n_estimators = 400
```

Other parameters were left to their default values. A graphical representation of the Classification Report for the `AdaboostClassifier` trained model is shown below as well as the confusion matrix:



Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**12 695**	87	0	0
	medium	22	**4 487**	112	0
	high	0	8	**524**	32
	Very high	0	0	0	**1 638**

[3] Ref. Module 121 – Classification – section 5

The Boosting model managed to increase the accuracy of the model from 98 to 99% by slightly improving the performance of the model for all the four classes but specially for the 'medium' emissions in the simple Decision Tree Classifier.

3.3.2. Bagging

The term Bagging comes from the contraction of Boostrap Aggregating, it gathers a set of methods introduced by Leo Breiman (1996) aiming at reducing the variance and increasing the stability of Machine Learning algorithms used for classification or regression. The general method of Bagging consists mainly in training a model on different subsets of the same size as the initial sample, using the Bootstrap technique, i.e. a random draw with discount. The method thus builds a set of independent estimators, unlike Boosting, which are then aggregated (or bagged) into a meta-model, with a majority vote for the classification, and a mean for the regression.

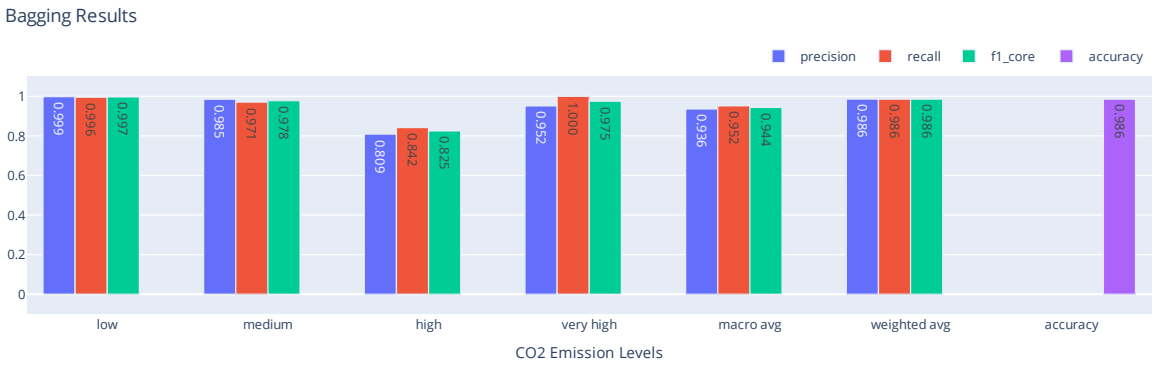
Unlike Boosting, choosing a large number of estimators will not result in additional risk of overfitting. Indeed, the higher the number of estimators, the more the bias of the final model will be equivalent to the average of the aggregated biases and the variance will decrease as the estimators that are aggregated will be decorrelated. It is therefore in our interest to choose the highest possible number of estimators, depending on the time we wish to allow for the training process.

The AdaboostClassifier is trained with the following parameters:

```
n_estimators = 500
oob_score = True        # Out Of Bag (OOB) error
```

he class `BaggingClassifier` of the package `sklearn.ensemble` allows to create a classifier using the Bagging algorithm from default classification trees.

Other parameters were left to their default values. A graphical representation of the Classification Report for the `BaggingClassifier` trained model is shown below as well as the confusion matrix:



Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**12 723**	59	0	0
	medium	17	**4 488**	112	4
	high	0	10	**482**	72
	Very high	0	0	0	**1 638**

The Baggin model also managed to slightly improve the accuracy of the mode compared to the simple Decisions three classifier and presents a performance close to the 99% of the boosting model.

3.3.3. Summary

A comparison between the performance of the Decision tree classifier, the Bagging and the boosting models is shown in the graph below:

All the models perform very well, Boosting being slightly above the other two in all scores.

We can try to further improve the classification by looking for the best hyperparameters for each model. This is what we will do in the next section, by using a Grid Search.

3.4. Grid Search[4]

In most cases, when several hyperparameters are to be set, and one does not know which ones to use to obtain the best possible model, the most efficient strategy is to create a search grid. We indicate the parameters to be varied. Then, thanks to the `GridSearchCV()` function of the module `model_selection`, the parameters are crossed and a model is created and evaluated for each possible combination by cross validation.

3.4.1. Logistic Regression

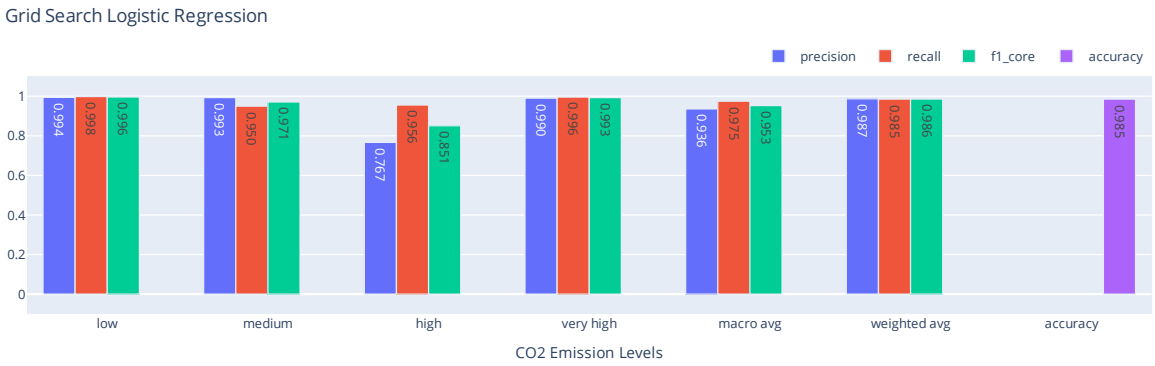
A grid Search is applied to the Logistic Regression with the following parameters:

```
C = [0.1, 1, 10]
kernel = ['rbf', 'linear', 'poly']
gamma = [0.001, 0.1, 0.5]
```

The scores for each combination of the parameters are shown here-under. The best classifier found by the Grid Search is highlighted in **bold**.

	params	mean_test_score
5	**{'C': 100, 'solver': 'newton-cg'}**	**0.989336**
3	{'C': 50, 'solver': 'newton-cg'}	0.987279
1	{'C': 30, 'solver': 'newton-cg'}	0.984768
4	{'C': 100, 'solver': 'liblinear'}	0.839055
2	{'C': 50, 'solver': 'liblinear'}	0.838517
0	{'C': 30, 'solver': 'liblinear'}	0.837799

A graphical representation of the Classification Report for the Logistic Regression Grid Search model trained with the best parameters is shown below as well as the confusion matrix:



Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**12 761**	21	0	0
	medium	75	**4 389**	157	0
	high	0	9	**539**	16
	Very high	0	0	7	**1 631**

The grid search manages to improve the performance of the Logistic regression from 97.9% to 98.5% and get very close to the results of the Boosting classifier, which is the best we have until now.

[4] Ref. Module 121 – Classification – section 2

3.4.2. SVM

A grid Search is applied to the SVM with the following parameters:

```
kernel = ['linear', 'rbf']
C = [1, 10, 50, 100]
gamma = [1, 0.1]
```

The scores for each combination of the parameters are shown here-under. The best classifier found by the Grid Search is highlighted in **bold**.

	params	mean_test_score
4	**{'C': 10, 'gamma': 1, 'kernel': 'linear'}**	**0.992647**
6	{'C': 10, 'gamma': 0.1, 'kernel': 'linear'}	0.992647
12	{'C': 100, 'gamma': 1, 'kernel': 'linear'}	0.991356
14	{'C': 100, 'gamma': 0.1, 'kernel': 'linear'}	0.991356
8	{'C': 50, 'gamma': 1, 'kernel': 'linear'}	0.991284
10	{'C': 50, 'gamma': 0.1, 'kernel': 'linear'}	0.991284
0	{'C': 1, 'gamma': 1, 'kernel': 'linear'}	0.987518
2	{'C': 1, 'gamma': 0.1, 'kernel': 'linear'}	0.987518
15	{'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}	0.933622
11	{'C': 50, 'gamma': 0.1, 'kernel': 'rbf'}	0.932235
7	{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}	0.929234
3	{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}	0.888574
13	{'C': 100, 'gamma': 1, 'kernel': 'rbf'}	0.828795
5	{'C': 10, 'gamma': 1, 'kernel': 'rbf'}	0.828066
9	{'C': 50, 'gamma': 1, 'kernel': 'rbf'}	0.827875
1	{'C': 1, 'gamma': 1, 'kernel': 'rbf'}	0.818836

A graphical representation of the Classification Report for the SVM Grid Search model trained with the best parameters is shown below as well as the confusion matrix:

Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**12 761**	21	0	0
	medium	69	**4 432**	120	0
	high	0	9	**540**	15
	Very high	0	0	7	**1 631**

The grid search manages to greatly improve the performance of the SVM from 85.3% to 98.8% and get close accuracy compared to the Boosting classifier.

3.4.3. K-Nearest Neighbor (KNN)

A grid Search is applied to the KNN with the following parameters:

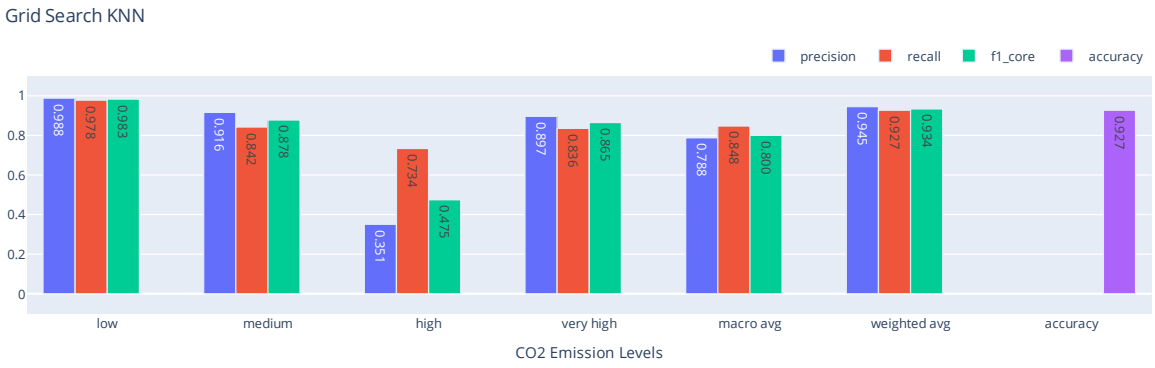
```
n_neighbors = [2, 3, 5]
algorithm = ['auto', 'ball_tree', 'kd_tree']
```

The scores for each combination of the parameters are shown here-under. The best classifier found by the Grid Search is highlighted in bold.

	params	mean_test_score
0	**{'algorithm': 'auto', 'n_neighbors': 2}**	**0.850448**
3	{'algorithm': 'ball_tree', 'n_neighbors': 2}	0.850448
6	{'algorithm': 'kd_tree', 'n_neighbors': 2}	0.850448
1	{'algorithm': 'auto', 'n_neighbors': 3}	0.848068
4	{'algorithm': 'ball_tree', 'n_neighbors': 3}	0.848056
7	{'algorithm': 'kd_tree', 'n_neighbors': 3}	0.848056
2	{'algorithm': 'auto', 'n_neighbors': 5}	0.845713
5	{'algorithm': 'ball_tree', 'n_neighbors': 5}	0.845701
8	{'algorithm': 'kd_tree', 'n_neighbors': 5}	0.845701

The grid search presents the same result as the previous simple KNN model.

A graphical representation of the Classification Report for the KNN Grid Search model trained with the best parameters is shown below as well as the confusion matrix:



Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**12 503**	279	0	0
	medium	126	**3 893**	540	62
	high	1	53	**414**	96
	Very high	20	23	226	**1 369**

3.4.4. Random Forest[5]

Random Forest algorithms are a special case of Bagging applied to decision trees (CART). In addition to the Bagging principle, random forests add randomness to the variables. For each tree, a subsample is selected by bootstrapping individuals and at each step, the construction of a node of the tree is done on a subset of randomly drawn variables.

The working principle of random forests is simple: many small classification trees are produced on a random fraction of data. The random forest then votes these poorly correlated classification trees to infer the order and importance of the explanatory variables.

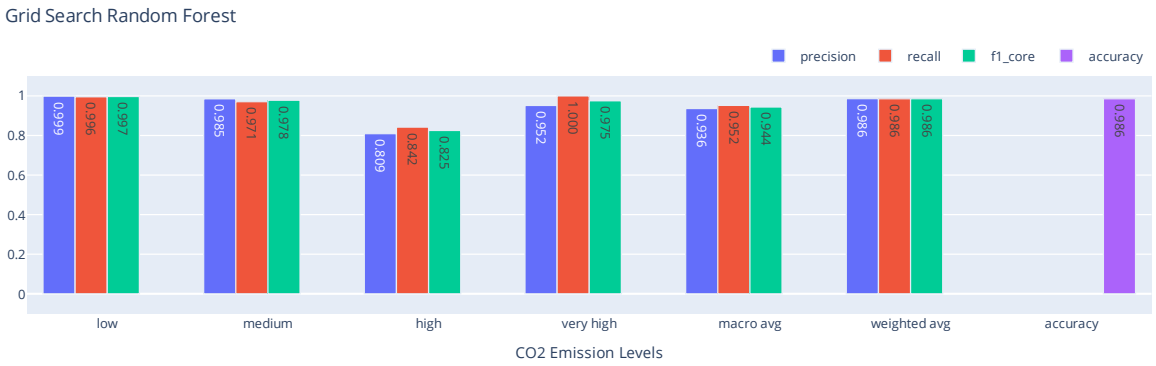
A grid Search is applied to the Random Forest Classifier with the following parameters:

```
max_features = ['sqrt', 'log2', None]
min_samples_split = [2, 5, 10, 30]
```

The scores for each combination of the parameters are shown here-under. The best classifier found by the Grid Search is highlighted in bold.

	params	mean_test_score
8	**{'max_features': None, 'min_samples_split': 2}**	**0.996270**
10	{'max_features': None, 'min_samples_split': 10}	0.995290
9	{'max_features': None, 'min_samples_split': 5}	0.995218
11	{'max_features': None, 'min_samples_split': 30}	0.994740
0	{'max_features': 'sqrt', 'min_samples_split': 2}	0.982055
1	{'max_features': 'sqrt', 'min_samples_split': 5}	0.971391
2	{'max_features': 'sqrt', 'min_samples_split': 10}	0.970422
3	{'max_features': 'sqrt', 'min_samples_split': 30}	0.968988
6	{'max_features': 'log2', 'min_samples_split': 10}	0.968127
5	{'max_features': 'log2', 'min_samples_split': 5}	0.967266
4	{'max_features': 'log2', 'min_samples_split': 2}	0.967206
7	{'max_features': 'log2', 'min_samples_split': 30}	0.965401

A graphical representation of the Classification Report for the Random Forest Grid Search model trained with the best parameters is shown below as well as the confusion matrix:



Confusion Matrix:

Real \ Predicted	low	medium	high	very high
low	**12 724**	58	0	0
medium	22	**4 483**	112	4
high	0	10	**474**	80
Very high	0	0	0	**1 638**

[5] Ref. Module 121 – Classification – section 6

The grid search presents a result close to the simple Decision Tree Classifier.

3.4.5. Summary

The graph below summarizes the results of all grid searchs. The best performing model is the SVM grid search with the following parameters:

```
kernel = 'linear'
C = 10
```

gamma = 1

The grid Search allowed us to bring the SVM from the worse position in previous section to the best performing model.

We can try to further improve these results by combining all of them into a single estimator. This is done by using a voting classifier, as is shown in the next section.

3.5. Voting classifier[6]

The Voting Classifier is a meta-classifier of scikit-learn allowing to combine several Machine Learning estimators, similar or conceptually different. Specifically, it consists in constituting a college of experts which is represented by models such as decision trees, the k nearest neighbors method or logistic regression, and then to make them vote. The VotingClassifier class of scikit-learn allows to perform a hard or soft vote.

- In hard voting, each classification model predicts a label, and the final label produced is the one predicted most frequently.
- In soft voting, each model returns a probability for each class, and the probabilities are averaged to predict the final class (only recommended if the classifiers are well calibrated).

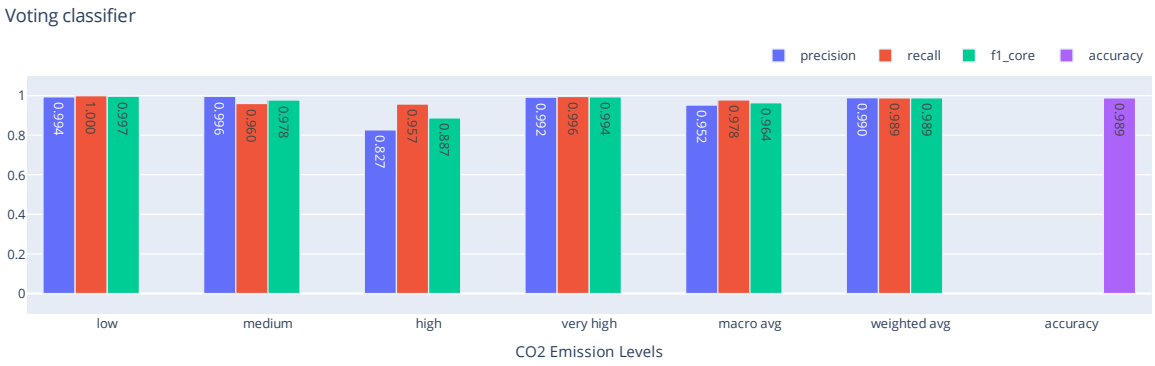
In both cases, it is possible to assign a weight to each estimator, allowing to give more importance to one or more models.

To try to further improve the model we combine the best of each classifier found in the previous sections:

```
lrg_clf = LogisticRegression(max_iter = 300, C = 100, solver = 'newton-cg')
svm_clf = svm.SVC(C = 10, kernel = 'linear', gamma = 1)
knn_clf = neighbors.KNeighborsClassifier(algorithm = 'auto', n_neighbors = 2)
rdf_clf = ensemble.RandomForestClassifier(max_features = None, min_samples_split = 2)
```

The classifier performs a hard voting.

A graphical representation of the Classification Report for the Voting Classifier is shown below as well as the confusion matrix:



Confusion Matrix:

		Predicted			
		low	medium	high	very high
Real	low	**12 776**	6	0	0
	medium	81	**4 436**	104	0
	high	0	11	**536**	17
	Very high	0	0	7	**1 631**

We notice that the voting classifier got almost the same result as the SVM grid Search and slightly better, thanks to the voting with the other models.

We managed to improve the Classification models using the different tools available for a data Scientist. The final classifier has a high performance and manages to correctly classify 99% of the samples.

[6] Ref. Module 121 – Classification – section 7

4. CONCLUSION

We managed to create and train high performance models for both problems presented in this project. Both final Regression and Classification models presents a score around 99%.

Beyond managing to train high performance models, this project allowed us to apply in a real-world problem the different techniques learned in the Data Upscaling program, both for Regression and Classification solving problems. We worked with the different kinds of Linear Regression (Classical, Ridge, Lasso and Elastic Net) to solve the overfitting problem and to improve our models. For Classification we used the main algorithms (Logistic Regression, K-Nearest Neighbours, SVM and Decision Tree Classifiers). And for improving the results we used Boosting and Bagging techniques followed by Grid Search and finally a Voting Classifier.

In addition to the modelling, this project also allowed us to face and learn how to solve recurrent problems in Data Science project development as ensuring reproducibility in different machines as well as errors coming from that (by sharing the packages requirements of the project). Which was not faced during the classes as all environments were managed by the Data Scientest servers.

