



**Faculty of Engineering & Technology Electrical &
Computer Engineering Department**

ENCS5341

Machine Learning

Assignment 2

Prepared by:

Alhasan Manasra 1211705

Mohammad Khmour 1212517

Instructor: Dr. Ismail Khater

Section: 2 & 3

Date: November 28, 2024

Abstract

In this assignment, we focus on building and evaluating regression models to predict car prices using the YallaMotors car dataset. This work on assignment involves implementing both linear and nonlinear regression models, including Linear Regression, LASSO, Ridge Regression, Polynomial Regression, and Radial Basis Function (RBF) Kernel Regression. Also we checked missing values are handled, features are normalized, and the dataset is divided into training, validation, and test sets through data preprocessing.

To increase accuracy and reduce overfitting, the models are trained and refined using regularization techniques, gradient descent, and closed-form solutions. Validation measures such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared are used in the model selection process. The best-performing models are chosen by hyperparameter tuning, which includes adjusting regularization parameters and changing polynomial degrees.

Finding the most influential features also involves investigating feature selection through forward selection. Visualizations are shown to show discoveries and performance patterns, and the results are thoroughly assessed and contrasted on both the validation and test sets.

Table Of Contents

Abstract	I
Table Of Contents	II
Table Of Figures	III
Dataset and Attributes Description	1
Data Cleaning and Feature Engineering	2
● Normalize all numeric data	4
● Split the dataset into training, validation, and test set	4
● calculate the mean absolute error, mean square error, and r2 score for all modules	4
● Plot training data with validation data	5
● linear regression model	6
● lasso and ridge regression	8
● Closed-form solution	9
● Non-linear regression	10
● Model with Gaussian (RBF) kernel	11
● Feature Selection with Forward Selection	12
● Grid search	13
Conclusion	15

Table Of Figures

Figure 1 - number of missing value	2
Figure 2 - total missing values	3
Figure 3 - unique currency	3
Figure 4 - evaluation metrics for each regression model	5
Figure 5 - distribution of the training and validation data	5
Figure 6 - Linear Regression Fit Plot	6
Figure 7 - Residuals Distribution Plot	7
Figure 8 - performance of Lasso Regression and Ridge Regression models	8
Figure 9 - closed-form solution for Linear Regression	9
Figure 10 - Polynomial regression Degree 2.. Figure 11 - Polynomial regression Degree 10	10
Figure 12 - Kernel Ridge Regression model using a Gaussian (RBF) kernel	11
Figure 13 - Forward Feature Selection algorithm	12
Figure 14 - optimal alpha values and their respective validation errors	13
Figure 15 - performance of a Polynomial Regression model (degree 2)	14

Dataset and Attributes Description

According to the car dataset, there are 6308 samples and 9 features distributed in the following form:

Car Name: The specific name or model of the car (e.g., "Toyota Corolla" or "Ford Mustang"), providing an identifier for the car's make and type.

Price: The monetary value of the car, usually in a specific currency (e.g., USD, EUR). Used for regression models to predict car prices or to classify vehicle prices.

Engine Capacity: The size of the car's engine, typically measured in liters (e.g., 2.0L) or cubic centimeters (cc). It is important for evaluating performance and efficiency characteristics.

Cylinder: The number of cylinders in the engine indicates the engine's configuration (e.g., 4-cylinder or 6-cylinder). This helps assess the engine's power and fuel efficiency.

Horsepower: The power output of the car's engine, usually measured in horsepower (HP). A key feature for performance metrics like acceleration and speed.

Top Speed: The maximum speed the car can achieve, measured in kilometers per hour (km/h).

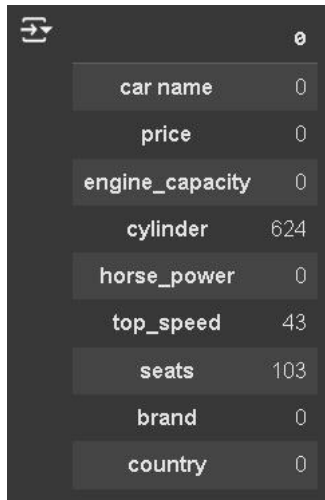
Seats: The number of passenger seats in the car, indicating its capacity (e.g., 2 seats for sports cars or 7 seats for SUVs).

