Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

ENCS5341

Machine Learning and Data Science

Assignment 2

Regression Analysis and Model Selection

Sarah Hassouneh **1210068**

Sondos Qasarwa **1210259**

Instructor: **Dr. Ismail Khater**

Section 2

Date: **27 November 2024**

Table of contents

Table of figures

List of tables

Project Overview

In this project we aim to discover different regression techniques and methods of enhancement. Using the Car data set provide we essentially want to build a regression model that is able to predict the car price based on its different features. We Build a series of regression models using the provided dataset, these included both linear and nonlinear models. In addition we applied various techniques including regularization techniques, hyperparameter tuning and feature selection method to improve model accuracy and prevent overfitting. We evaluated each of the model and compare different models' performance. In this report, we will demonstrate the steps taken to build a final model capable of generalizing to unseen data with an R^2 score of 0.67, indicating that the model explains 67% of the variance in the target variable.

Data Preprocessing

Data Set Description

The cars dataset provided includes around 6,308 rows and 9 columns. It is considered to be a small dataset but it has very sophisticated features that can help sufficiently create a regression model. It's suited for Exploratory Data Analysis (EDA) and for predictive modeling with Linear Regression. The main objective of this dataset is to **predict car prices,** making it ideal to understand the relationship between various features and **the target variable (car price).**

The table below describes the dataset columns:

Column	Description	Measurement Level
Car Name	The name of the car	Nominal
Price	The price of the car	Numeric
Engine Capacity	The car engine capacity	Numeric
Cylinder	The car cylinder power	Numeric
Horsepower	The car horsepower	Numeric
Top Speed	The car top speed	Numeric
Seats	Number of seats in the car	Numeric
Brand	The car brand	Nominal
Country	The country where the site sells this car	Nominal

Data Preprocessing

We first imported the data and applied some EDA techniques to be able to understand the data better. These were some of the **initial observations** from the data:

1. The column car name serves as an identifier and it not missing for any of the cars and so no more processing for this part needed.

2. The price is listed in different currency. Further, some are (TBD, Following, Discontinued), these entries need to be removed for Linear regression because it is a supervised learning model and missing target values make it impossible to calculate the error or fit the model properly.
3. The engine capacity is not standard some are denoted in liters as (1.4, 2.6) and others are in cubic centimeters like (1,400) and these need to be standardized
4. The cylinder column has N/A values for electric cars, because electric cars do not have cylinders and this needs to be handled. Also some are stated as (Single or Double) that needs to be handled.(will be shown later)
5. The top speed column has some incorrect entries (like 4 Seaters) and this need to be handled. Further some values are (automatic or N A) and these needs to be addressed.
6. The seats column has some incorrect entries , either floats or too large entries like 150.
7. The brand and country is correctly entered for all.

Data Cleaning

In this part we want essentially performed the following:

- Handle missing values
- Handle incorrect entries or values
- Encoding categorical features (brand, country)
- Normalize or standardize numerical features where necessary
- Split the dataset into 60% for training, 20% for validation, and 20% for testing. This is a recommended measure and is suitable for small data sets like the set provided .

Below we' ll explain the steps taken to ensure each how anomalies of different features were handled :

⇒ **Price :**

For price there were two issues to be handled :

1. Missing values : some values were marked as (' TBD', 'N A', 'Following', 'DISCONTINUED', 'Follow'), it is important to understand that these are target features, so any adjustments or inference on them would mean introducing bias to the model. The only solution here would be to remove them.
2. Wrong entries : some values where invalid prices were wrongly entered as colors for example ('Carrara White Metallic', 'Jet Black Metallic'), these samples also needed to be removed.
3. Prices are listed in different currencies: Prices were listed in different currencies , we had to standardize them, so we standardized all of them to USD. This will ensure that we avoid discrepancies and improve the accuracy of any predictive modeling

After the correct processing of price 1329 samples were lost, that is 21% of the original data. It is imporatnt to note that as for these values this loss can't be minimized any further.