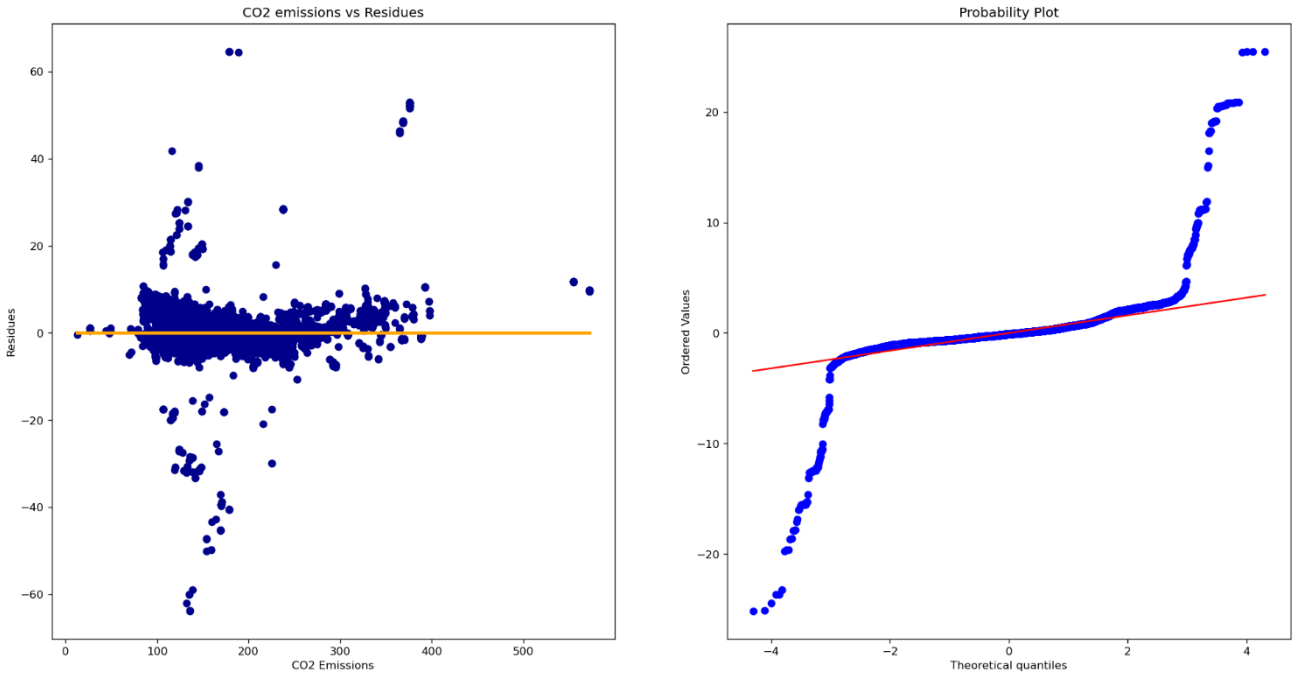
APPENDIX I. STATISTICAL ANALYSIS

Linear Regression is based on two assumptions about errors: homo-sedasticity and normality. The residues of linear model are studied to ensure the compliance of these assumptions

To do that, two different graphs are represented:

- The residues as a function of the target variable to have an estimate of the homoscedasticity of the residues
- A Quantile-Quantile or (Q-Q plot) diagram

Figure 4-1: Scatter plot of the residues vs the target variables and QQ-Plot for the Linear Regression model



The first scatter plot representing the residues vs CO2 emissions is analysed. Normally it should be randomly scattered around y = 0 with no apparent structure. Here it seems that there is no clear structure in their distribution and the points are evenly distributed around y = 0.

We can implement the Breusch-Pagan test to check if the homo-sedasticity is present. The hypothesis are the following:

- Null Hypothesis (H0): homo-sedasticity is present (the residuals are distributed with equal variance)
- Alternative Hypothesis (H1): Heteroscedasticity is present (the residuals are not distributed with equal variance)

The p-value is < 0.05 therefore we can reject the Null Hypothesis and we conclude H1. So, Heteroscedasticity is present

The Theoretical quantiles graph is now analyzed: The normality of the residues is validate if the point are aligned with the first bisector. Here we can see that this not the case.