Brand: The manufacturer of the car (e.g., Toyota, BMW, Ford), which provides insight into brand popularity and distribution in the market.

Country: The country where the car was manufactured or designed (e.g., Japan, USA, Germany). Useful for studying regional preferences and trends in automotive markets.

Data Cleaning and Feature Engineering

In the first step, we read the dataset from a file and specified that any values labeled as 'N A' or 'None' should be treated as missing values. Then, we computed the number of missing values in each column of the data frame.




	0
car name	0
price	0
engine_capacity	0
cylinder	624
horse_power	0
top_speed	43
seats	103
brand	0
country	0

Figure 1 - number of missing value

The output shows the count of missing values in columns like cylinder (624 missing values), top_speed (43 missing values), and seats (103 missing values).

To clean and normalize particular dataset columns, we carried out data validation and preparation in this stage. To make sure the price column adhered to the required format (currency and numeric value), a validation function (validate_price) was put in place, replacing invalid values with NaN. Likewise, non-numeric values in columns like cylinder and horsepower were changed to NaN, while invalid entries were forced to NaN by converting top speed and engine capacity to numeric values. NaN was also used to replace rows in the seat column that included invalid seat information. Following these cleaning procedures, the dataset's preparedness for additional processing was evaluated by recalculating the number of missing values in each column.

Each dataset column's missing values are compiled in the output. While the price has 1,329 missing entries, cylinder has 734, top_speed has 433, and seats have 519. Other columns with less missing values include engine_capacity and horse_power, which had 3 and 122 missing entries, respectively. In order to guarantee that the dataset is prepared for analysis, it is imperative that missing data in columns such as price, cylinder, top_speed, and seats be handled.




	0
car name	0
price	1329
engine_capacity	3
cylinder	734
horse_power	122
top_speed	433
seats	519
brand	0
country	0

Figure 2 - total missing values

To fix the dataset's missing values. The mode, or most frequent value, was used to fill in the missing data for the seat column. Likewise, the median values for engine_capacity, top_speed, cylinder, and horse_power were substituted for any missing values. Additionally, the cylinder column was changed to a numeric format, and the median was used to replace any erroneous values. Lastly, any records in the price column that had missing values were removed. Following these procedures, the dataset was cleaned, and the output verified that there were no missing values in any of the columns. This guarantees that the dataset is comprehensive and prepared for additional examination.

After that, we standardize all prices to a common currency into USD. So, we split the price values by whitespace and select the first part it's the currency code, and the second part for price. The output shows a list of unique currency codes:



```
[ ] list(df['price'].apply(lambda x: x.split()[0]).unique())
[ 'SAR', 'EGP', 'BHD', 'QAR', 'OMR', 'KWD', 'AED' ]
```

Figure 3 - unique currency

After that, we used preset exchange rates to convert the dataset's car price from many currencies to a single currency, USD. To map each currency (such as SAR, EGP, and BHD) to its exchange rate versus USD, a dictionary called exchange_rates was made. A new column called price_usd was created and used to hold the converted values. For precise analysis and modeling, this guarantees that all prices are standardized.

● **Normalize all numeric data**

All of the dataset's numerical data was normalized in this phase to minimize values between 0 and 1. First, the seats and cylinder columns were left out, and only the numeric columns (float64 and int64) were chosen. The normalization formula, which involves subtracting the minimum value and dividing by the range (highest value minus minimum value), was then applied to each numeric column. This guarantees that every numerical feature is on the same scale, which is crucial for enhancing machine learning models' stability and performance.

● **Split the dataset into training, validation, and test set**

The dataset is prepared and divided into test, validation, and training sets in this step. The feature `horse_power` is reshaped and allocated to `X`, while the target variable (`price_usd`) is assigned to `Y`. Categorical features could be one-hot encoded and coupled with numerical features to produce a processed dataset. After then, the dataset is divided: half is used for testing, and the other half is used for validation, leaving 40% of the data aside. This guarantees a suitable separation for training and assessing the model's functionality.