⇒ **Engine Capacity :**

To ensure consistency, we need to standardize all engine capacities to liters and see we would need to converts any values in cubic centimeters to liters. We did that be dividing by 1000 for values larger than 1000.

⇒ **Cylinder:**

The most pressing issue with the cylinder was having many missing values, around 600 records. Removing these would waste many of the samples and so we investigated replacement methods. The following plot shows the frequencies of cylinders values:

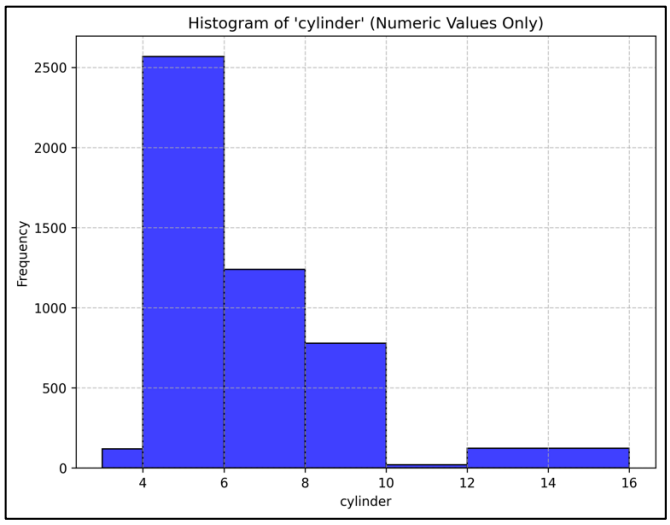

Figure 1: Cylinder Histogram

We can observe that the plot is highly positively skewed, the clacultaed skew was around 1.40. The mean is 5.36 and median: 4.00 and mode: 4.00. This sugests that replaving with the mean could bnot well represent the data and so we chose to replacemissing values with mode.

Also, we will apply feature extraction here, in the cylinder column if it is stated electric we will set to 0. For most vehicles, cylinder counts are typically 3, 4, 6, 8, 10 or 12, using a negative value does not overlap with valid cylinder counts (e.g., 3, 4, 6, 8, etc.).

It explicitly indicates the absence of a conventional cylinder count, making it interpretable as "no cylinders."

⇒ **Horsepower:** We replace Single, Double, Triple with their standard values

⇒ **Speed**: Removed any non-numeric values.

⇒ **Seats**: We extracted seats into their numeric values, so values like 5 seaters need to be stated as 5.

Note : reaching this here some values that were already invalid like floats (1.4, 2.5) and large values (150), were already removed since there has been multiple issue with these records entry.

Summary:

The current number of records is 4622, the original was 6308. We lost around 26% of the data, we will continue and assume acceptable. The original data had 1329 records missing price (target), that is 21% of the data that cannot be used for a supervised learning problem like regression. So in other words, we lost 5% in the data cleaning for the data with correct targets.

Encode Categorical Features

Perform one-hot encoding for country.

Given that the number of countries is only 7, one-hot encoding is suitable and was used for each country adding 7 new features.

Perform Target encoding for country**

Mean encoding involves replacing each category with the mean value of the target variable for that category. This method can be very effective, especially for supervised learning tasks, but it can lead to overfitting if not managed properly. It is suitable for categorical features exhibiting a high number of categories.
Because we will later use regularization techniques, this will be safe to use. Target encoding leverages the relationship between categorical variables and the target variable, making it a powerful encoding technique when this relationship is significant. However, one of the significant drawbacks of target encoding is the potential for overfitting, especially when applied to small datasets. It suffers from the problem of target leakage as the target variable is used to directly encode the input feature and the same feature is used to fit a model on the target variable.

This is important to note because this type of encoding should only be applied on the training data after the split. So we did apply it after splitting data to prevent data leakage. The encoder learns the target means for each category in the training data, and then we use the fitted encoder

to apply the same encoding to validation and test datasets.

Normalize Numeric

Normalization is important as it helps us create faster and more effiecnt models. All the numeric features were normalized using the Z-Score normalization. This normalization scheme has many advantages as it allows us to directly connect to mean and understand relationships and any outliers.