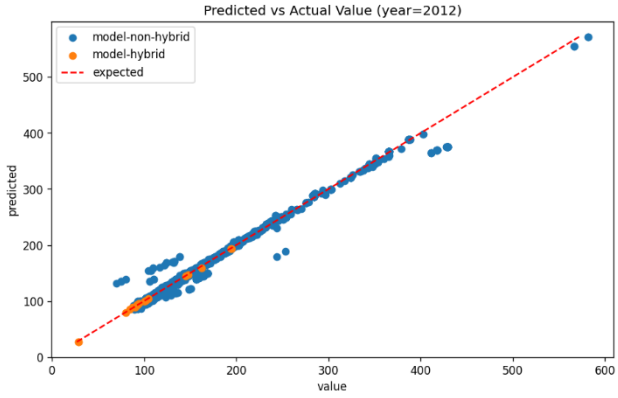
We can implement the Jarque-Bera Test to check the normality of the residues. The hypothesis are the following:

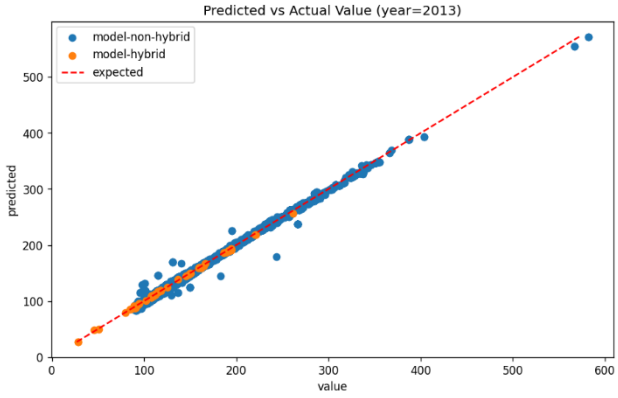
- Null Hypothesis (H0): Data follow a normal distribution.
- Alternative Hypothesis (H1): Data does not follow a normal distribution.

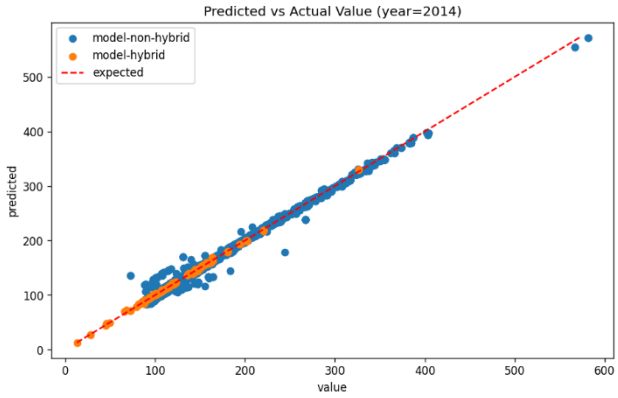
The p-value < 0.05 therefore we can reject the Null Hypothesis and we conclude H1. So, Data doesn't follow a normal distribution.

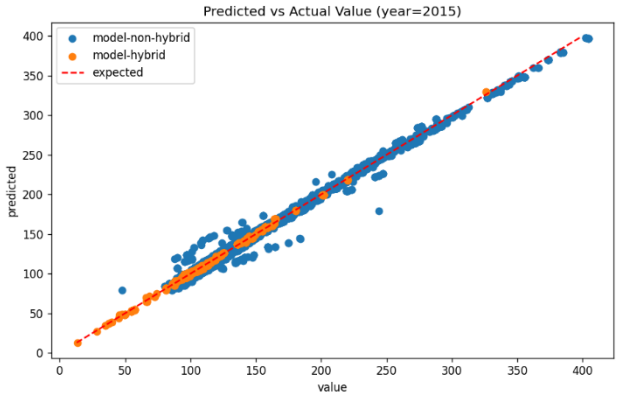
APPENDIX II. DATAVIZ REGRESSION PERFORMANCE

This appendix shows the results of predictions of the Elastic Net model in more details per year for all data:

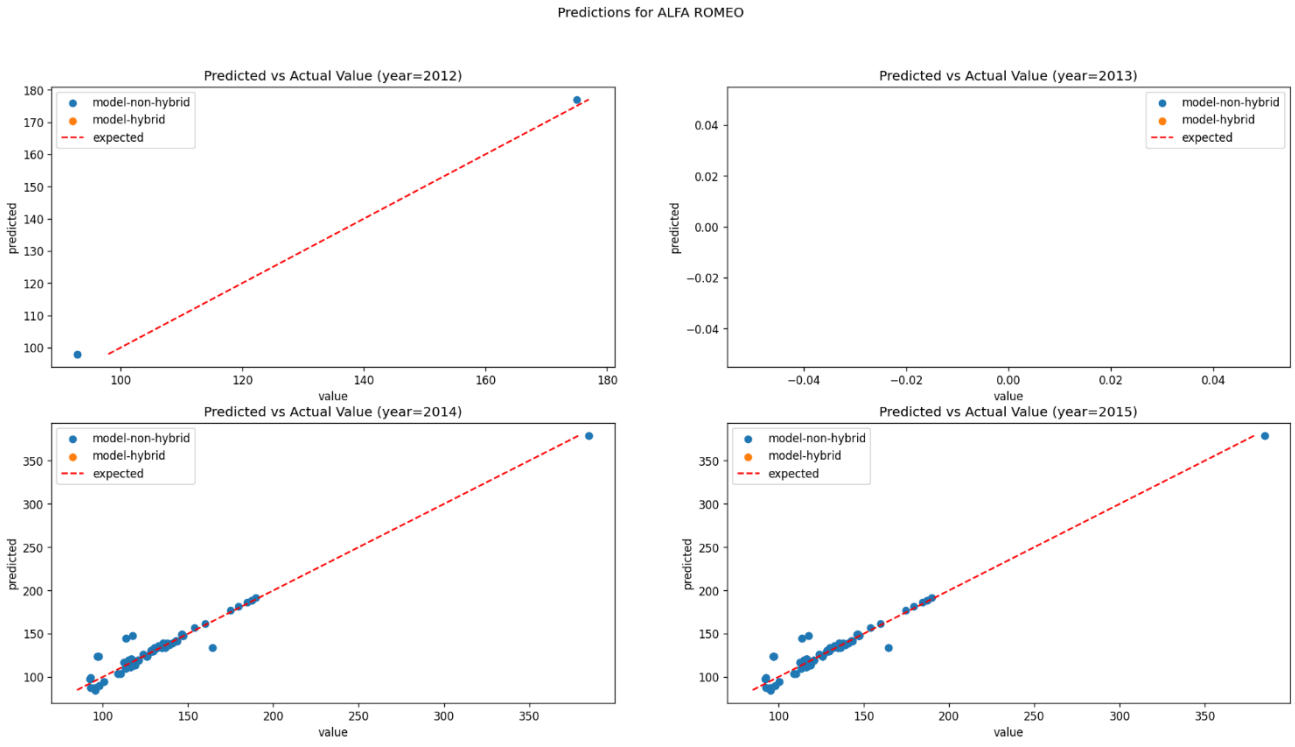




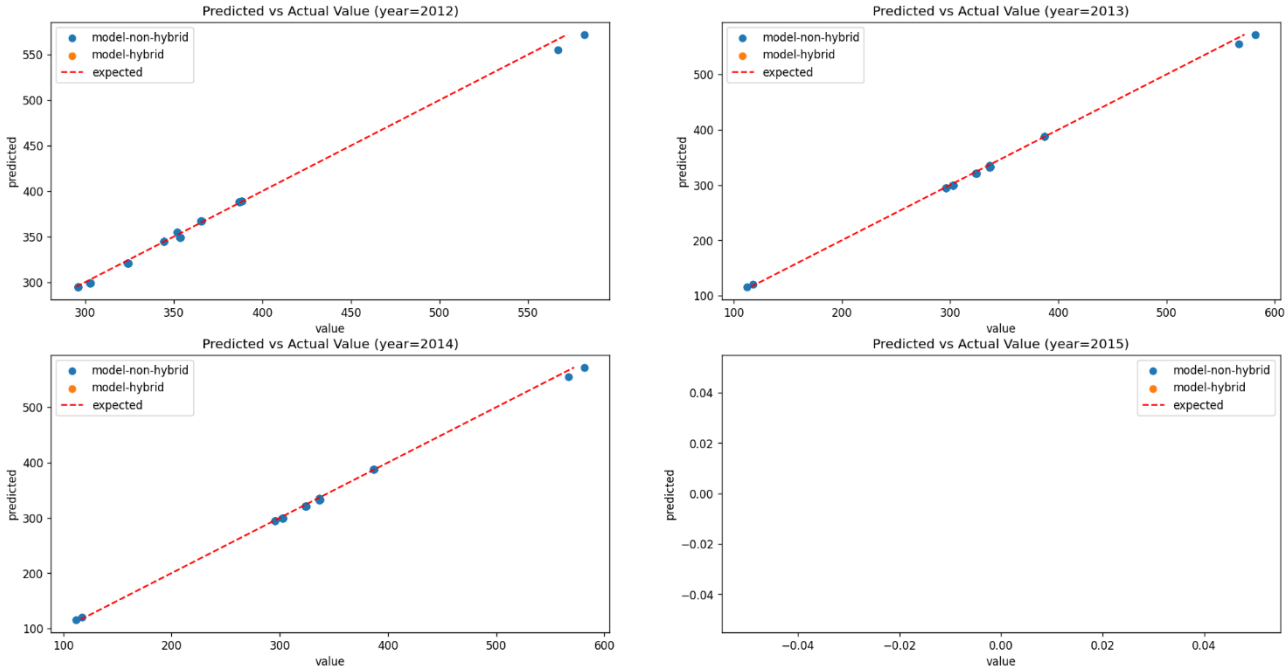




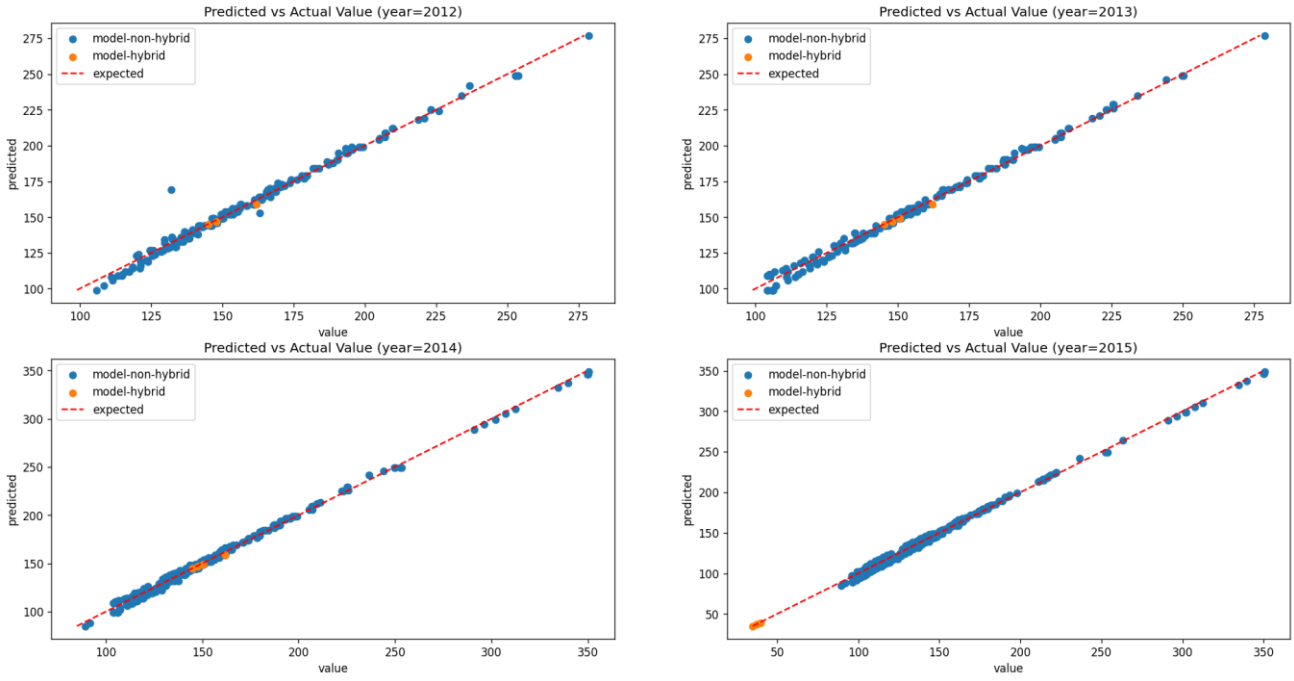
And per brand:



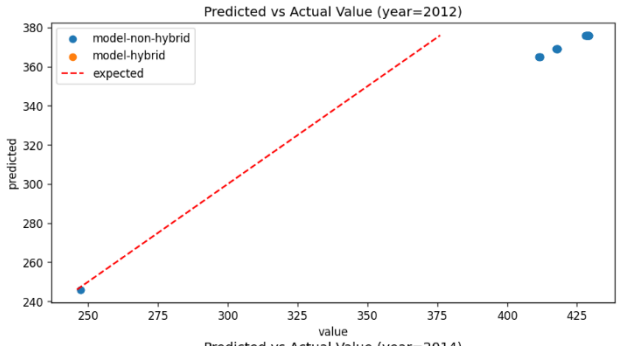
Predictions for ASTON MARTIN

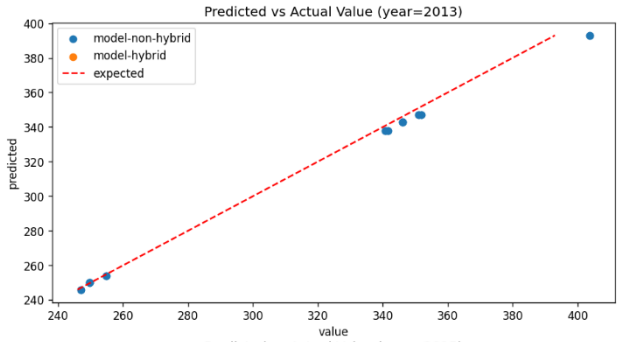


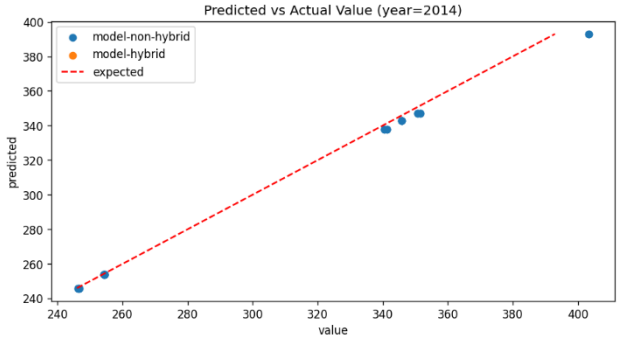
Predictions for AUDI

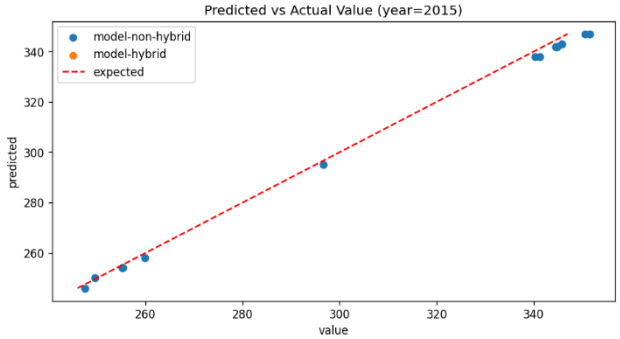


Predictions for BENTLEY

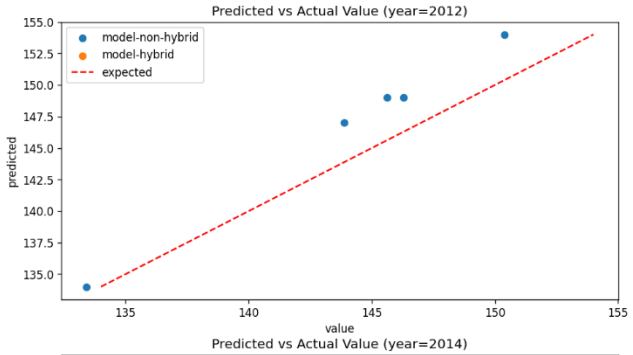


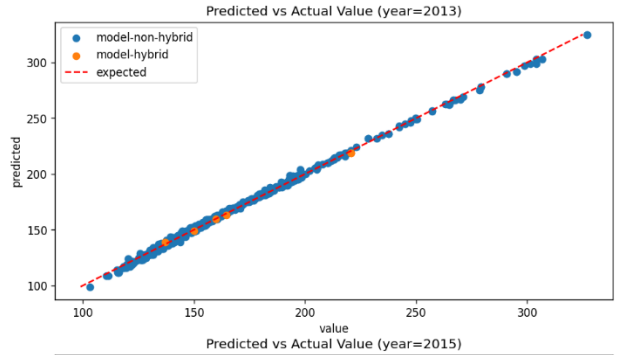


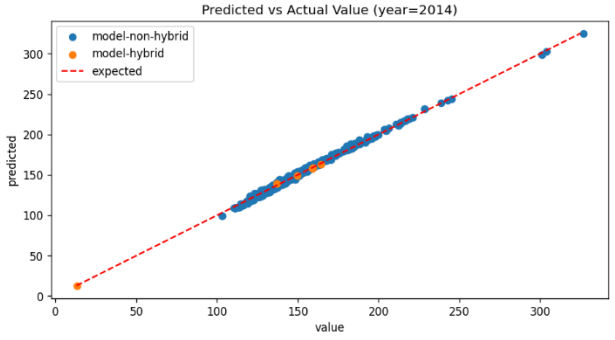


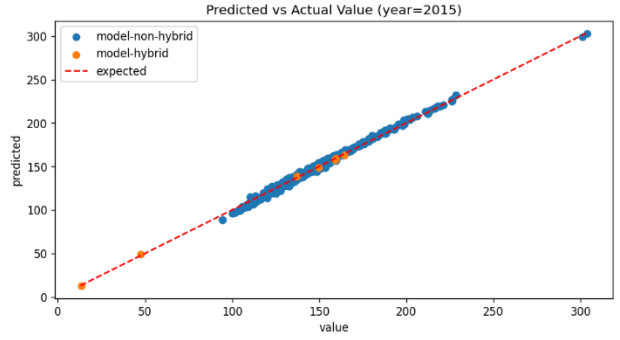


Predictions for BMW

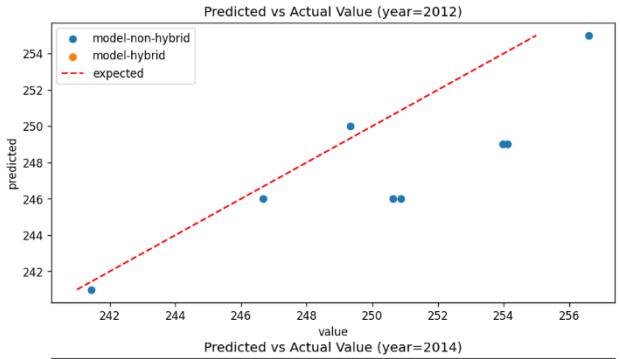


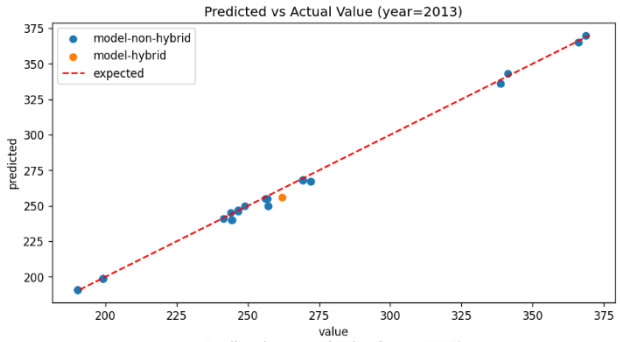