● **calculate the mean absolute error, mean square error, and r2 score for all modules**

This section assesses the performance of many regression models on a validation dataset, including Lasso, Ridge, Kernel Ridge, Polynomial, and Linear regression. Each model computes three performance metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2). It also fits the training data (`X_train`, `Y_train`) and makes predictions on the validation set (`X_val`). While Lasso and Ridge Regression use regularization to lessen overfitting, which is regulated by the alpha parameter, Polynomial Regression uses Polynomial Features to change features. An RBF kernel with predetermined alpha and gamma parameters is used in Kernel Ridge Regression. The performance measures for each model are shown in the final output, enabling comparison of how well each model predicts the target variable.


```

mae for linear Regression = 0.00734
mse for linear Regression = 0.00052
r2 score for linear Regression = -1.55059

mae for plnomyal Regression = 0.00904
mse for plnomyal Regression = 0.00051
r2 score for plnomyal Regression = -0.22413

mae for lasso Regression = 0.01417
mse for lasso Regression = 0.00105
r2 score for lasso Regression = 0.00000

mae for ridge Regression = 0.00739
mse for ridge Regression = 0.00052
r2 score for ridge Regression = -1.69585

mae for model Regression = 0.00738
mse for model Regression = 0.00053
r2 score for model Regression = -1.57439

```

Figure 4 - evaluation metrics for each regression model

Because of the size of the data, all models show very low errors (MAE and MSE); yet, their negative or nearly zero R2 values indicate poor explanatory power and an inability to make accurate predictions. The only one of these having a non-negative R2 is Lasso Regression, however accuracy issues persist.

● Plot training data with validation data

This stage involved analyzing the distribution of the target variable (y) against the feature (x) by visualizing the training and validation data using a scatter plot. The training examples are shown by blue dots, and the validation examples are shown by red dots. For clarity, the x and y axes are labeled, and a legend is included to help differentiate between the two datasets. By evaluating the distribution and overlap of data points, this graphic makes sure that both datasets are appropriately sampled for training and validation.

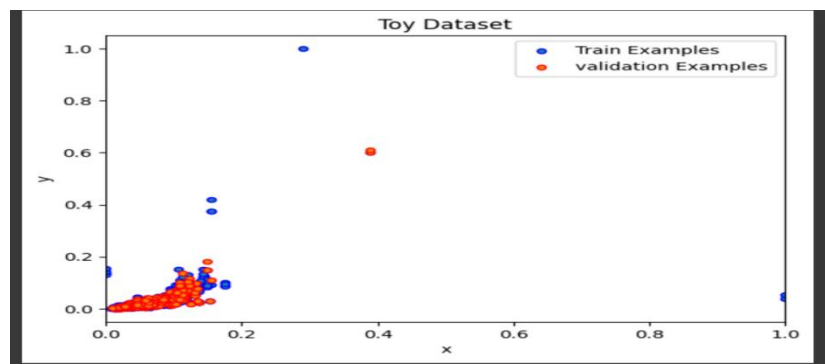


Figure 5 - distribution of the training and validation data

The blue dots represent the training examples, showing the data used to fit the model.

The red dots represent the validation examples, showing the data used to evaluate the model's performance.

The plot shows a concentration of smaller values for both the feature and the target variable, with the majority of the data points grouped close to the lower-left corner. A few dots are dispersed farther apart, indicating a greater range of values or outliers. A proper assessment of the model's performance depends on determining if the distributions of the training and validation datasets coincide, which is made easier with the aid of this graphic.