Split Data:

A recommended split was used here to split data into 60% for training, 20% for validation, and 20% for testing. This is suitable for small datasets and will give us enough values to train, validate and correctly state performance.

Building Linear Models:

Simple Linear Model

We started this part by build a simple linear model using built in library. Table 1 (see feature extraction part) shows the mean squared error, coefficient of determination (R^2) and the mean absolute difference metrics as evaluated on the validation set. This intial model has a large Mean Squared Error of 1194863124, R^2 Score of 0.65 and a mean Absolute Error of 26655. Suggesting that this model would need imporovement.

Closed form

In the closed form solution we aim to find the coefficients (weights) of each feature by solving the model using linear algebra techniques. In this context we refer to each weight as β_0, β_1 ,

where β_0 is wight of the first feature and so on. The $\boldsymbol{\beta}$ **stated in bold** represent the matrix of the coefficients , derivation of the equation and $\boldsymbol{\beta}$ is shown as follows:

As compared to the simple linear model we concluded with the same metrics as the built-in library as seen in Table1.

The formula for the regression coefficients $\boldsymbol{\beta}$ is:

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Where:

- $\mathbf{X}$: Feature matrix (including bias).
- $\mathbf{y}$: Target vector.

The bias term is added by concatenating a column of ones to the feature matrix:

$$\mathbf{X}_{\text{train}} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots \\ 1 & x_{21} & x_{22} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Similarly bias is added for $\mathbf{X}_{\text{val}}$ and $\mathbf{X}_{\text{test}}$.

Once $\boldsymbol{\beta}$ is computed, predictions for the validation and test sets are calculated as:

$$\mathbf{y}_{\text{val}} = \mathbf{X}_{\text{val}}\boldsymbol{\beta}$$

$$\mathbf{y}_{\text{test}} = \mathbf{X}_{\text{test}}\boldsymbol{\beta}$$

Where:

- $\mathbf{X}_{\text{val}}$: Validation feature matrix (including bias term).
- $\mathbf{X}_{\text{test}}$: Test feature matrix (including bias term).
- $\mathbf{y}_{\text{val}}$: Predicted values for the validation set.
- $\mathbf{y}_{\text{test}}$: Predicted values for the test set.

Gradient descent

Here we applied the gradient descent to search for the coefficients. We implemented this with a learning rate(alpha) equal to 0.1 and number of iterations equal to 200. The mathematical equations and solution using linear algebra can be analyze as follows:

We start by adding Bias Feature to all matrices, where the bias term is added to the feature matrix $\mathbf{X}$ by concatenating a column of ones:

$$\mathbf{X}_{\text{train}} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}$$

Similarly for $\mathbf{X}_{\text{val}}$ and $\mathbf{X}_{\text{test}}$.

Then we use random initialization to initialize the weight vector $\mathbf{w}$ (which represents initial values of the coefficients)

For each iteration $t = 1, \dots, T$, the predicted values are computed as:

$$\hat{\mathbf{y}} = \mathbf{X}_{\text{train}} \cdot \mathbf{w}.$$

The gradient of the Mean Squared Error (MSE) with respect to $\mathbf{w}$ can be expressed as:

$$\mathbf{g} = -\frac{1}{n}\mathbf{X}_{\text{train}} \cdot (\mathbf{y} - \hat{\mathbf{y}})$$

Where:

- $\mathbf{y}$: Actual target values.
- $\hat{\mathbf{y}}$: Predicted target values.
- n: Number of samples.

We then Update Weights using the gradient (in opposite direction) and the learning rate α :

$$\mathbf{w} = \mathbf{w} - \alpha \cdot \mathbf{g}$$

Where α is the learning rate.

Note : The MSE at each iteration can be calculated as follows:

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

After finding the final weights, predictions are made using them as follows:

$$\hat{\mathbf{y}}_{\text{val}} = \mathbf{X}_{\text{val}} \cdot \mathbf{w}$$

$$\hat{\mathbf{y}}_{\text{test}} = \mathbf{X}_{\text{test}} \cdot \mathbf{w}$$

The following figure shows the reduction in the mean squared error over iterations, we can see a large increase at the start, reaching almost a stable value at the 100th iteration.