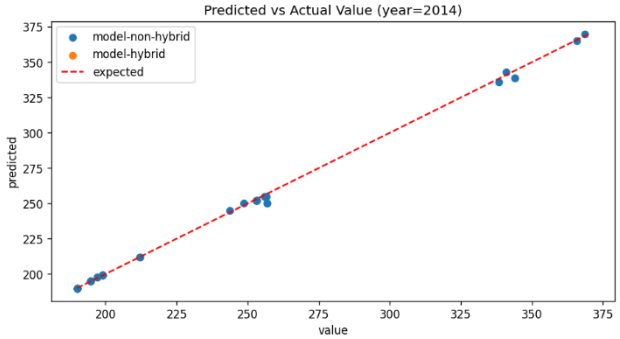


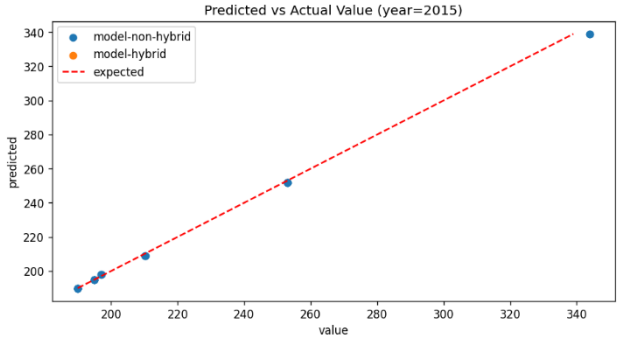


Predictions for CADILLAC

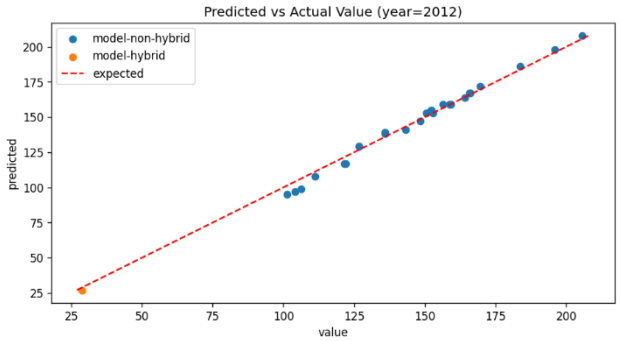


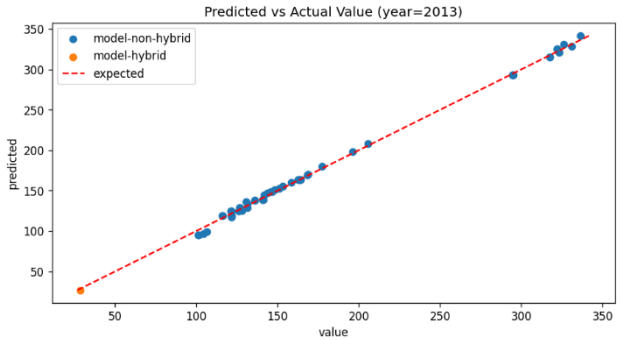


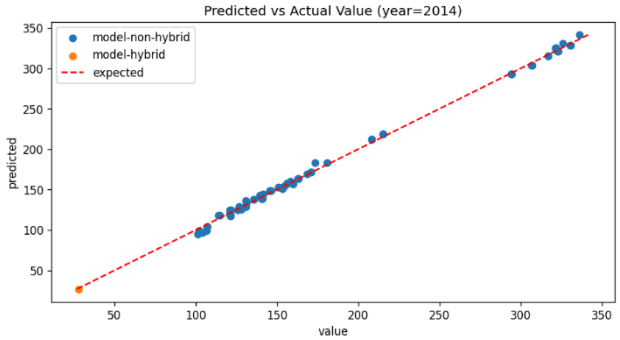


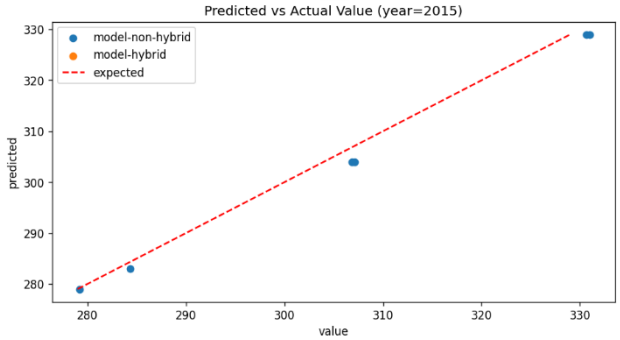


Predictions for CHEVROLET

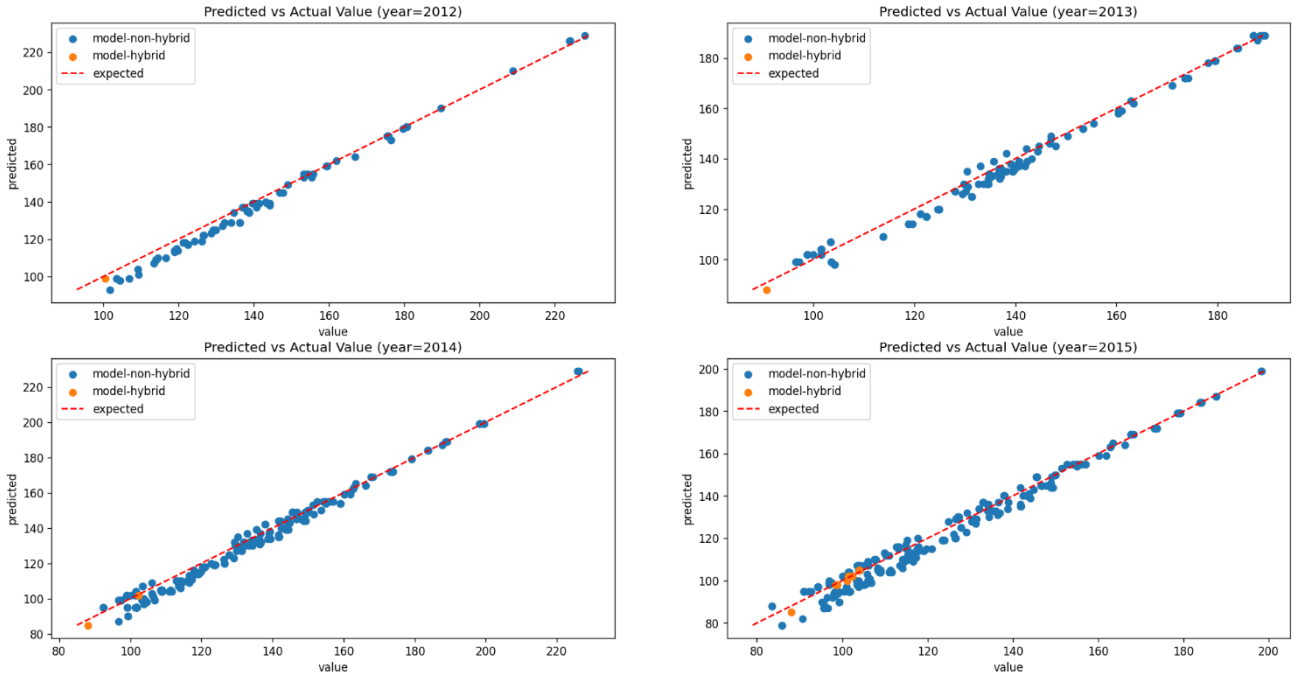




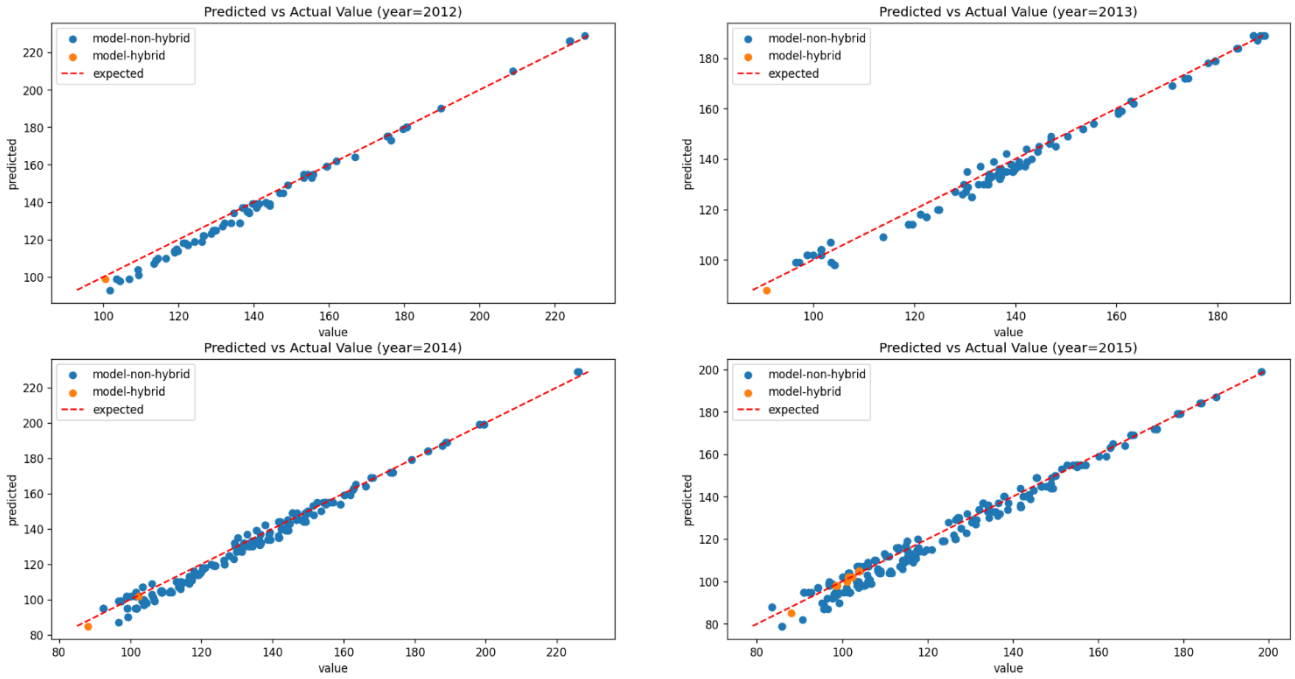




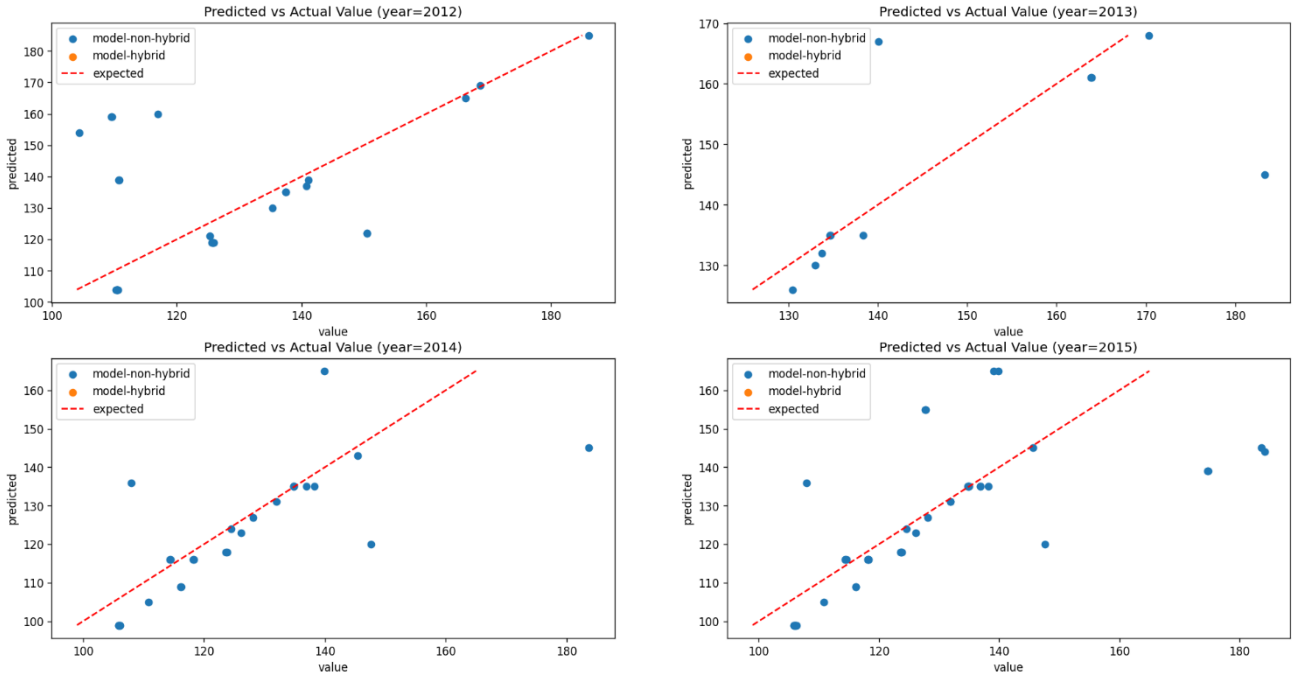
Predictions for CITROEN



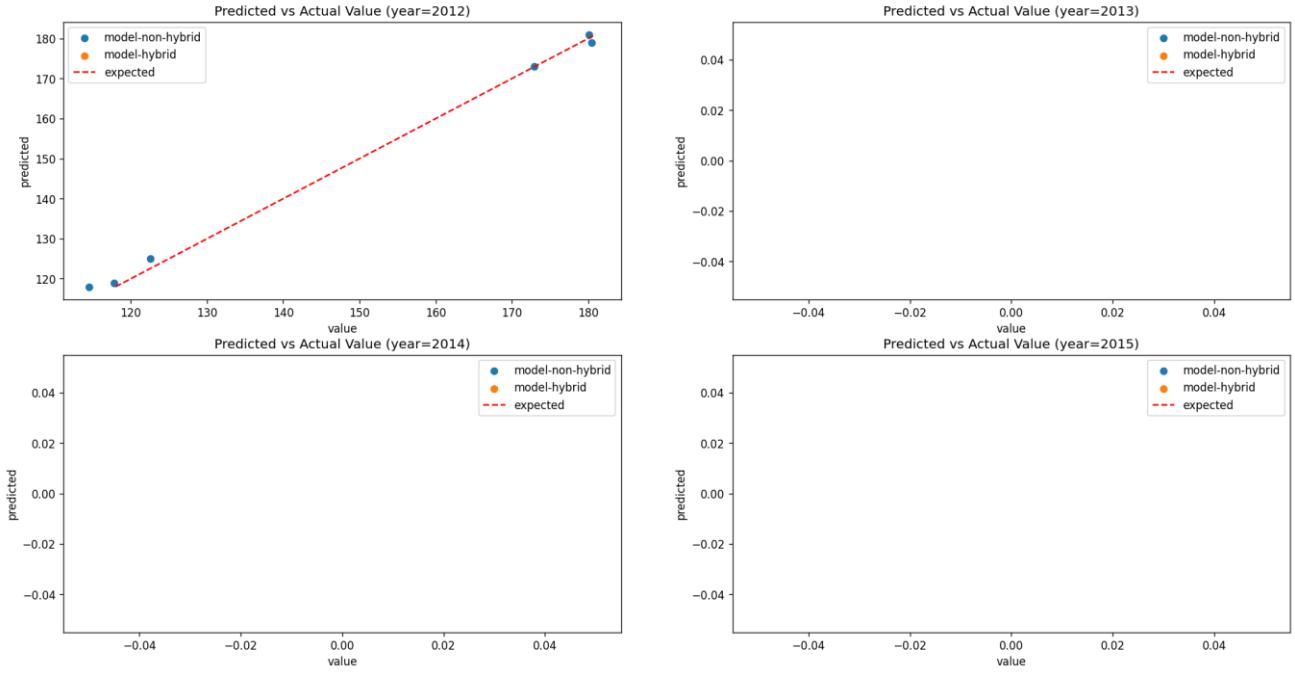
Predictions for CITROEN



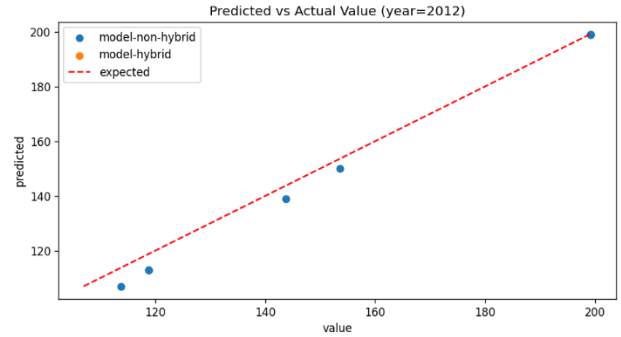
Predictions for DACIA

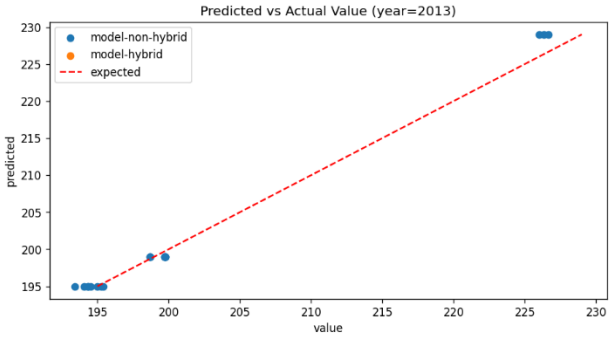


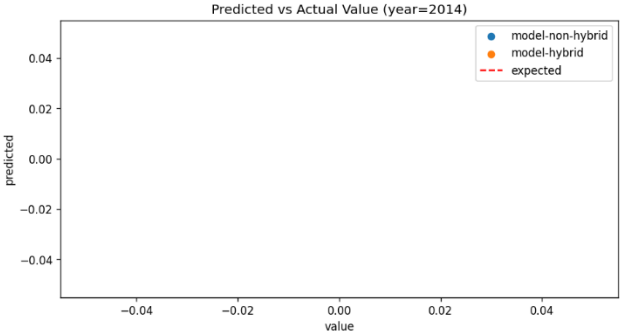
Predictions for DAIHATSU

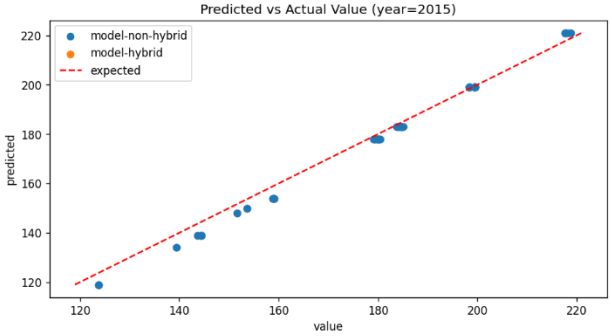


Predictions for DANGEL

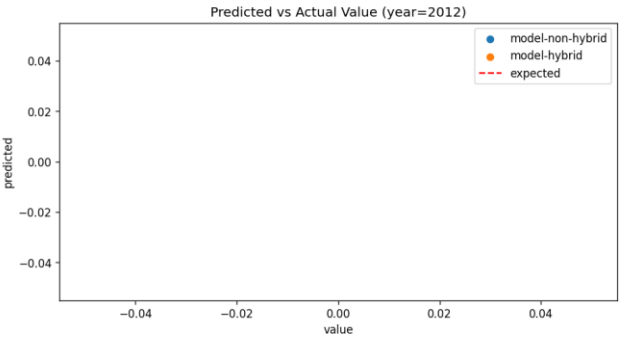


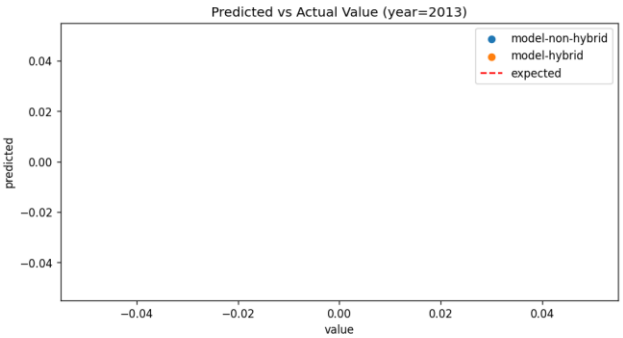


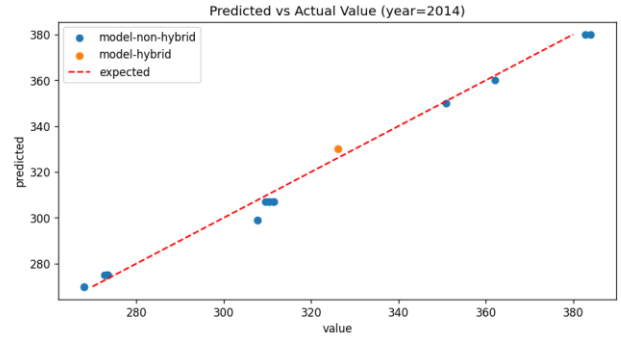


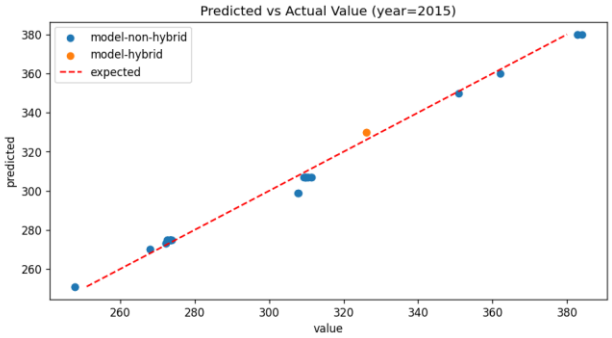


Predictions for FERRARI

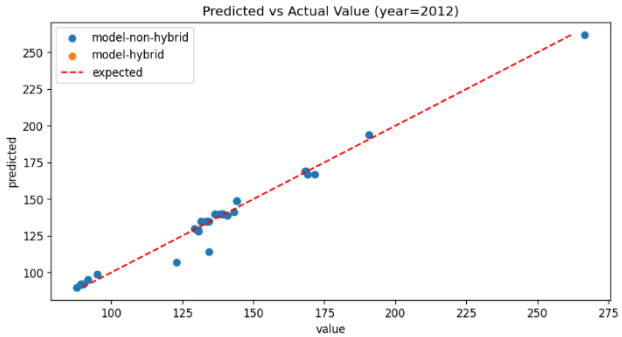


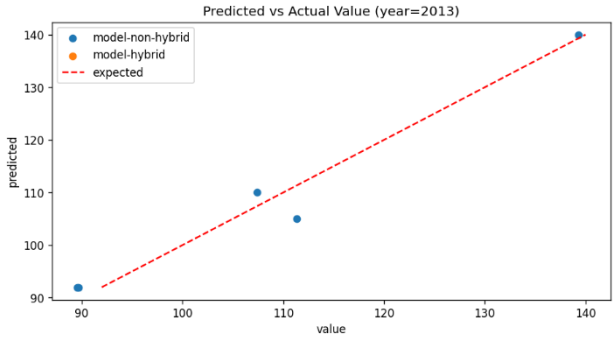


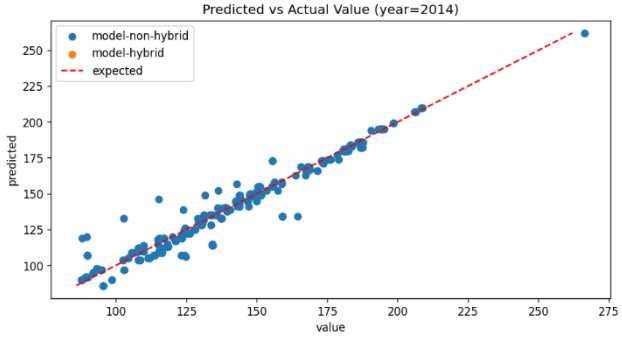


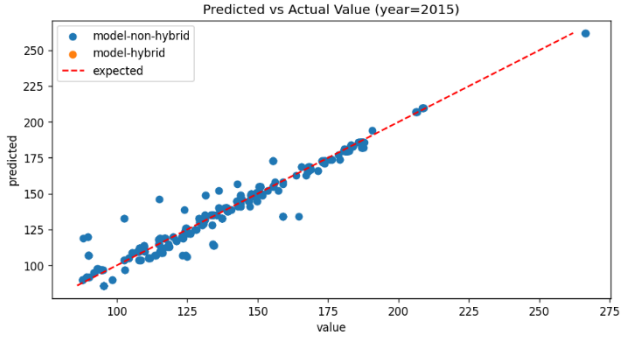


Predictions for FIAT

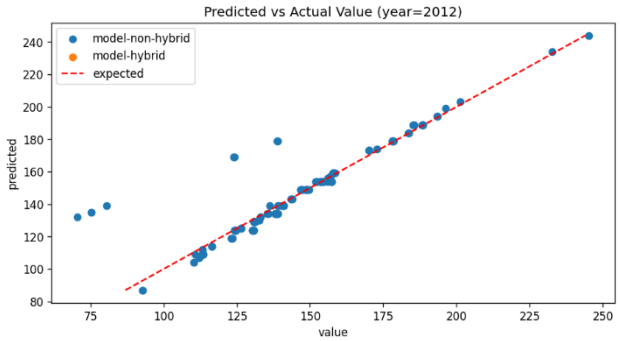


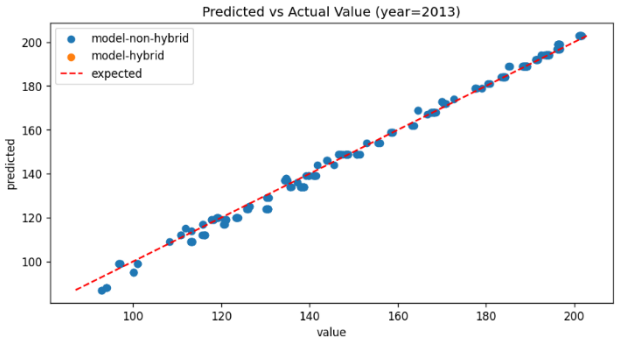


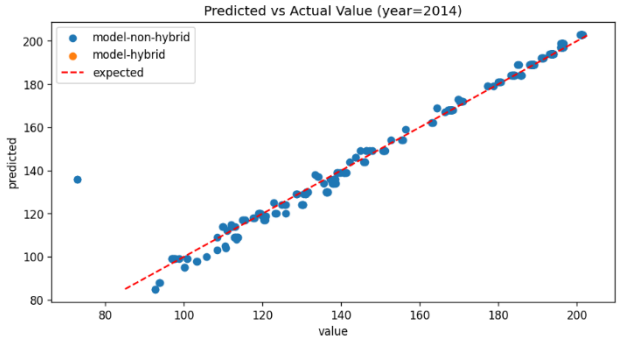


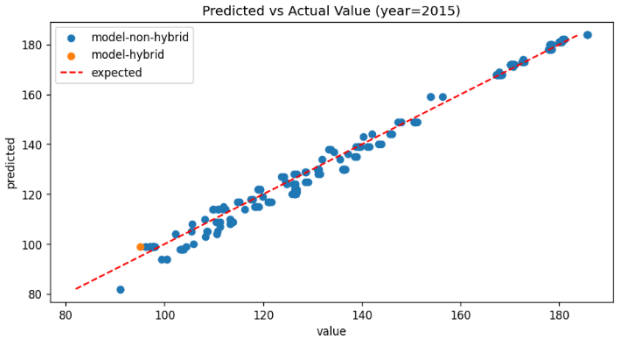


Predictions for FORD

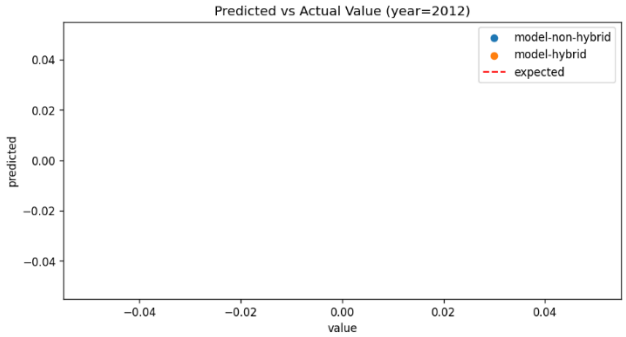