● linear regression model

This code analyzes the performance of a Linear Regression model by training and evaluating it. Predictions (y_{predict}) are made on the validation set (X_{val}) after the model has been trained using the training data (X_{train} and Y_{train}). To evaluate the prediction errors, residuals are computed, which show the discrepancy between the actual validation values (Y_{val}) and the predicted values. To illustrate the relationship that the model captured, a scatter plot is created to display the training data points and the model's predictions on the validation set. To examine the error distribution, a residuals histogram is also presented the ideal situation, in which residuals are centered around zero, is indicated by a red vertical line at zero. These visualizations help evaluate how well the model fits the data and whether the errors are randomly distributed, both of which are critical indicators of a good regression model.

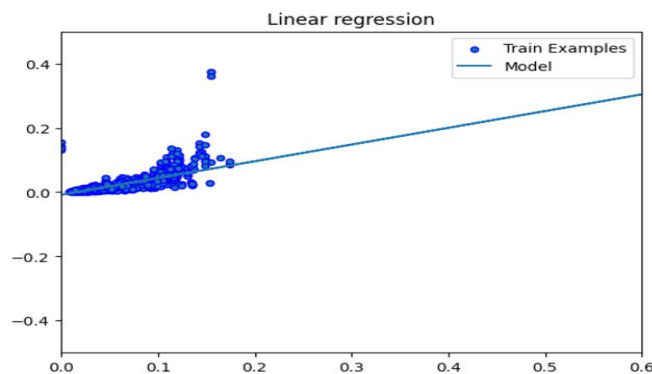


Figure 6 - Linear Regression Fit Plot

By displaying how well the Linear Regression model fits the training data, the figure above plot illustrates the model's performance. The model's anticipated relationship is depicted by the blue line, while the training examples are represented by the blue dots. The graphic shows that although the model attempts to depict a linear trend in the data, it may not provide the best match, particularly for data points that are more widely distributed. Given that it has trouble capturing the more intricate correlations found in the training examples, this shows that the model might be underfitting the data.

the figure below plot represents the distribution of residuals, which are the differences between the actual and predicted values of the target variable. The red vertical line, which denotes perfect predictions, highlights the fact that the majority of residuals are concentrated around zero in the histogram. Nonetheless, there is some spread in the residuals, which suggests prediction mistakes. A well-performing model is indicated by a symmetric, narrow distribution that is centered around zero. This figure aids in locating potential weak points in the model, such as greater residuals or symmetry deviations.

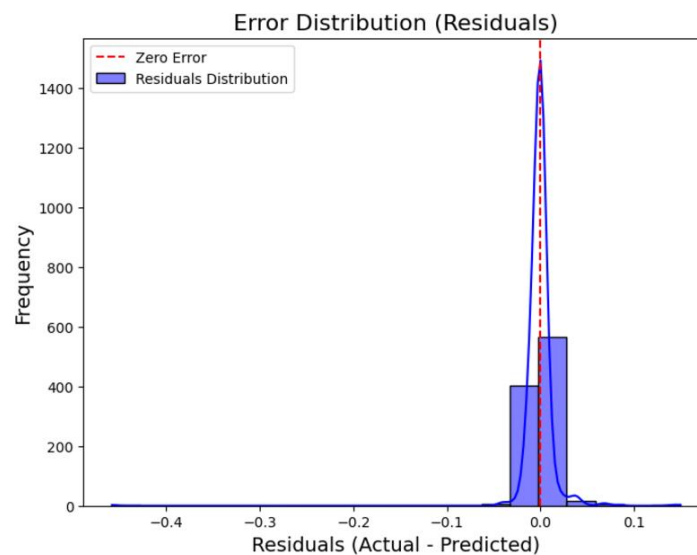


Figure 7 - Residuals Distribution Plot

● lasso and ridge regression

This part implements and visualizes the performance of Lasso Regression and Ridge Regression. Although regularization is used by both models to avoid overfitting, their approaches to handling coefficients are different. Lasso Regression penalizes big coefficients and can decrease some to zero, thereby performing feature selection, when alpha is set at 0.1. Ridge Regression is more stable when multicollinearity is present because it penalizes high coefficients without shrinking them to zero (alpha=0.1). The models' predictions are presented for comparison after they were trained on the same dataset (X_{train} and Y_{train}). The regression lines display the predictions, with the Ridge regression line being more flexible (orange) and the Lasso regression line being flatter (blue). The training instances are shown by the blue dots.