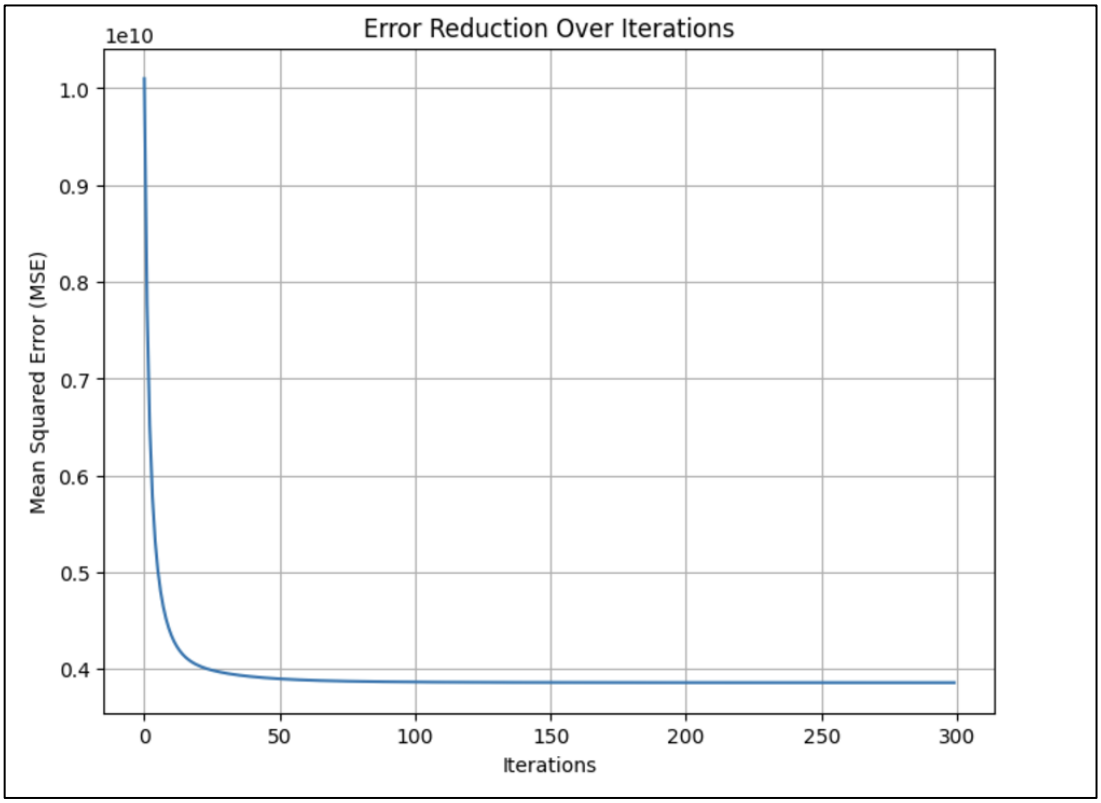

Figure 2: Gradient Descent Error Reduction over Iterations

As compared to the simple linear model we concluded with the same metrics as the built-in library and closed form as seen in Table1.

Applying Regularization technique

LASSO Regression (L1 Regularization)

Lasso stands for Least Absolute Shrinkage and Selection Operator. It is one of regularization techniques, which are typically used to prevent overfitting. By employing a regularization

technique in the conventional regression equation, Lasso Regression improves upon the linear regression idea.

The regularized loss function is: $L_{\text{LASSO}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w^T x_i})^2 + \lambda \sum_{j=1}^{d} |\text{w}_\text{j}|$

λ is the tuning parameter that controls the strength of the penalty. As lambda increases, more coefficients are pushed towards zero. selecting the tuning parameter lambda is essential. Grid search method is employed to determine the ideal value of lambda that strikes a balance between predicted accuracy and model complexity.