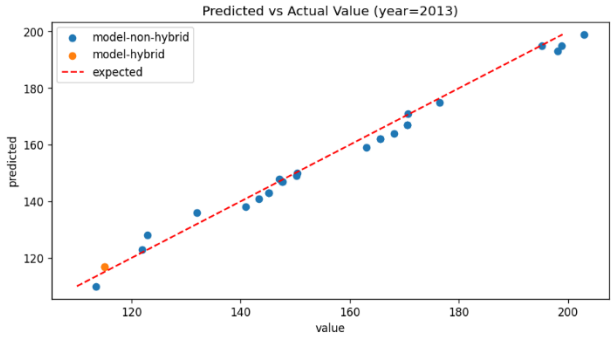


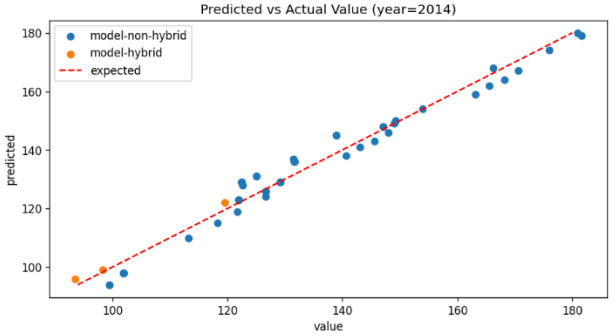


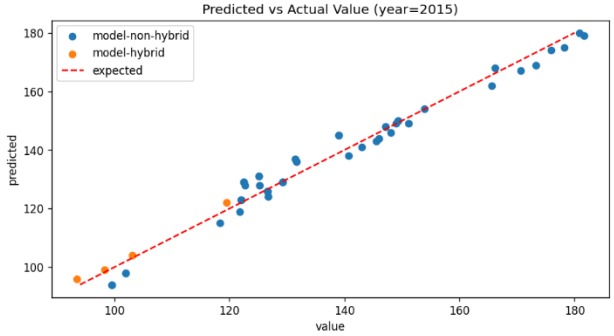


Predictions for HONDA

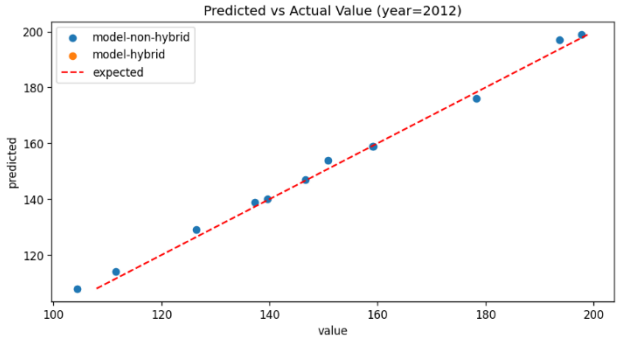


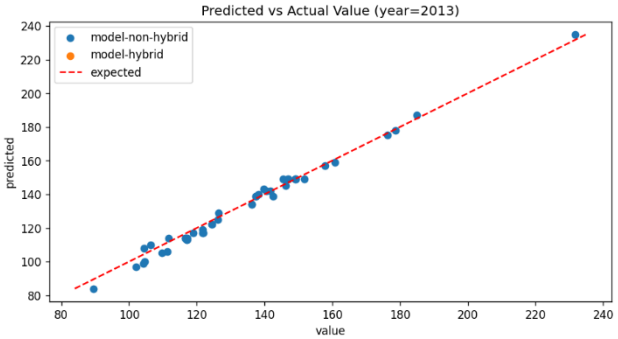


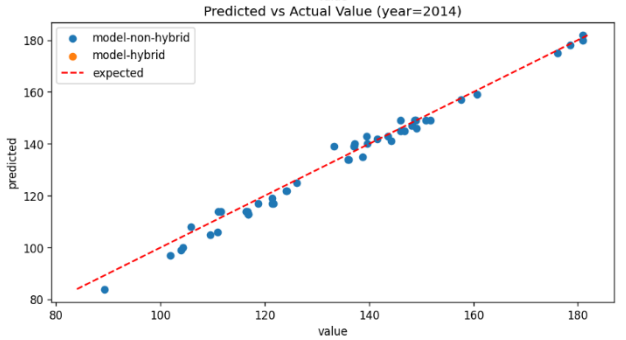


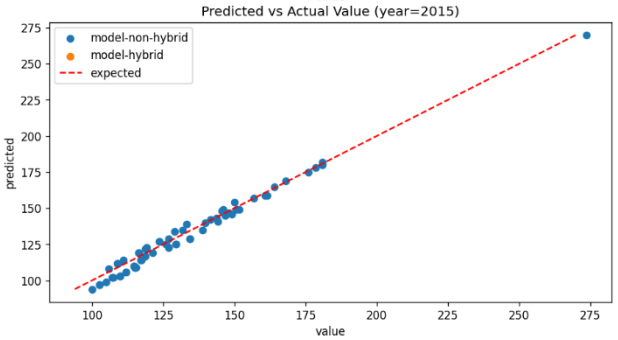


Predictions for HYUNDAI

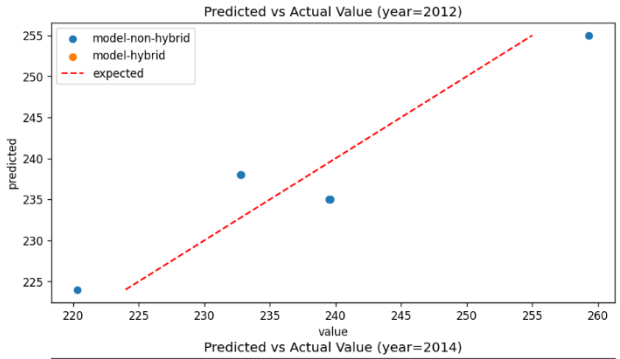


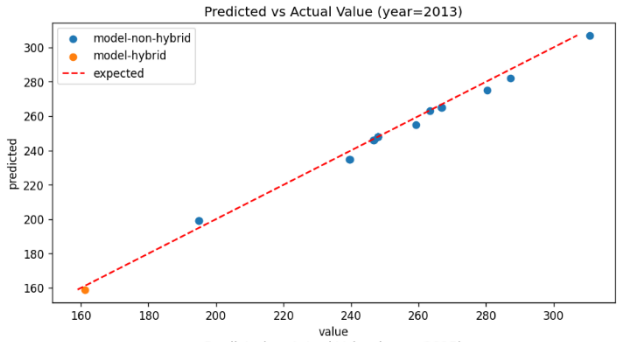


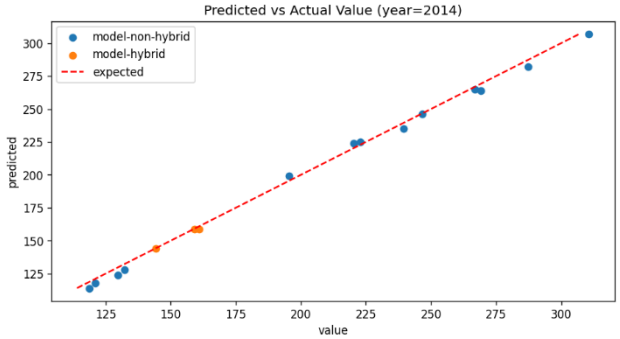


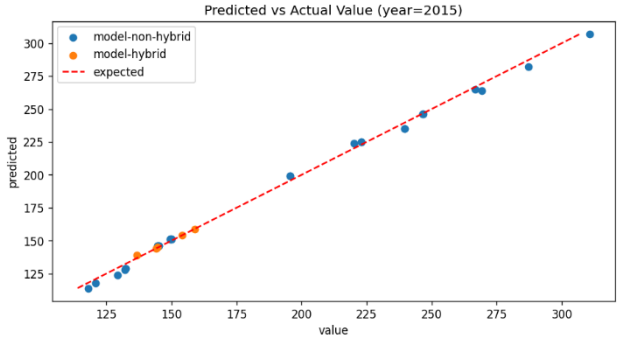


Predictions for INFINITI

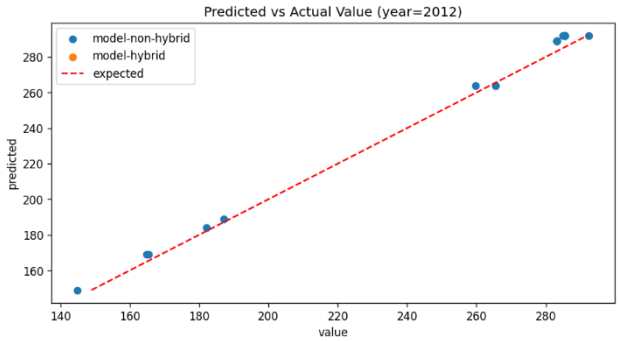


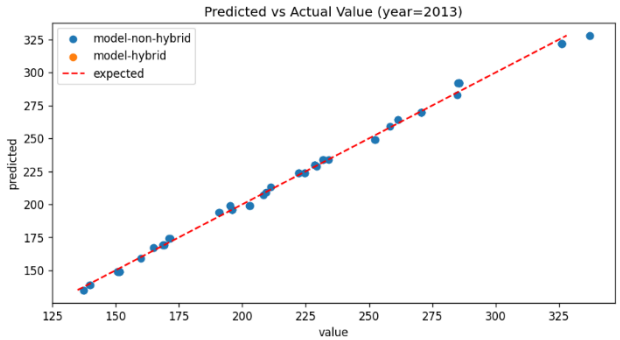


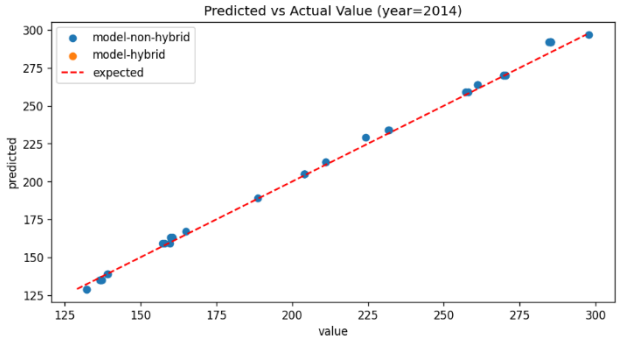


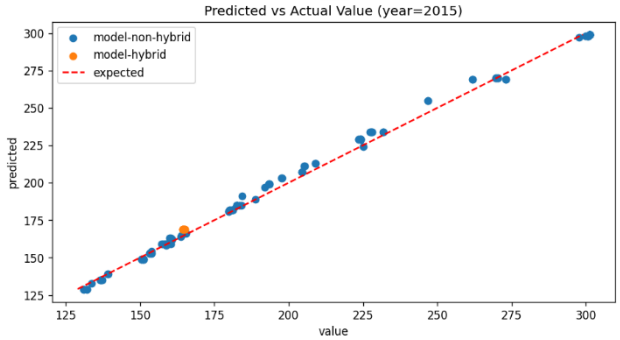


Predictions for JAGUAR

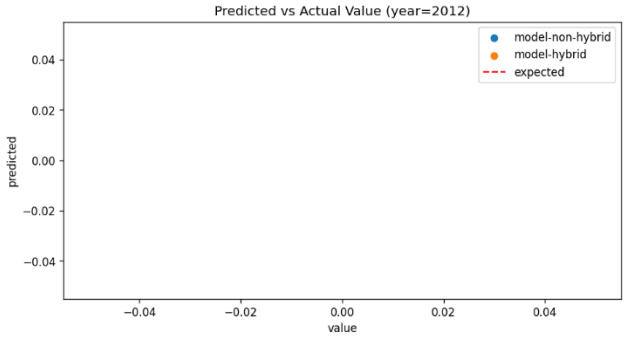


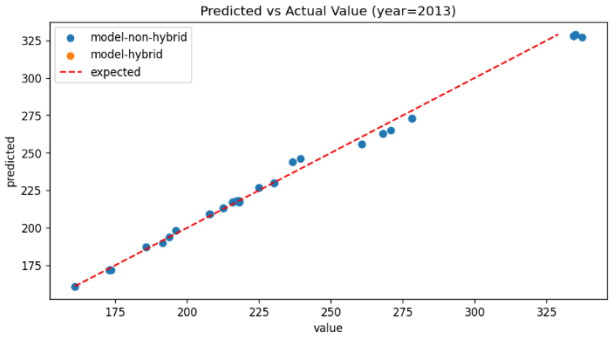


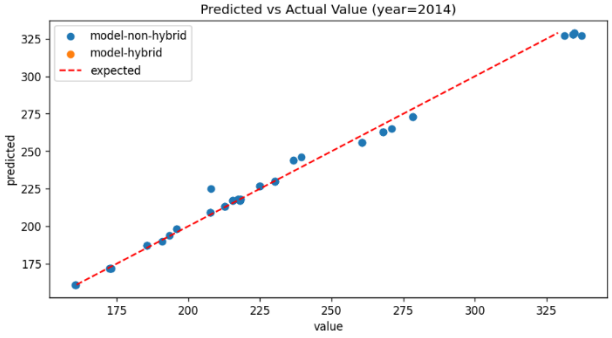


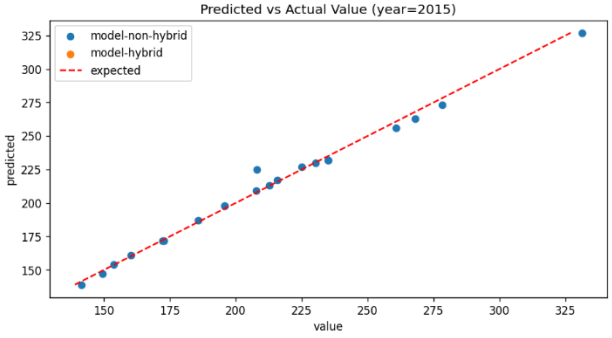


Predictions for JEEP

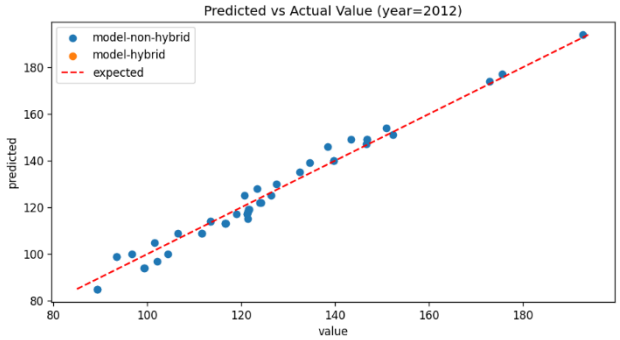


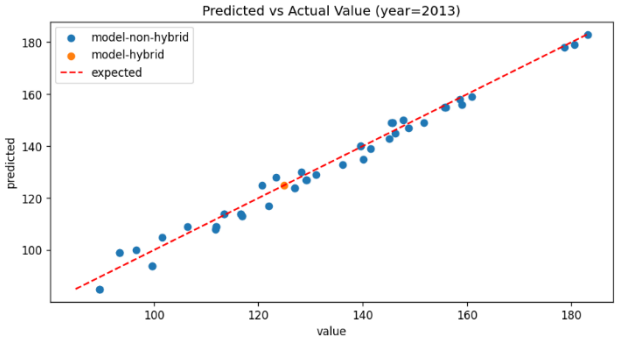


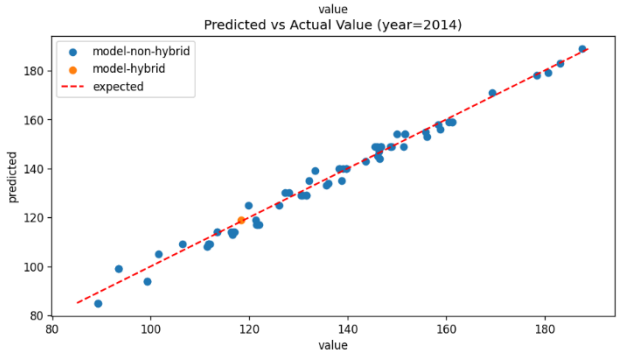


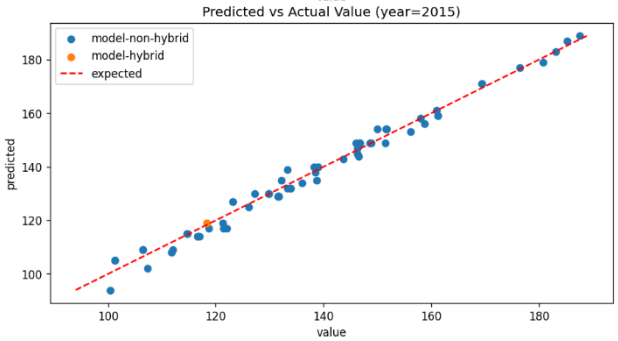


Predictions for KIA

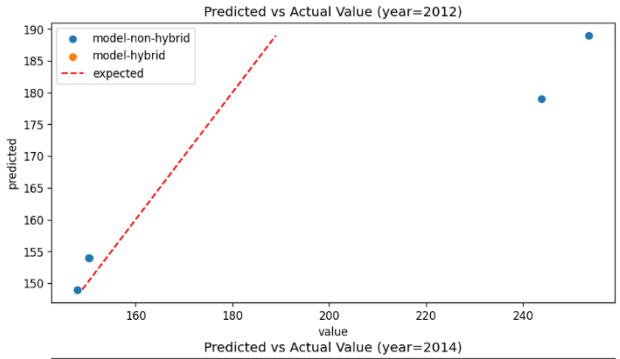


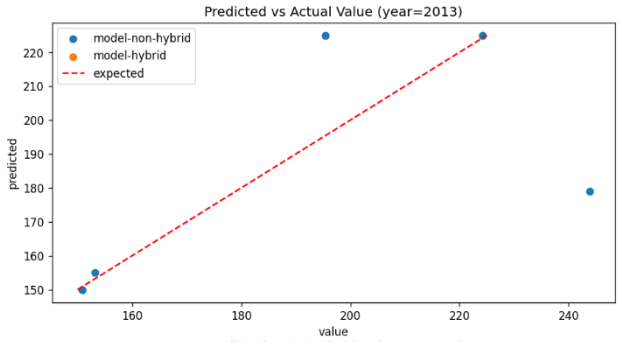


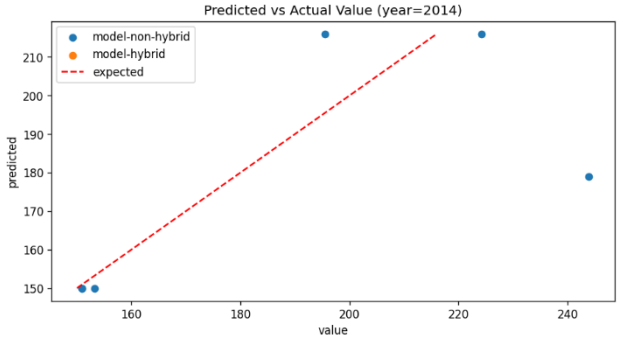


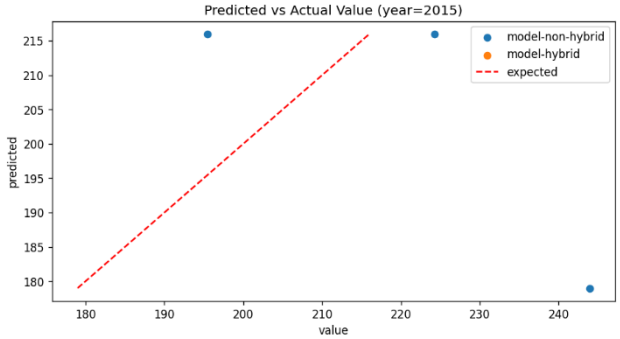


Predictions for LADA

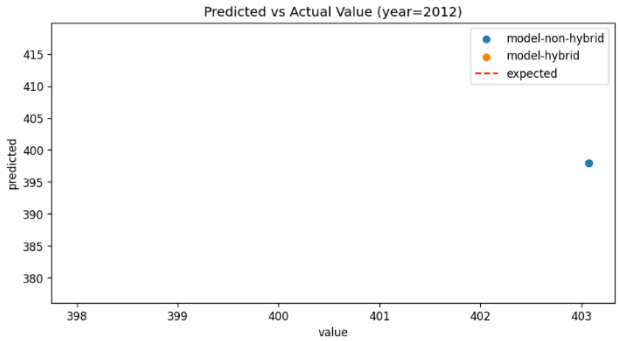


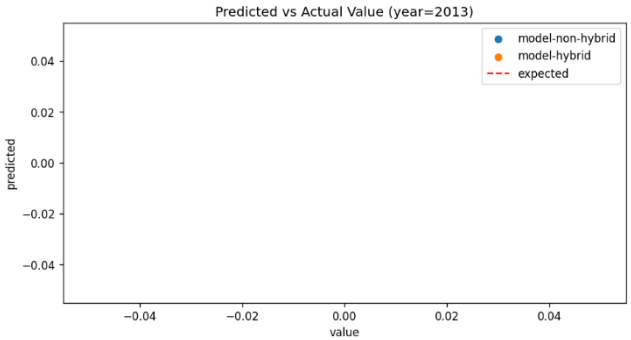


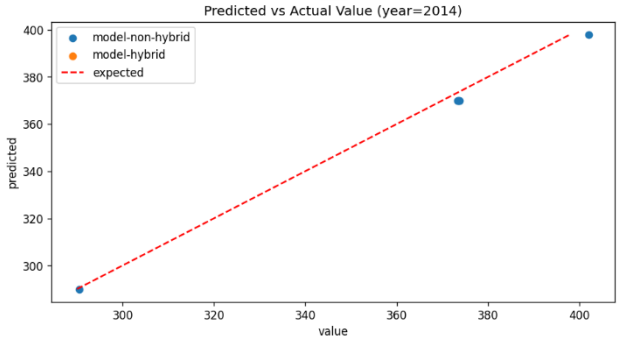