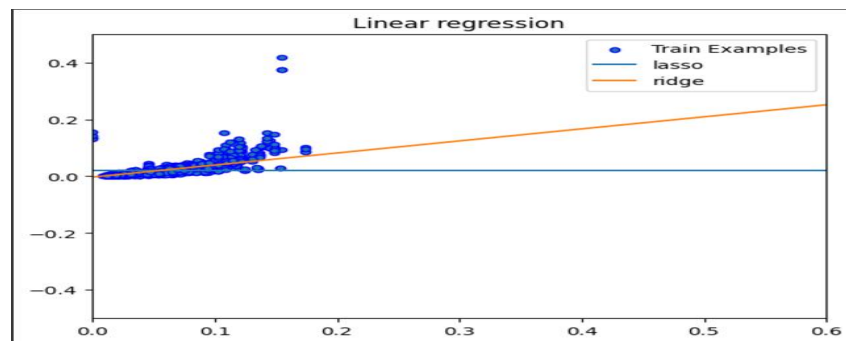


Figure 8 - performance of Lasso Regression and Ridge Regression models

The figure above comparing how well the Lasso Regression and Ridge Regression models fit the training data, the plot illustrates their respective performances. The lines show the regression predictions, with the orange line representing the Ridge model and the blue line representing the Lasso model. The blue dots show the training examples.

The Lasso model has used more regularization, which may have caused certain coefficients to decrease towards zero, as seen by the blue Lasso Regression line appearing flatter and closer to zero. As a result, the model becomes less complex and simpler.

Less aggressive regularization is seen by the orange Ridge Regression line, which has a steeper slope and adjusts more to the data. Ridge Regression produces a more flexible fit by penalizing the magnitudes of all the coefficients while keeping the others.

This graphic demonstrates how regularization affects model behavior. Lasso simplifies the model by maybe setting some of the coefficients to zero, whereas Ridge keeps all of the features and modifies their values to strike a balance between fit and complexity.

● Closed-form solution

In order to determine the ideal model parameters analytically, we applied the linear regression closed-form solution in this section. In order to account for the regression model's intercept, a bias term was first introduced to the feature matrix (X_{train}). The closed-form solution was computed using matrix operations, which involved multiplying the transposed feature matrix by the target variable (Y_{train}) and determining the inverse of the product of the transposed feature matrix and itself. The training data was then subjected to predictions ($Y_{\text{pred_closed_form}}$) using the final model parameters.

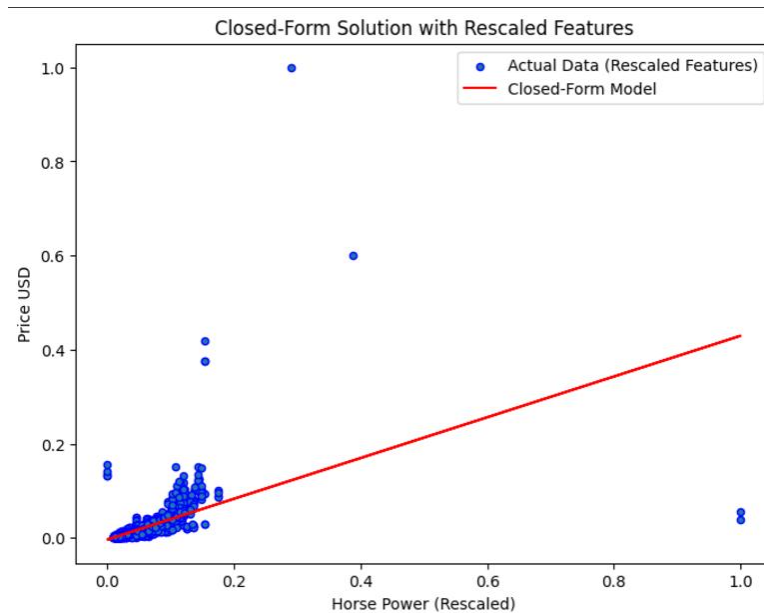


Figure 9 - closed-form solution for Linear Regression

The output was displayed as a scatter plot, with the red line denoting the closed-form solution's predictions and the blue dots representing the actual data points (rescaled features). The graphic, which displays a linear trend that the regression model was able to capture, demonstrates how well the model matches the data. By using the mathematical definition of linear regression without the need for iterative optimization approaches, this method guarantees an exact solution for the parameters.