The obtained results indicates that the best λ value is 1000.

After choosing the optimal value, the model is evaluated on the validation set to be compared with the other models in model selection process. the performance metrics values are:

Mean Squared error	Mean Absolute Error	R-squared
1152940216.08	25628.79	0.662

Table 1-Performance measures for LASSO regression

The performance measures (MSE and MAE) are large because the target variable was neither scaled nor normalized. The following figure shows the predicted values using LASSO regression versus the actual values.

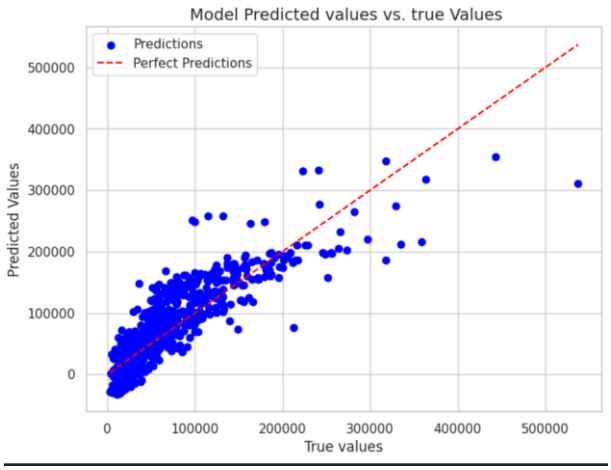

Figure 3- LASSO Predicted VS true values

Ridge Regression (L2 Regularization)

Ridge regression is a technique used for solving overfitting and multicollinearity issues typically associated with standard least squares regression. It achieves this by shrinking the model's coefficients, hence removing coefficient bias and lowering the mean square error.

The regularized loss function is: $L_{Ridge}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w^T x_i})^2 + \lambda \sum_{j=1}^{d} w_j^2$

Ridge regression is implemented in our code by solving its closed form solution. Optimal weights for ridge regression to minimize the loss function for : $\mathbf{w} = (\mathbf{X}^{\top\mathbf{X}} + \lambda \mathbf{I_d})^{-1}\mathbf{X}^{\top\mathbf{y}}$

Selecting the optimal value of λ

The λ is the regularization hyperparameter. It is essential for regularization because it controls how severely the model coefficients are penalized. Ridge Regression turns into a regular linear regression when λ is 0. The penalty on the coefficients rises with increasing λ, and the coefficient values fall toward zero. As the model gets simpler, this may decrease overfitting. But if λ is too large, the model may become too simple and underfit, failing to effectively capture the complexity of the data. As a result, choosing λ requires careful consideration.

Different values of the regularization parameter λ are used with ridge regression and grid search is used to find the optimal λ value that minimizes the error on the validation set.

The following figure shows the relationship between the mean squared error on the validation and log(λ) using ridge regression model.

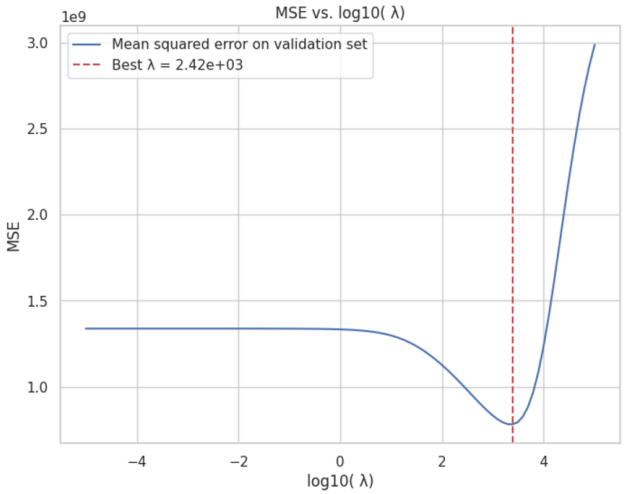

Figure 4- MSE VS log(λ)

The obtained results indicates that the best λ value is 2420.128.