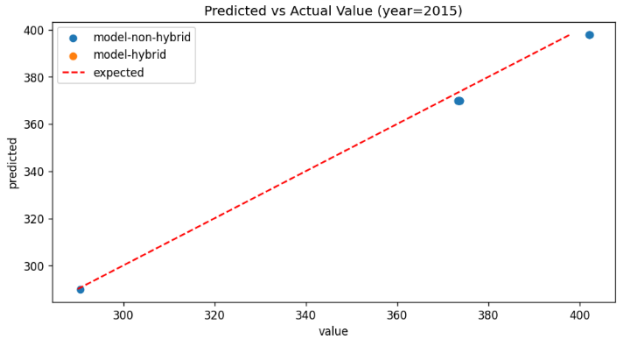


Predictions for LAMBORGHINI

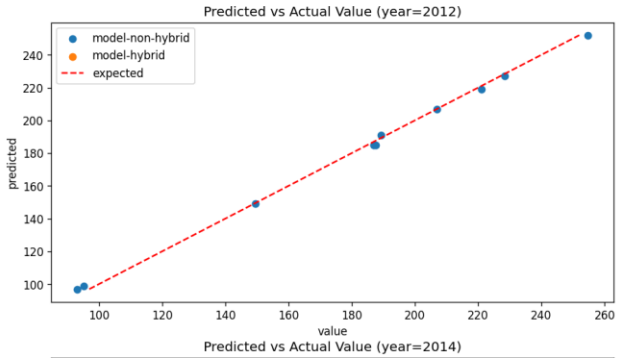


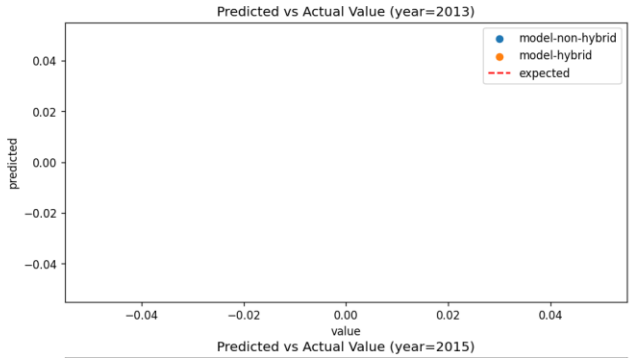


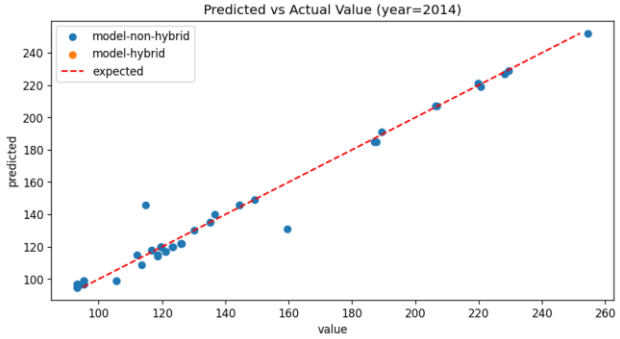


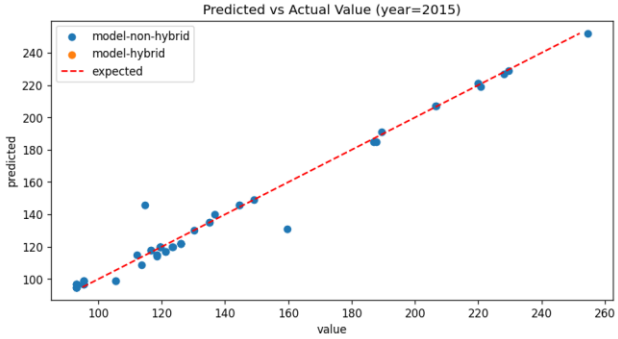


Predictions for LANCIA

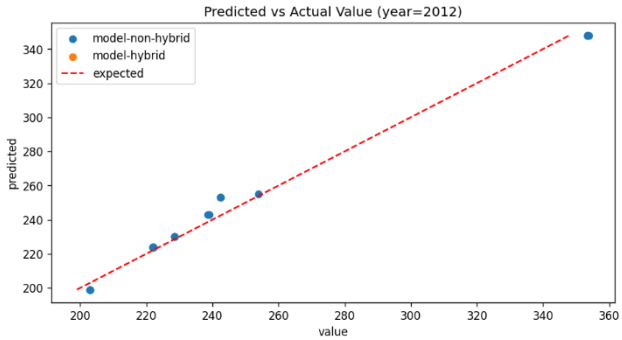


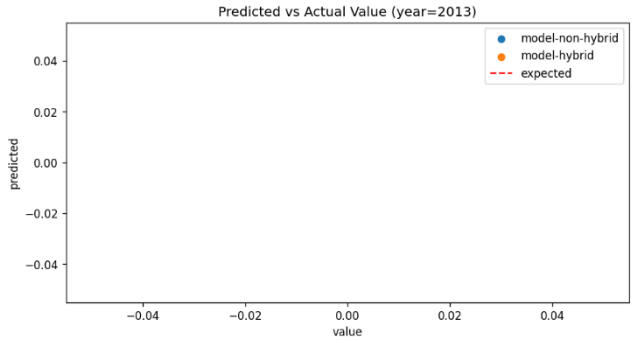


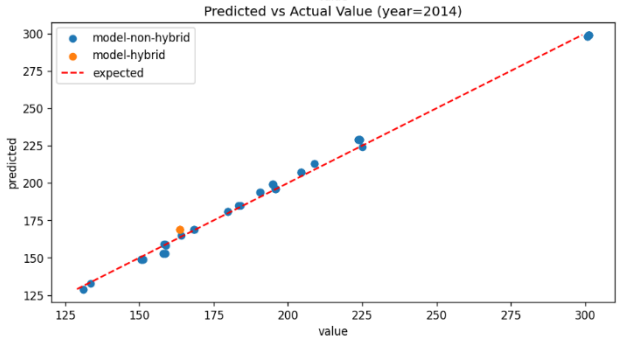


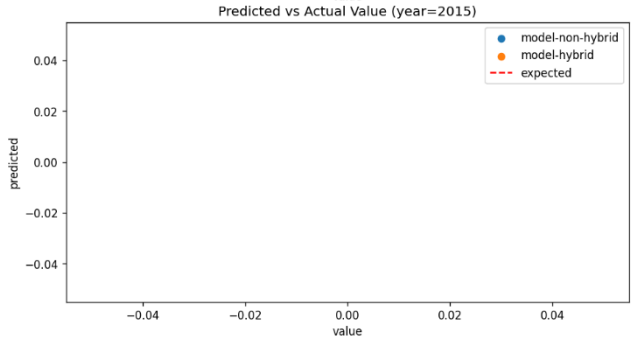


Predictions for LAND ROVER

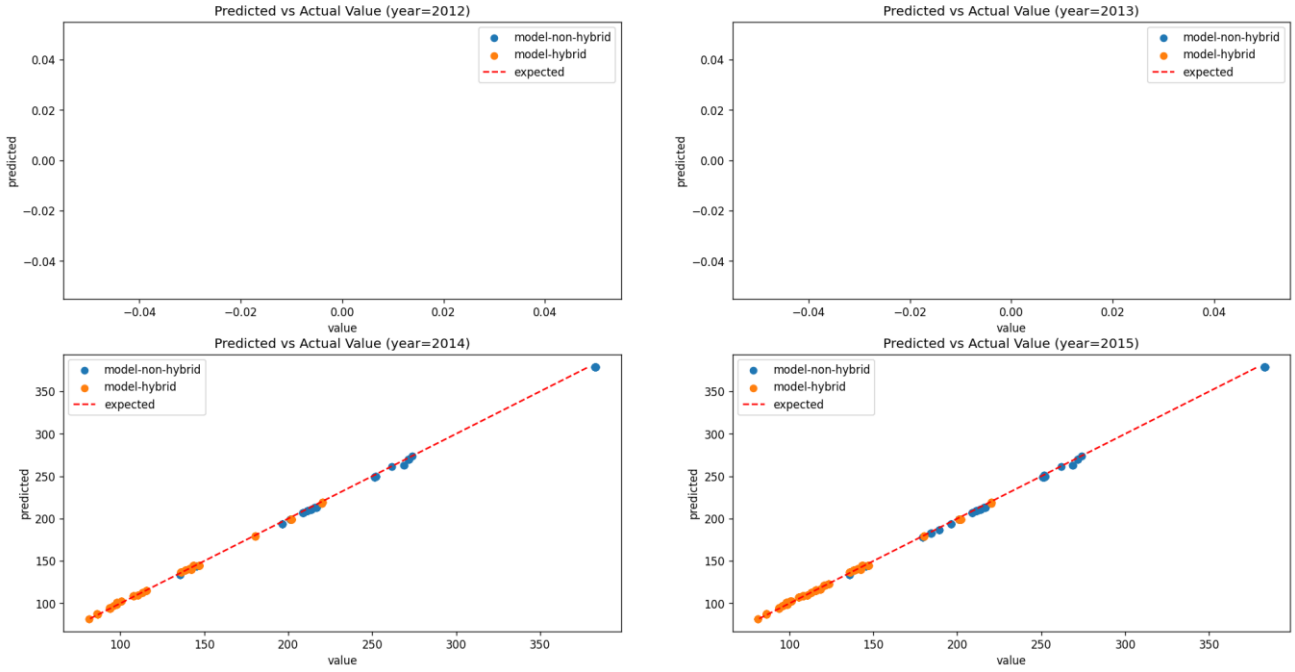




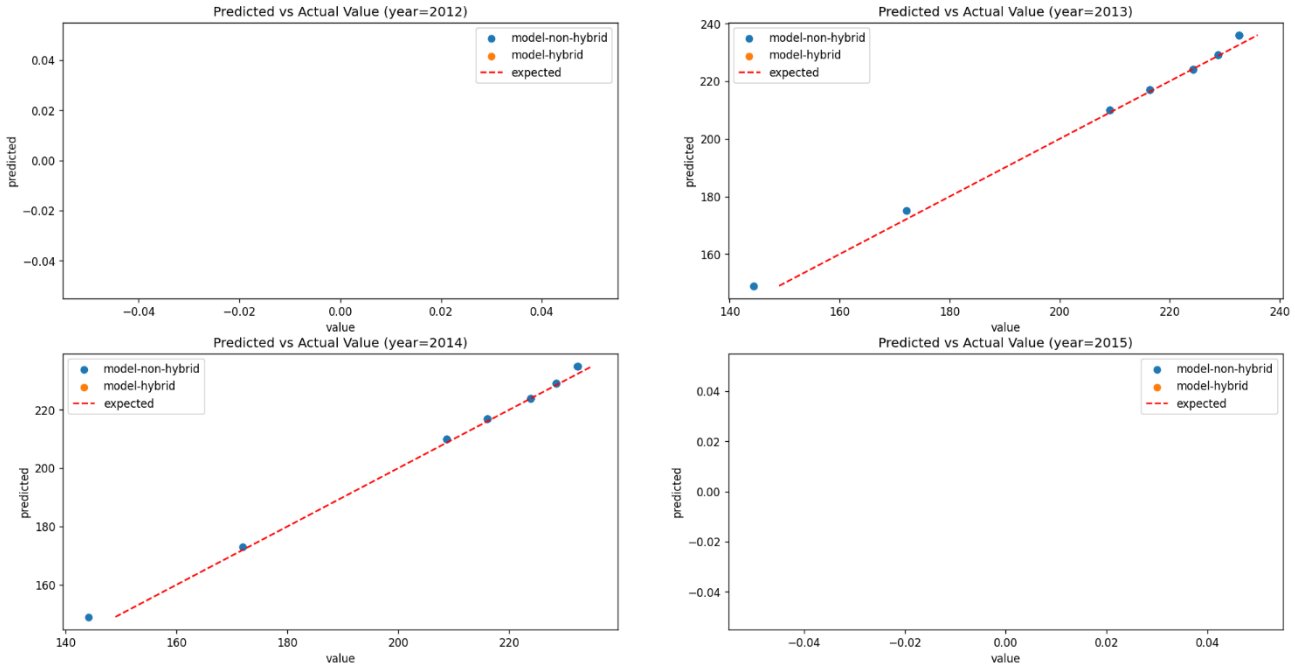




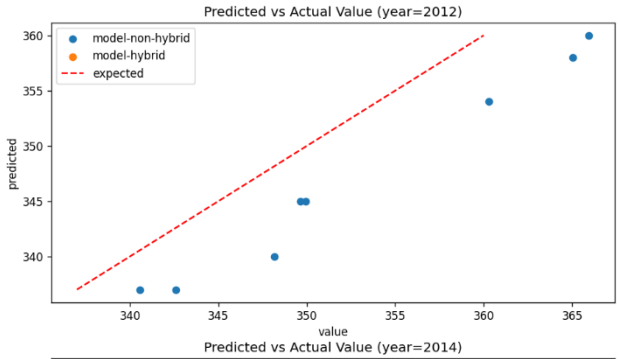
Predictions for LEXUS

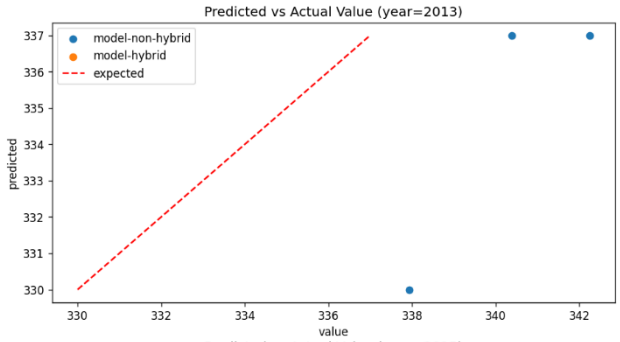


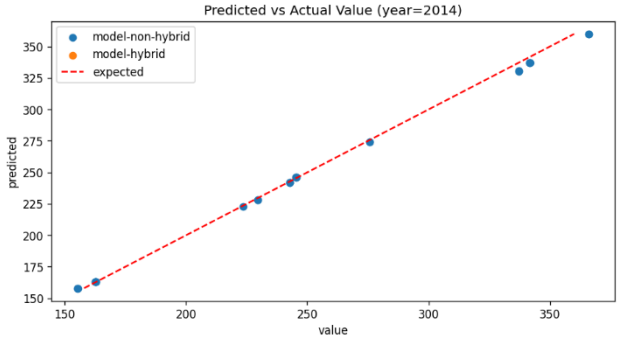
Predictions for LOTUS

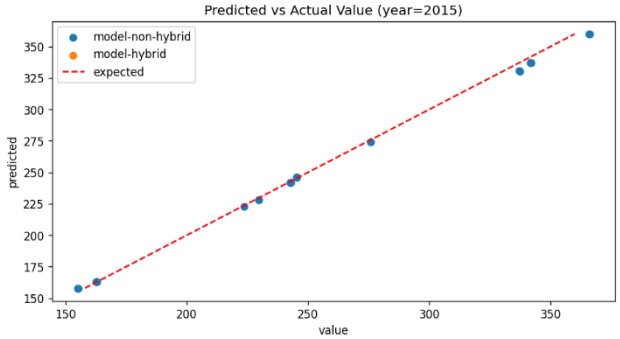


Predictions for MASERATI

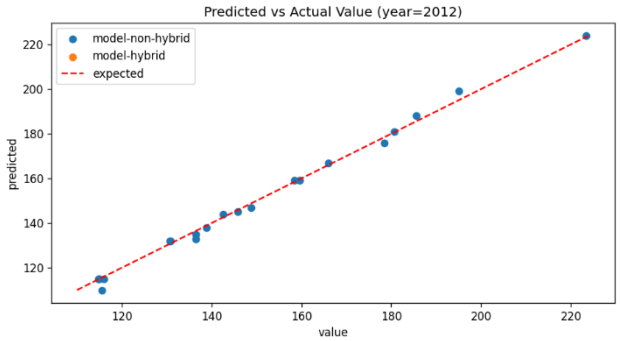


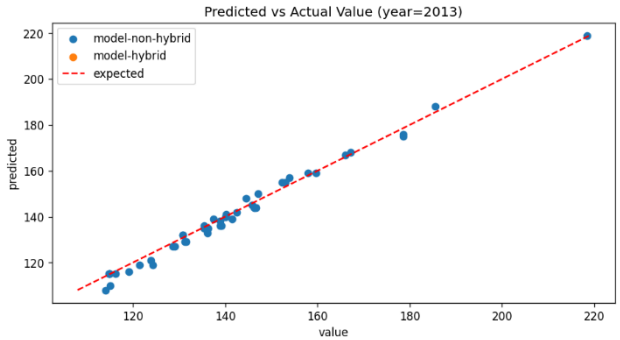


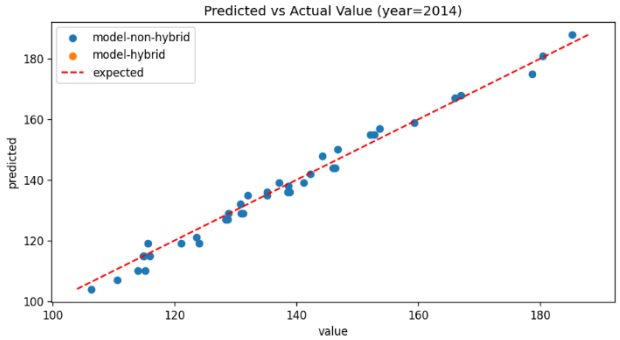


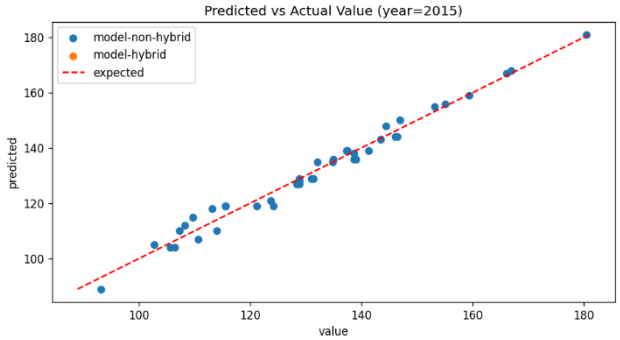


Predictions for MAZDA

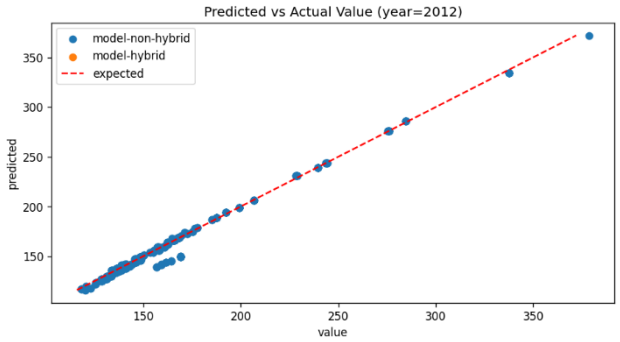


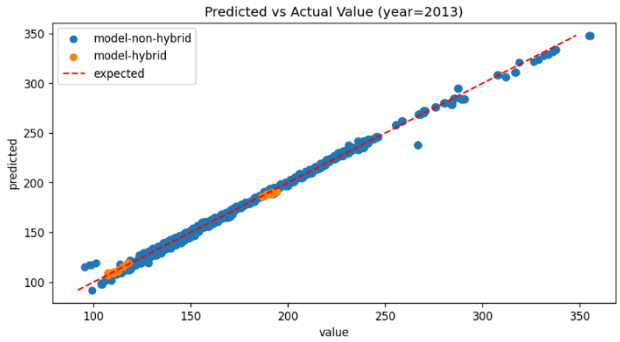


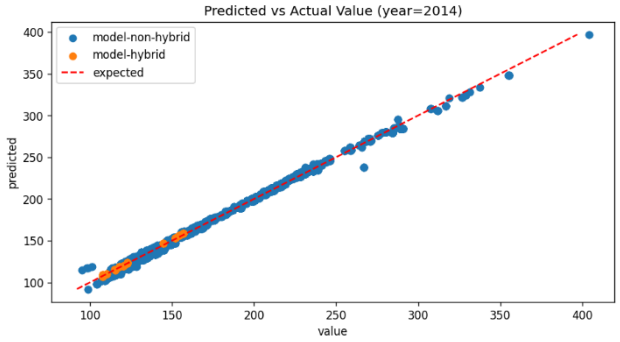


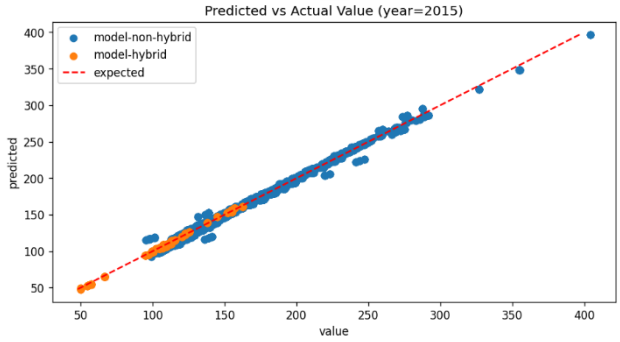


Predictions for MERCEDES

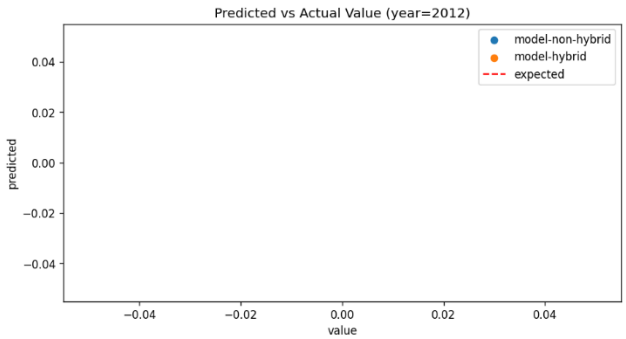


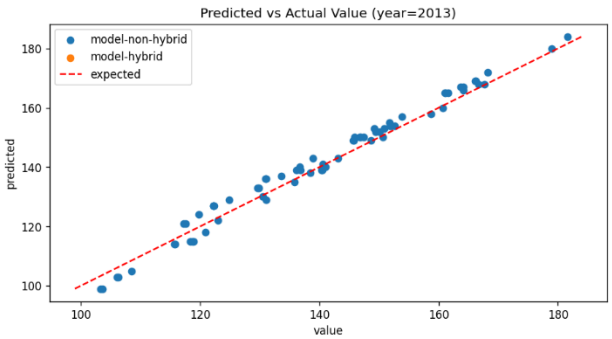


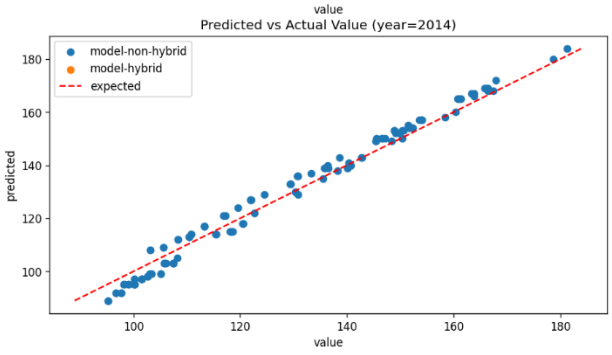


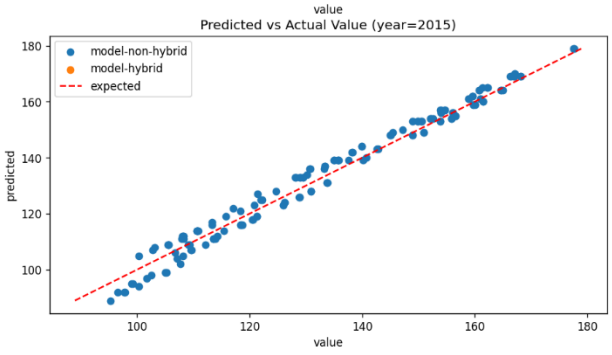


Predictions for MINI

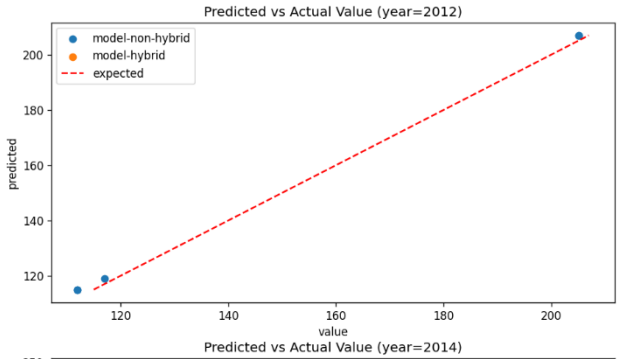