● Non-linear regression

In this part of the project, we implemented Polynomial Regression to model the relationship between the feature (horse power) and the target variable (price) with varying degrees of polynomial features (2, 4, 6, 8, 10).

We do Feature Transformation that we use PolynomialFeatures class was used to transform the input feature (X_train) into higher-degree polynomial features, allowing the model to capture more complex relationships. And we use Model Training is a LinearRegression model was fitted to the transformed polynomial features for each degree. This extends the linear regression framework to non-linear relationships. Also Visualization for each degree, the training data is represented as blue dots, and the model's predictions are plotted as a blue line, showing how well the model fits the data.

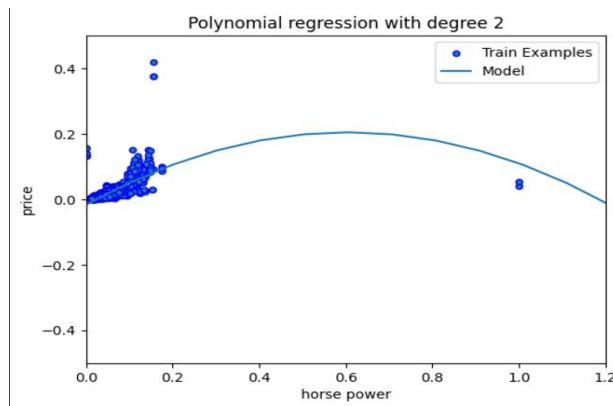


Figure 10 - Polynomial regression Degree 2

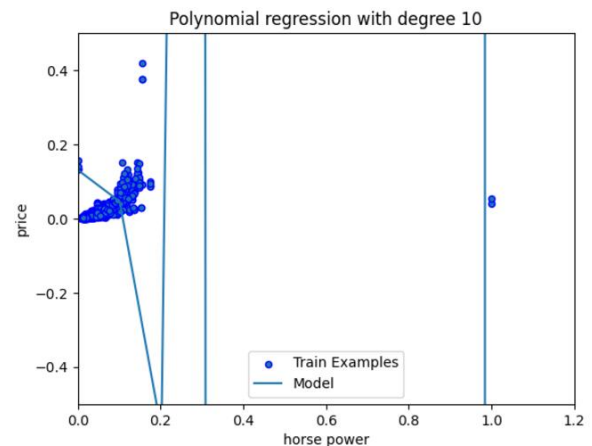


Figure 11 - Polynomial regression Degree 10

Degree 2: Although the model captures a smooth curve that somewhat resembles the distribution of the data, it underfits in several areas, suggesting a lack of flexibility.

Degree 4: As the curve starts to oscillate and veer off course from the overall trend, the model starts to exhibit symptoms of overfitting, despite becoming more adaptable and able to capture more abrupt changes.

Higher Degrees (6, 8, 10): The model gets more sophisticated as the degree rises, overfitting the training set. Extreme oscillations and irrational forecasts are the results, particularly in areas with limited data.

● Model with Gaussian (RBF) kernel

In order to capture non-linear correlations between the feature (Horse Power) and the target variable (Price USD), we used a Gaussian (RBF) kernel in our Kernel Ridge Regression model. The model was trained using the X_{train} and Y_{train} training data, with the hyperparameters gamma (kernel parameter) set to 0.5 and alpha (regularization strength) set to 1.0. The test data (X_{test}) was used to make predictions, and the performance was assessed using the Mean Squared Error (MSE), which came out to be 0.00063, suggesting an excellent fit.