After choosing the optimal value, the model is evaluated on the validation set to be compared with the other models in model selection process. the performance metrics values are:

Mean Squared error	Mean Absolute Error	R-squared
780895703.98	18911.86	0.77

Table 2- Performance measures for ridge regression

The performance measures (MSE and MAE) are large because the target variable was neither scaled nor normalized. The following figure shows the predicted values using ridge regression versus the actual values.

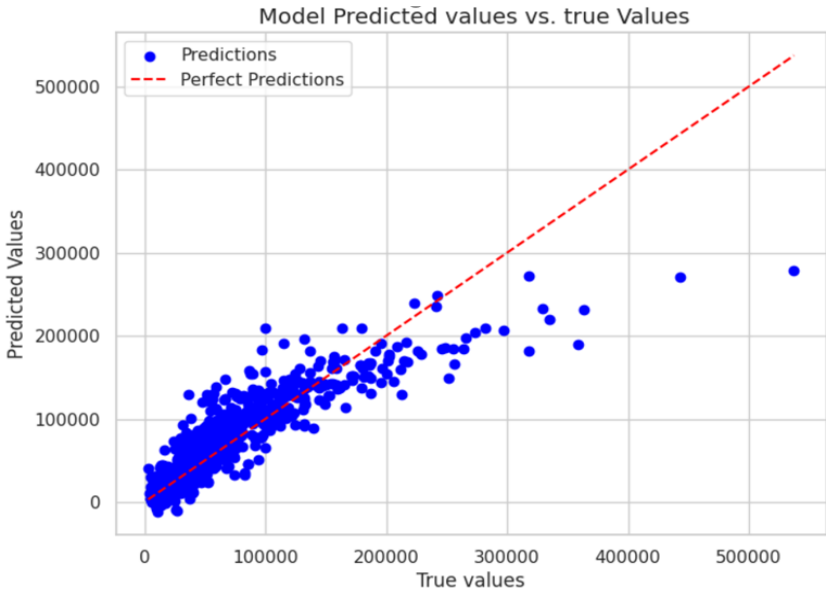

Figure 5-Ridge Predicted Values VS true values

Nonlinear models

Polynomial Regression

Polynomial regression is a type of regression analysis where the nth degree polynomial is used to model the relationship between the independent and dependent variables. Polynomial regression is typically employed when the linear regression model is unable to adequately describe the best result and the data points are not collected.

With polynomial regression, the initial input variable is transformed into polynomial features, such as x^2, x^3, ……, x^d, where d is the polynomial's degree. A linear regression model is used to fit the data after the polynomial features have been created.

The polynomial degree is varied from 2 to 10. the models are evaluated on the validation set to be compared with the other models in model selection process. The following table shows the mean squared error for the degrees 2, 3, and 4 on the validation set.

Degree	**MSE**
2	1964665717.01
3	166555178282.47
4	10980663014499.5
5	542159076289135
6	2.56+16

Table 3-Performance measures for polynomial regression

The best polynomial degree was 2, since it successfully caught the underlying pattern in the data and had the lowest error. Larger errors were the consequence of overfitting, which occurs when the model fits the noise in the training data instead of the actual relationship. This results in poor generalization on test or validation data.

The following figure shows the relationship between the predicted values and actual values for degree 2 on the validation set.

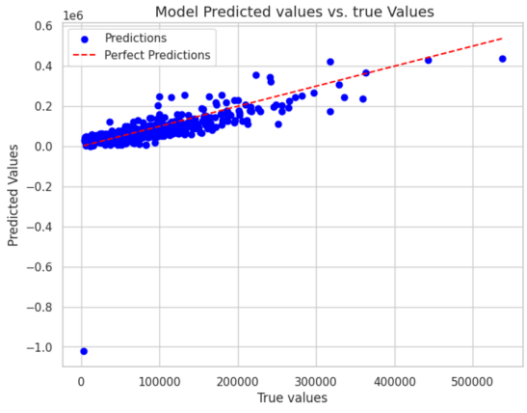

Figure 6- Predicted values versus actual values with degree 2 polynomial model

Radial basis function (RBF)

When performing non-linear regression, our high-level goal is to solve for the optimal linear combination of a collection of basis functions that allows us to represent something non-linear.