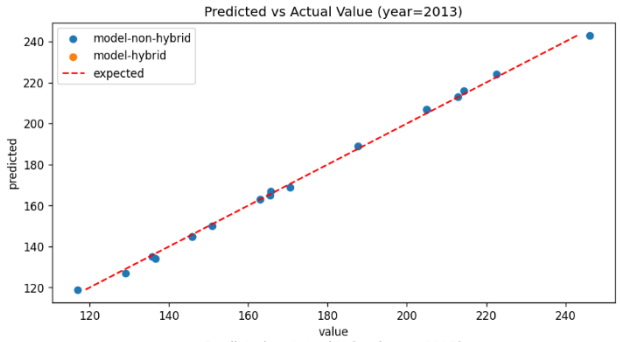


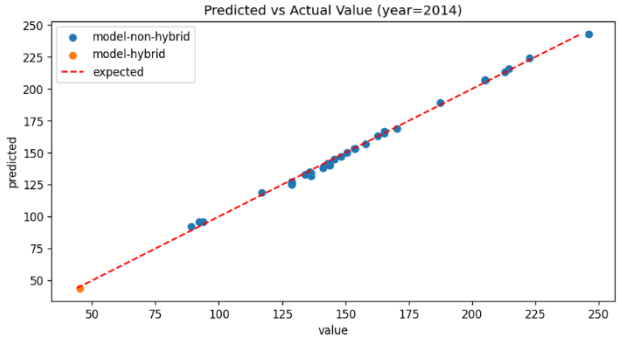


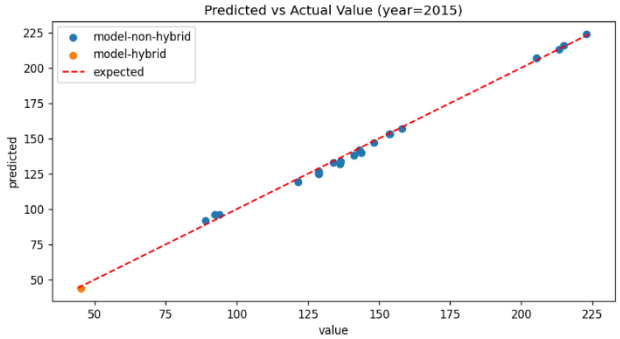


Predictions for MITSUBISHI

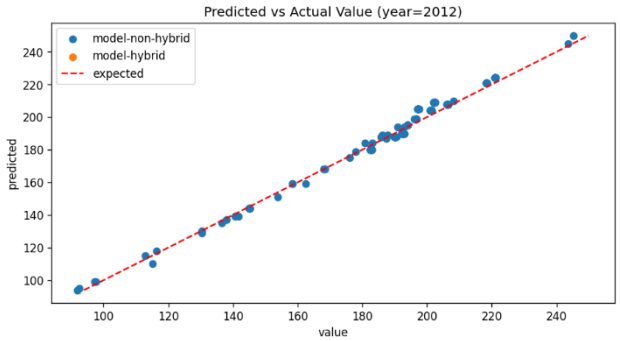


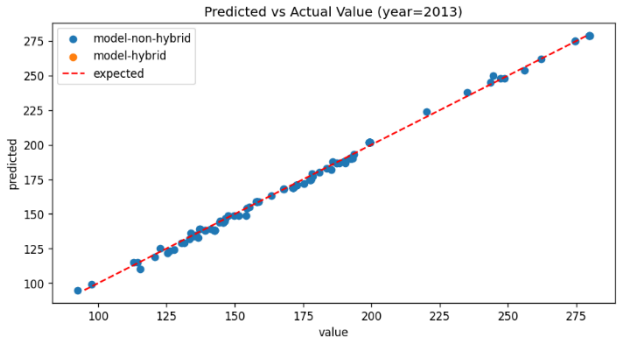


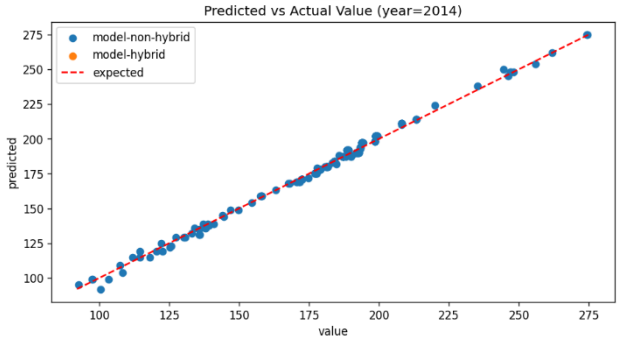


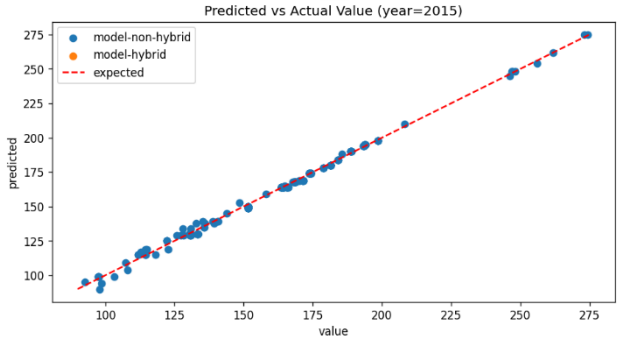


Predictions for NISSAN

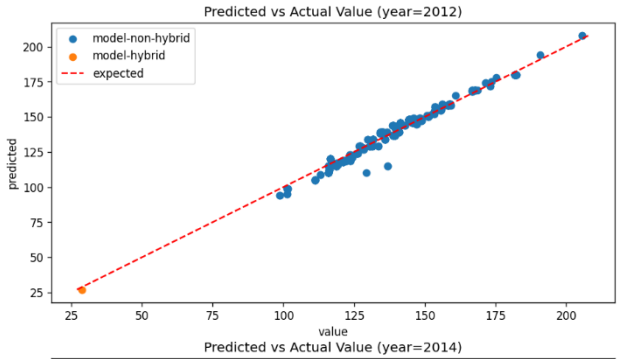


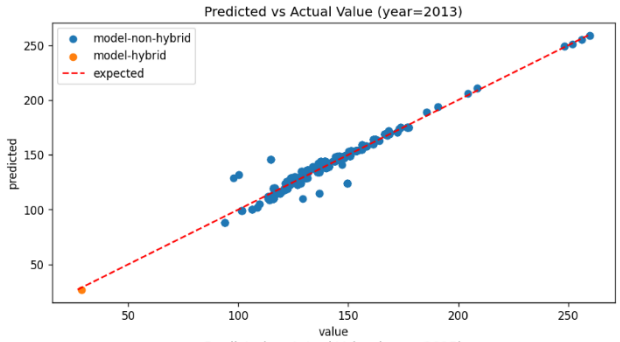


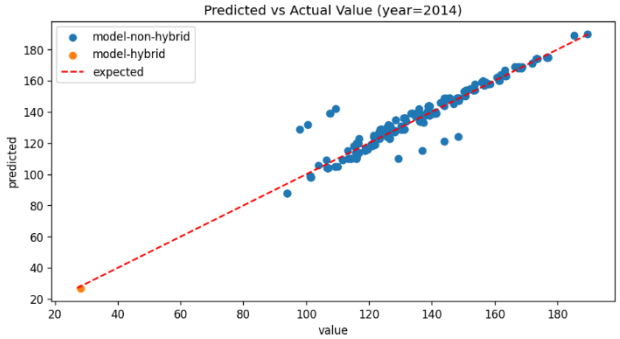


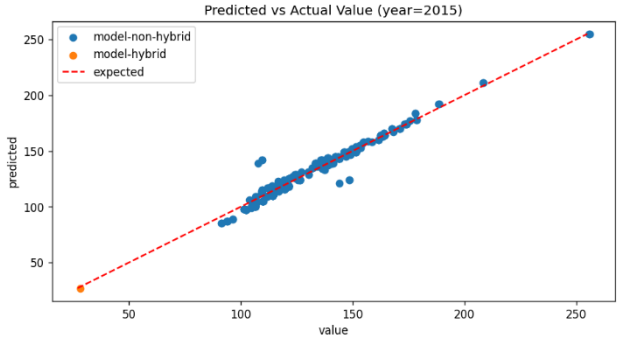


Predictions for OPEL

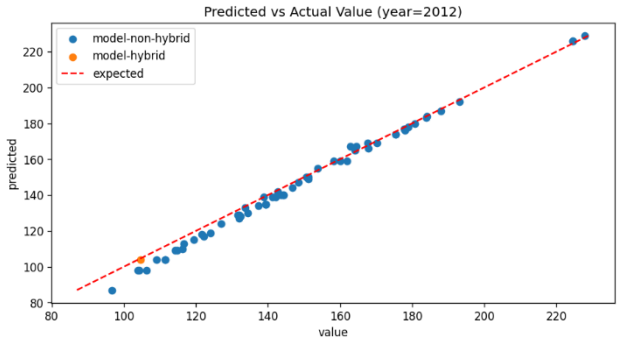


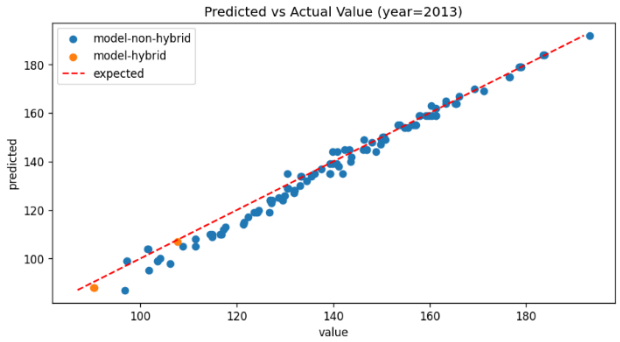


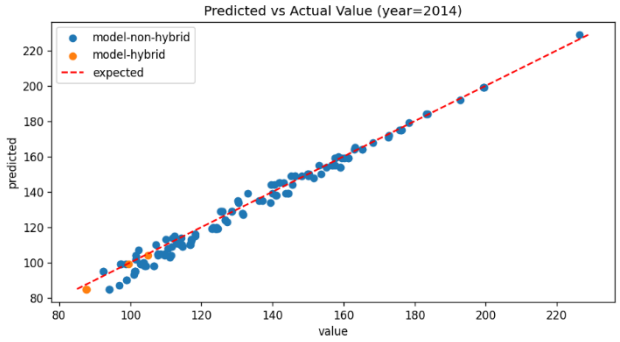


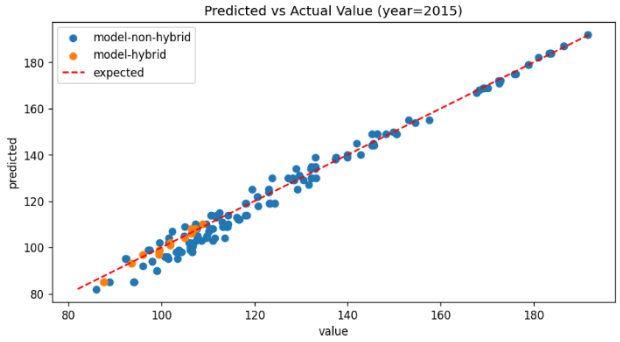


Predictions for PEUGEOT

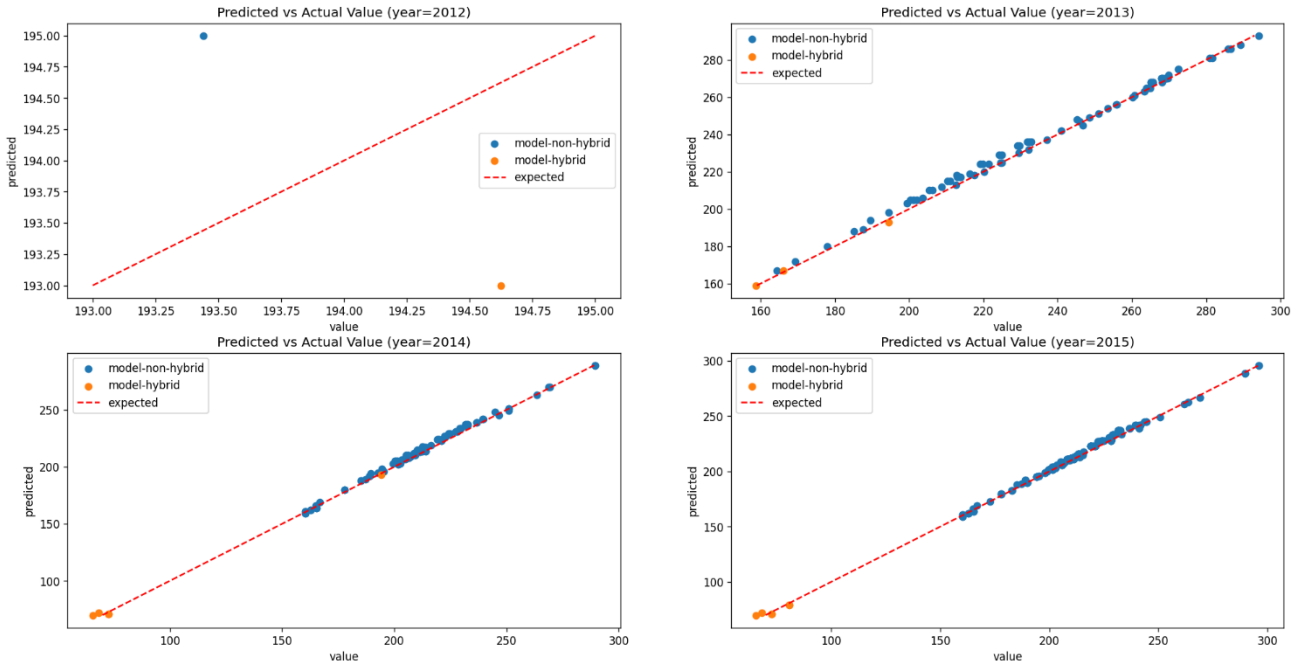




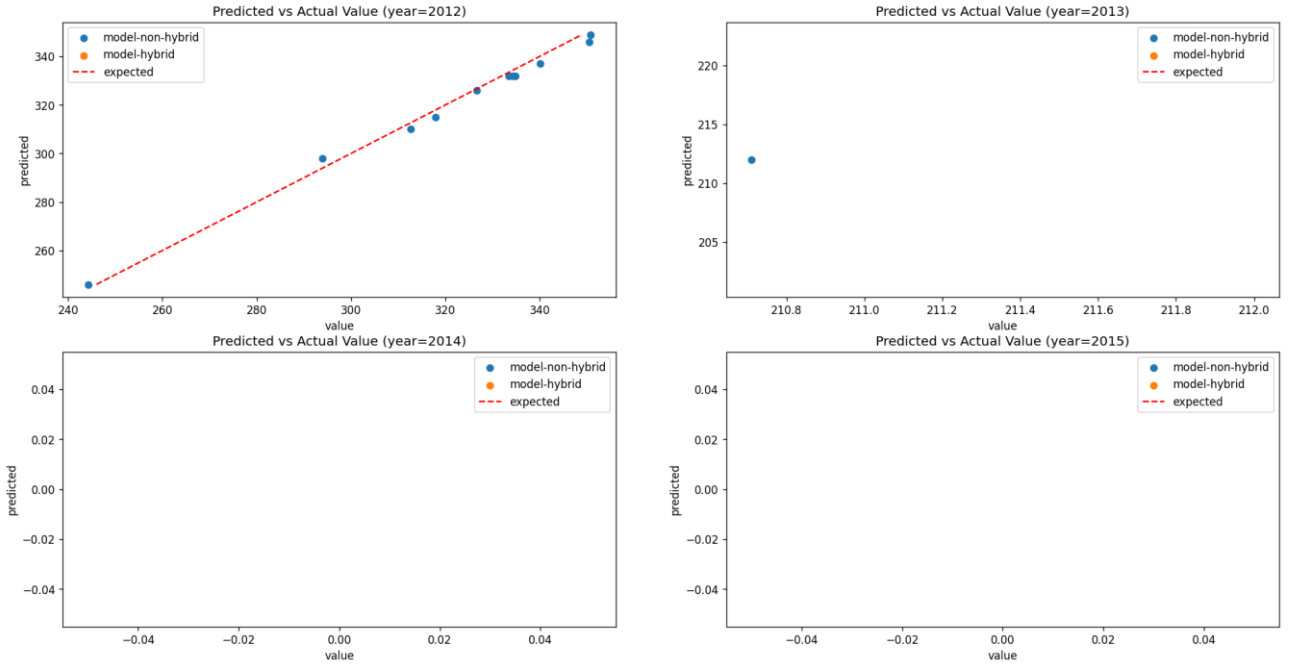




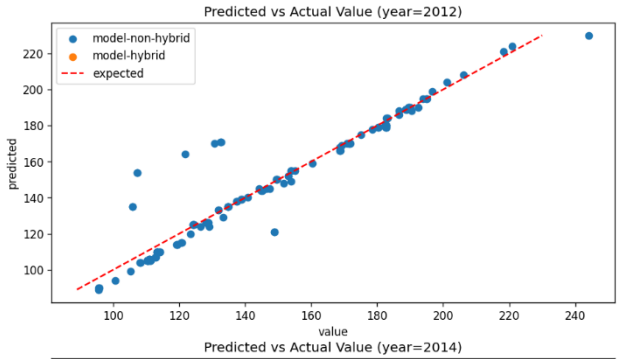
Predictions for PORSCHE

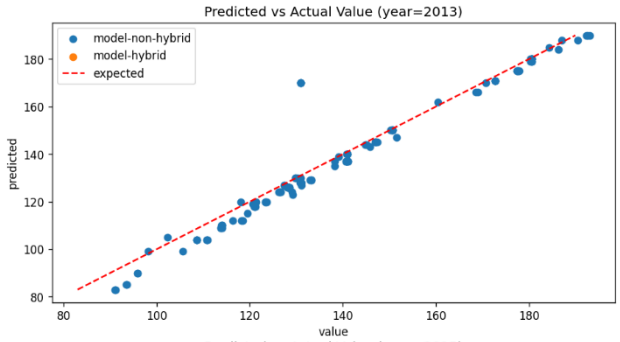


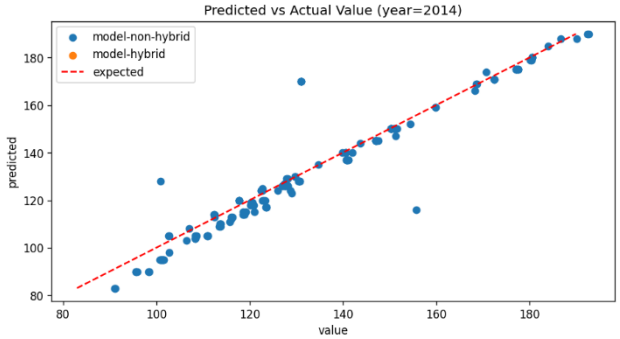
Predictions for QUATTRO

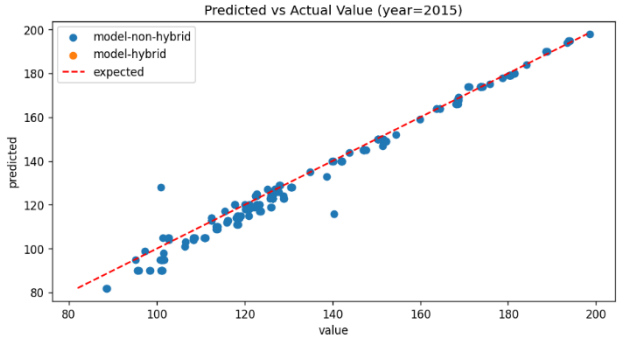


Predictions for RENAULT

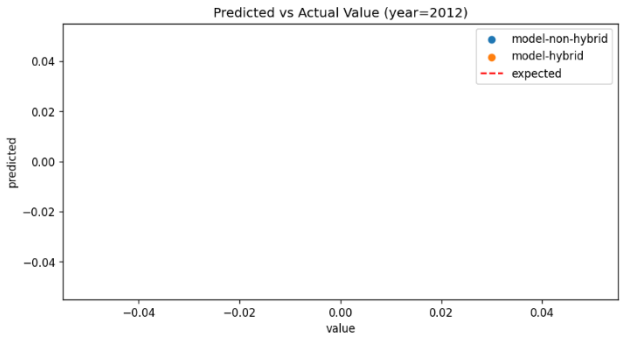


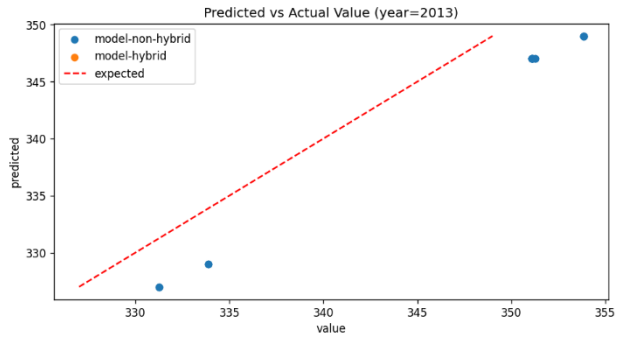


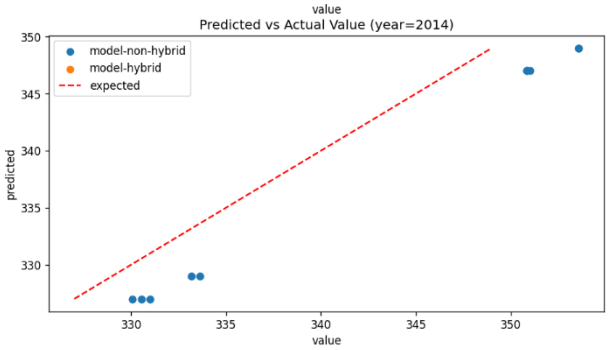


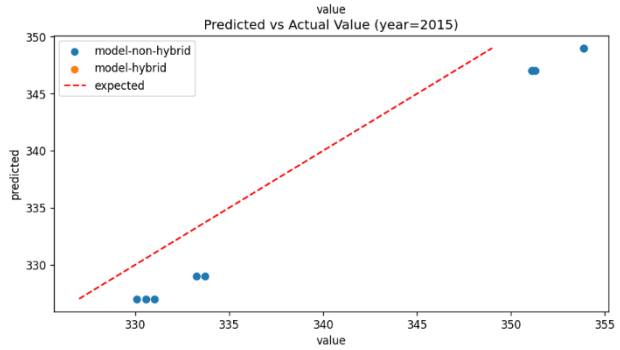


Predictions for ROLLS ROYCE

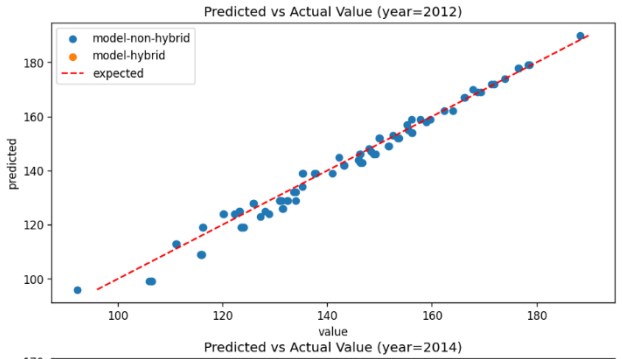


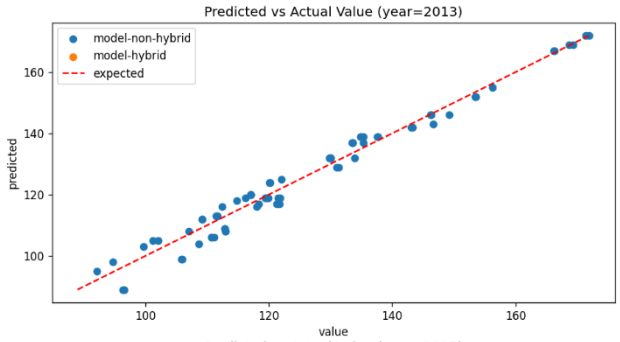


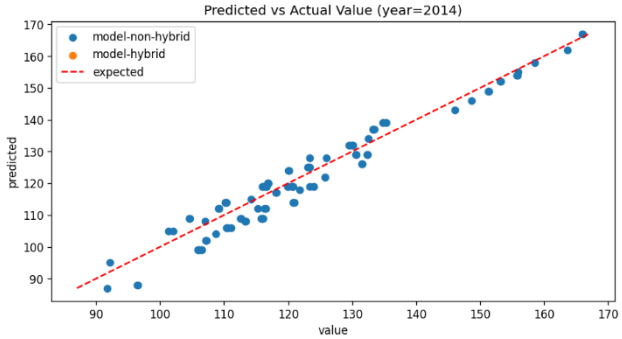


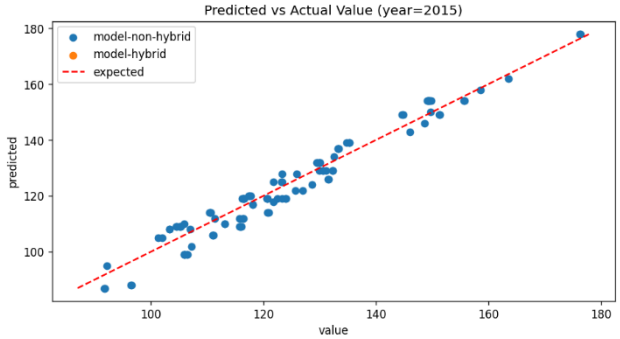