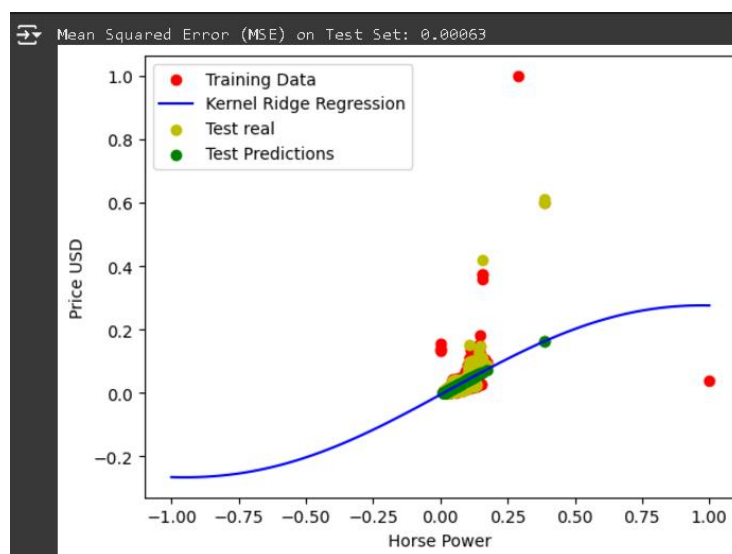


Figure 12 - Kernel Ridge Regression model using a Gaussian (RBF) kernel

The training data is represented by red dots in the output visualization.

A smooth blue curve representing the learnt relationship represents the model's predictions on a fine-grained range of inputs.

The model's predictions for the test data are represented by green dots, whereas the actual test data (Test real) is represented by yellow dots.

By closely matching the test predictions (green) with the actual test values (yellow), this graphic illustrates how effectively the model generalizes. As seen by the low MSE value and the smooth blue curve, the Gaussian kernel successfully captures the non-linear trend in the data while retaining strong generalization.

● Feature Selection with Forward Selection

In this section, we used a Forward Feature Selection approach to determine which features were most crucial for utilizing Linear Regression to predict automobile prices. The dataset is first divided into training and validation sets. The algorithm iteratively evaluates adding one feature at a time from the remaining features after initially selecting none. The model is trained on the training set for each candidate feature, and Mean Squared Error (MSE) is used to assess the model's performance on the validation set.

The chosen features list is expanded to include the option that offers the largest MSE reduction. Until there is no more improvement in MSE, this process is repeated.

```
Added feature: horse_power, MSE: 0.00139
Added feature: top_speed, MSE: 0.00131
Added feature: brand_lotus, MSE: 0.00127
Added feature: cylinder, MSE: 0.00124
Added feature: brand_mclaren, MSE: 0.00120
Added feature: brand_dodge, MSE: 0.00119
Added feature: country_ksa, MSE: 0.00118

Selected Features:
['horse_power', 'top_speed', 'brand_lotus', 'cylinder', 'brand_mclaren', 'brand_dodge', 'country_ksa']
```

Figure 13 - Forward Feature Selection algorithm

From the output, the selected features are: horse_power, top_speed, brand_lotus, cylinder, brand_mclaren, brand_dodge, and country_ksa. These features were chosen because they contributed to minimizing the validation MSE, with the final MSE reaching 0.00118. This method ensures that only the most impactful features are included in the model, improving its predictive accuracy and efficiency.

● Grid search

In this section, we used grid search with cross-validation to accomplish hyperparameter tweaking for Ridge Regression, Lasso Regression, and SGD Regressor. With 50 values, a range of regularization strengths (alpha, which stands for lambda) was defined, ranging from 10^{-4} to 10^3 . In order to get the ideal regularization parameter that minimizes the Mean Squared Error (MSE), grid search examined every potential value of alpha for each model.

Five-fold cross-validation was used to apply the grid search to the training dataset, and the best alpha value and matching model were taken out. The validation error (MSE) was then calculated by evaluating the chosen model on the validation set.

```
Ridge()
Optimal  $\lambda$ : 0.71969
Validation Error: 0.00031
Lasso()
Optimal  $\lambda$ : 0.00010
Validation Error: 0.00031
SGDRegressor()
Optimal  $\lambda$ : 0.00019
Validation Error: 0.00040
```

Figure 14 - optimal alpha values and their respective validation errors

The results show the optimal alpha values and their respective validation errors:

Ridge Regression: Optimal $\alpha=0.71969$, Validation Error = 0.00031

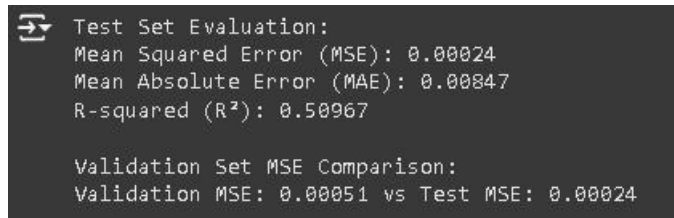
Lasso Regression: Optimal $\alpha=0.00010$, Validation Error = 0.00031

SGD Regressor: Optimal $\alpha=0.00019$, Validation Error = 0.00040

This process ensures that each model uses the best regularization strength for improved performance while balancing bias and variance.

After that we assessed how well a Polynomial Regression model (degree 2) performed on the test and validation datasets. Initially, the model was trained with the training data's modified

polynomial characteristics. To find the best fit, the Mean Squared Error (MSE) was used to evaluate the performance on the validation set. The model's capacity for generalization was then assessed by applying it to the test set and computing important metrics such as Mean Absolute Error (MAE), R-squared (R^2), and MSE.



```
Test Set Evaluation:
Mean Squared Error (MSE): 0.00024
Mean Absolute Error (MAE): 0.00847
R-squared (R²): 0.59067

Validation Set MSE Comparison:
Validation MSE: 0.00051 vs Test MSE: 0.00024
```

Figure 15 - performance of a Polynomial Regression model (degree 2)

The output of this part explain it to us that:

Low prediction error is indicated by the Test Set MSE of 0.00024.

Test Set MAE: 0.00847, which displays the mean absolute prediction error.

Test Set R2: 0.59067 indicates that approximately 59% of the volatility in the test data can be explained by the model.

Comparison of Validation and Test MSE: The test set MSE (0.00024) was less than the validation set MSE (0.00051), indicating that the model demonstrated good generalization and performed marginally better on the test data that was not visible.

This research demonstrates how well the model fits the data while preserving low error on the test and validation sets.

Conclusion

In this project, we used a dataset that needed extensive feature engineering and preprocessing to test different regression models for predicting car.csv file. To guarantee a high-quality dataset for modeling, the project started with data cleaning, normalization, and addressing missing values. We used grid search for hyperparameter tuning and constructed and assessed several models, such as Linear Regression, Polynomial Regression, Lasso Regression, Ridge Regression, and Kernel Ridge Regression with Gaussian (RBF) kernels. In order to maximize the model's performance by concentrating on pertinent predictors, forward feature selection was also employed to determine the most significant features.

As the model complexity rose, we were able to observe the trade-offs between underfitting and overfitting through these experiments. Polynomial and Kernel Ridge Regression successfully handled non-linear patterns, but linear models were inadequate for more intricate interactions. By reducing overfitting, regularization strategies like Lasso and Ridge enhanced model generalization. Low validation and test errors demonstrate that hyperparameter tweaking and cross-validation guaranteed the best parameter selection for every model.

With low Mean Squared Error (MSE) and significant R-squared values, the final models showed good performance overall, suggesting accurate and trustworthy predictions. In order to create successful predictive models, this assignment emphasizes the significance of methodical data preparation, model evaluation, and striking a balance between model complexity and generality.

● Working on project

In the first part of cleaning data we do it together because the dataset is not clear and take a lot of time and working. After that Khdour take the linear and non-linear models parts, part 3, part 5 and part 6 do it together.

Alhasan do the closed-form solution from part 2, part 4 and part 7.

The report Alhasan do the abstract and conclusion and all other parts that he do it and take linear and non-linear models. Khdour do all other parts that he do it in code.