Predictions for SEAT

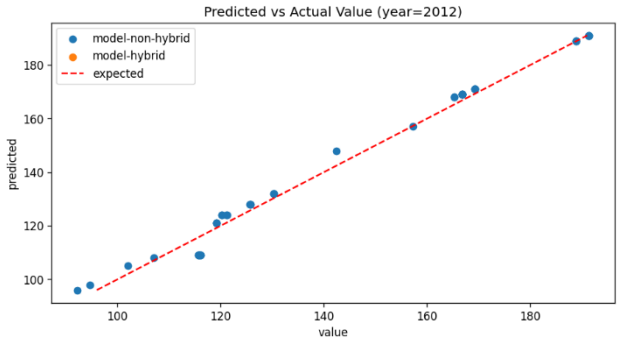


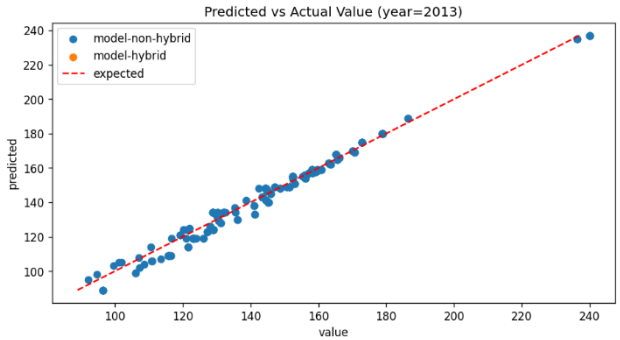


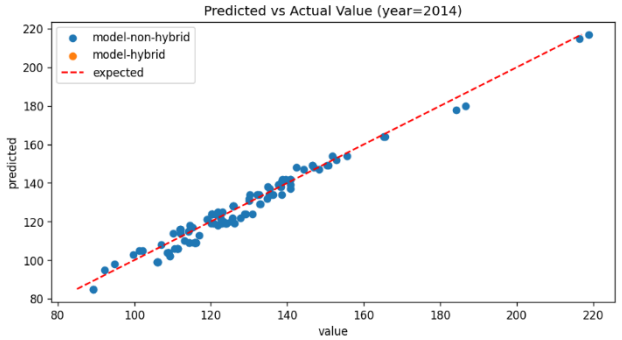


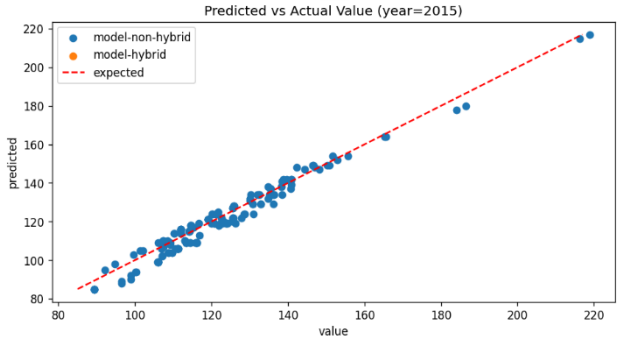


Predictions for SKODA

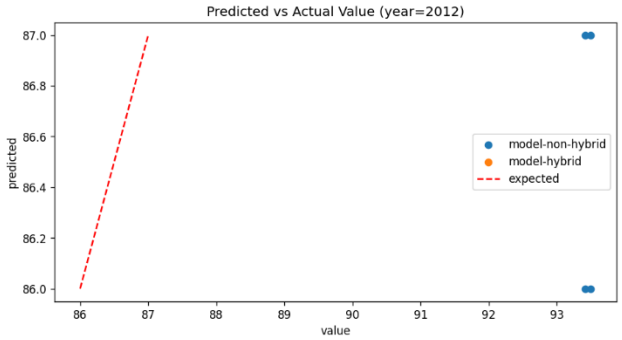


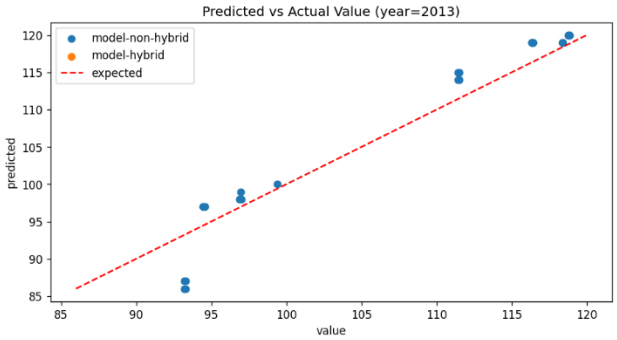


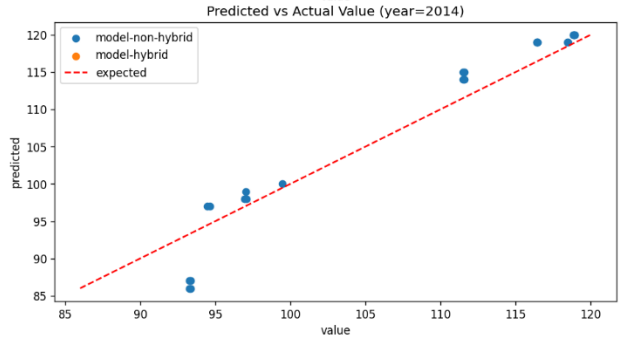


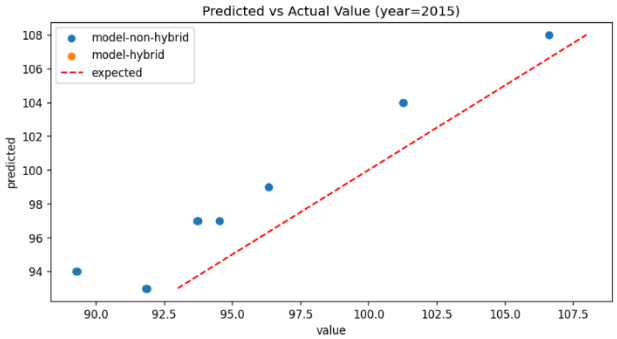


Predictions for SMART

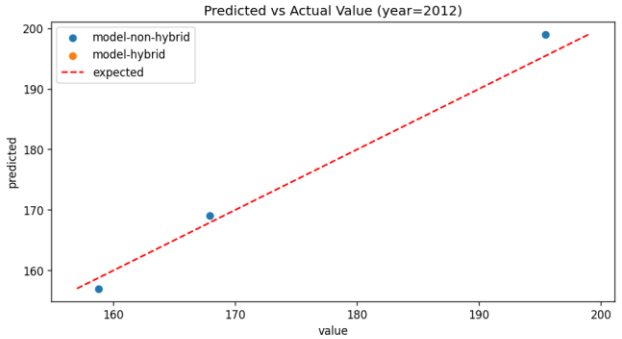


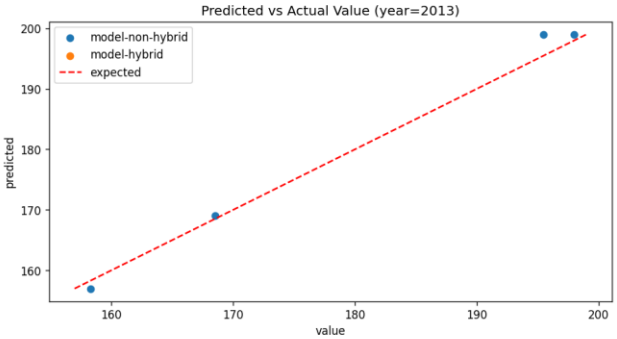


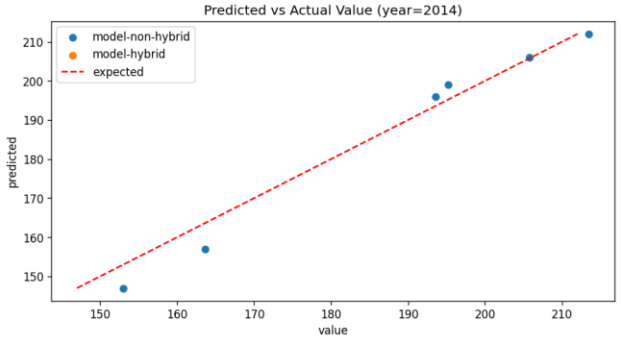


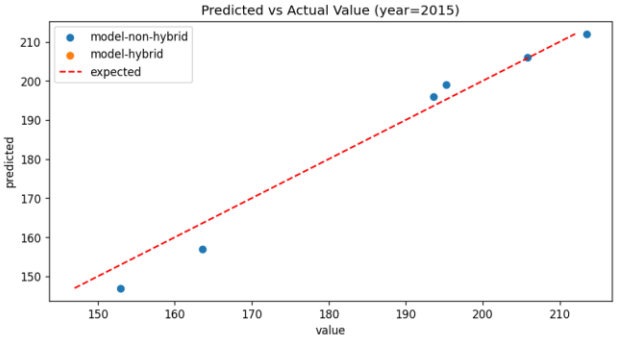


Predictions for SSANGYONG

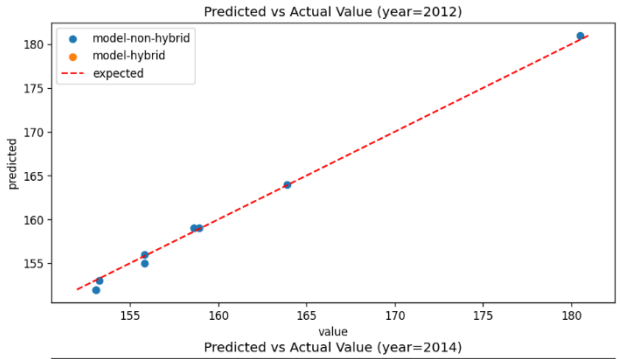


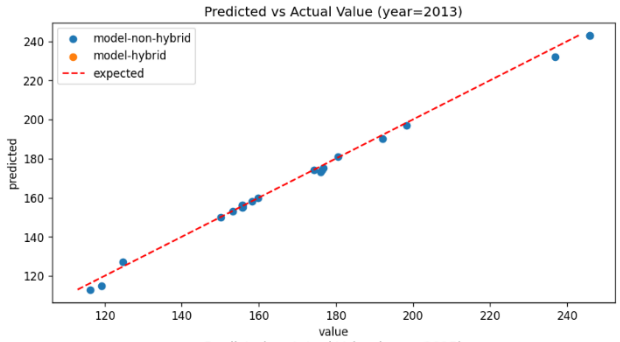


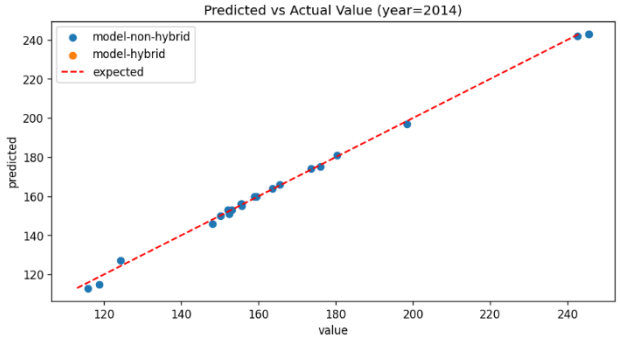


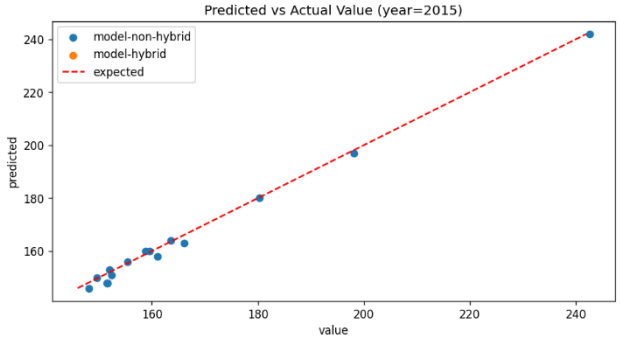


Predictions for SUBARU

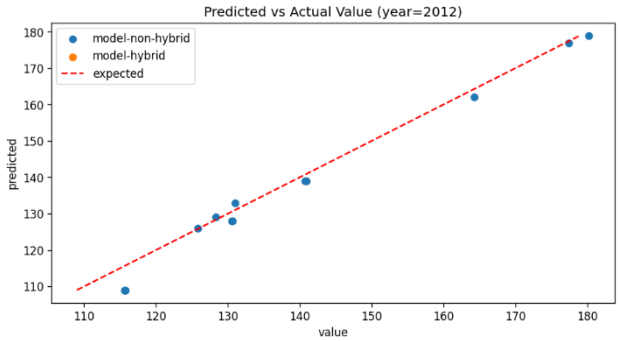


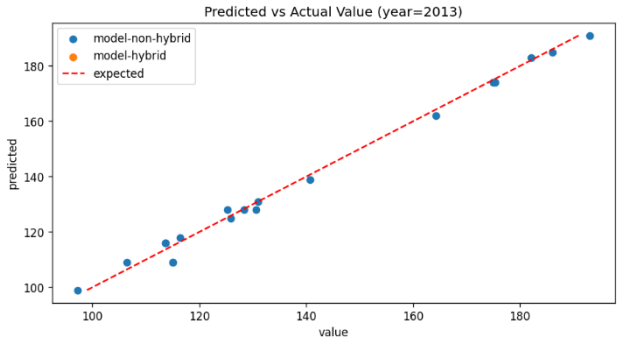


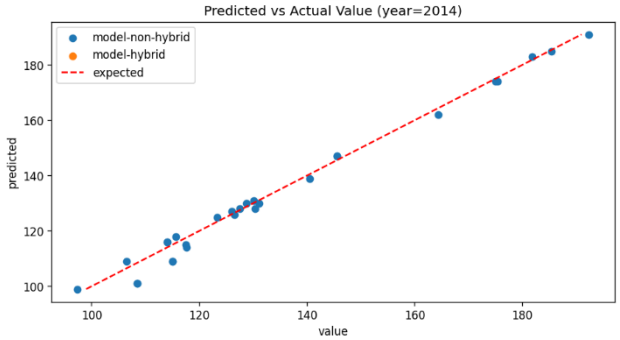


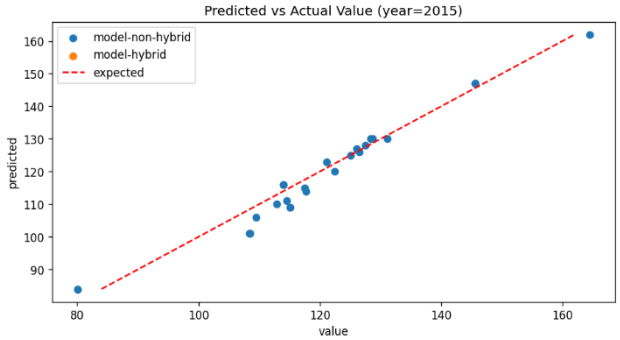


Predictions for SUZUKI

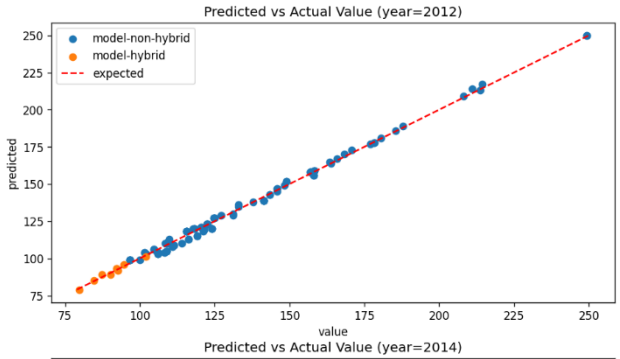


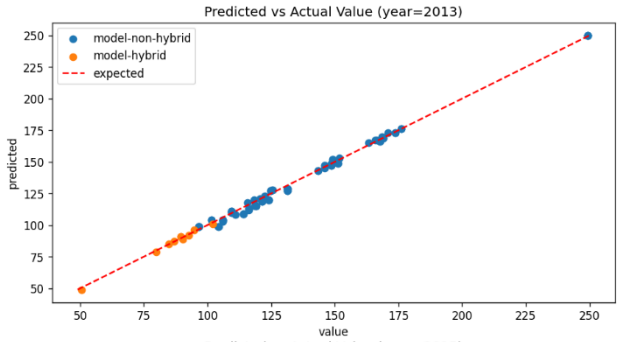


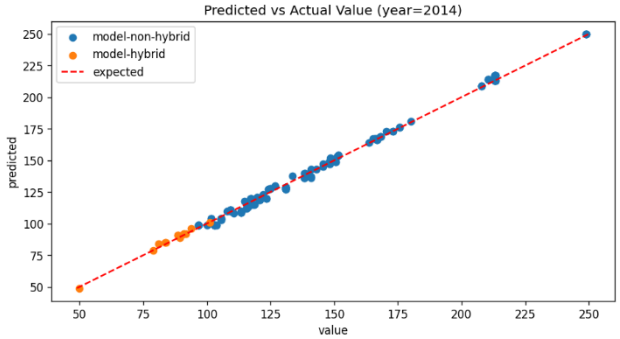


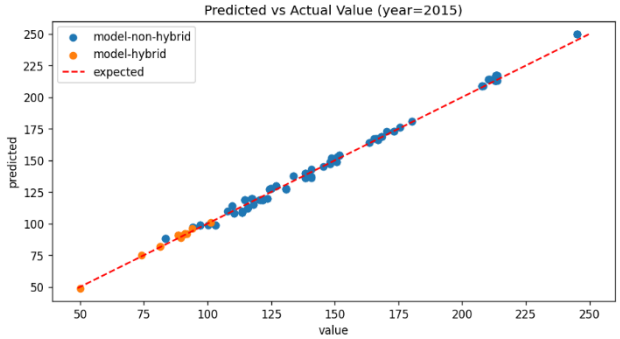


Predictions for TOYOTA

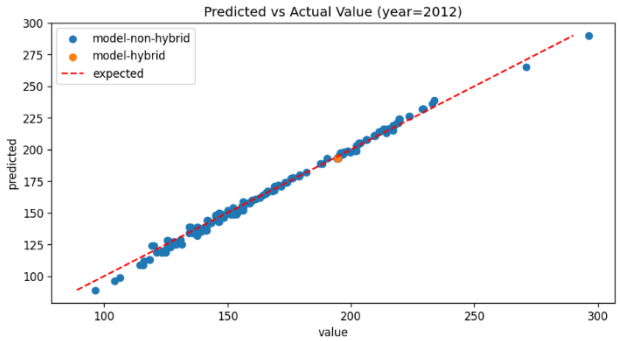


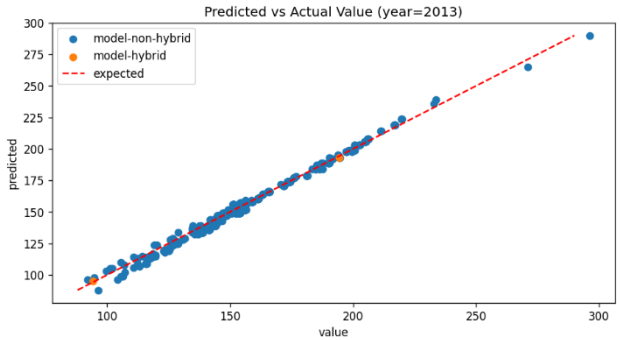


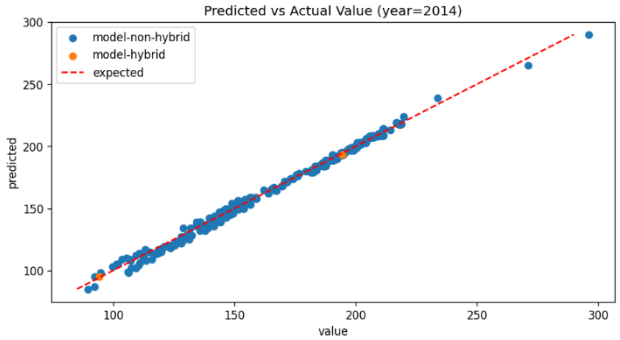


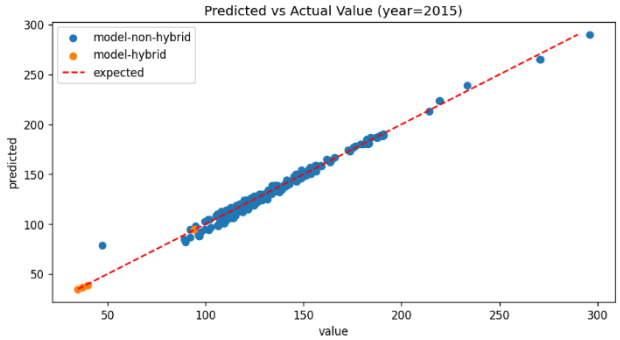


Predictions for VOLKSWAGEN









Predictions for VOLVO