For RBF regression, a collection of Gaussians is utilized.

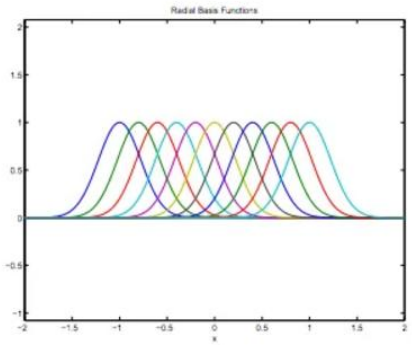

Figure 7- Sample Radial basis functions

For RBF, the following kernel is used to transform the input:

$$b(x) = e^{\frac{-(x-c_k)^2}{2\sigma^2}}$$

where σ is the gaussian's standard deviation and c_k is its center or mean. These are model's hyperparameters.

RBF model is implemented using Scikit Learn machine-learning library. Using class sklearn.svm.SVR [Support vector regression].

One popular kernel function in SVMs is the RBF kernel. In order to identify a hyperplane that divides the data points, the input data is transformed into a higher-dimensional space.

Hyperparameter Tuning with Grid Search

C parameter: The C parameter is a hyperparameter in support vector machines (SVMs) that establishes the trade-off between attaining a low testing error and a low training error.

Gamma parameter: In SVMs, the gamma parameter is a hyperparameter that regulates the decision boundary's form. It establishes the degree of overfitting or underfitting of the training data as well as the model's flexibility.

Grid search is used to find the optimal values for these hyperparameters that improve the performance measures.

The values tried in the grid search are :

parameters = {'C':[0.001, 0.01, .01, 1, 10, 100, 1000,10000,100000],
'gamma': [0.001, 0.001, 0.01, 1, 10, 100, 1000]}

The best values are: C= 100000 and gamma = 0.01

After that, the model is evaluated on the validation set to be compared with the other models in model selection process. the performance metrics values are:

Mean Squared error	Mean Absolute Error	R-squared
406718814.88	11850.87	0.88

Table 4- Performance measures for RBF regression

The performance measures (MSE and MAE) are large because the target variable was neither scaled nor normalized.

The following figure shows the predicted values using polynomial with RBF regression versus the actual values.

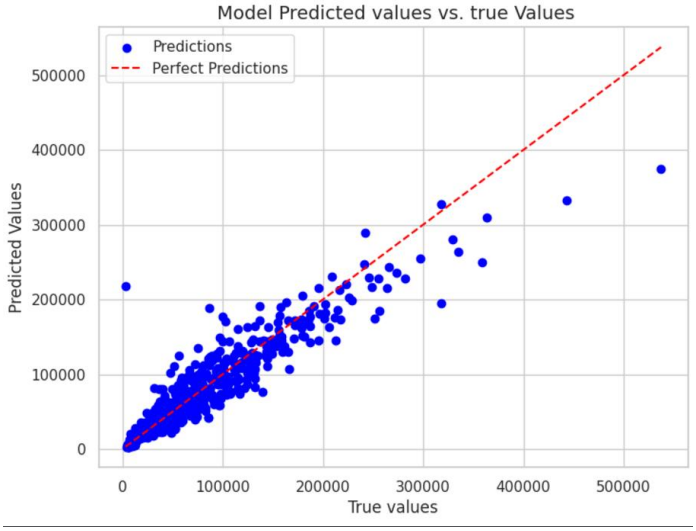

Figure 8- Predicted values VS actual values for RBF

Feature extraction

Summary Table

Model Name	MSE	R2	MAE
Linear Models			
Simple Linear Model	1194863124	0.65	26655
Closed Form Linear Model	1194863124	0.65	26655
Gradient Descent Linear Model	1194863124	0.65	26655
Ridge Linear Model	764542685	0.78	18752
LASSO Linear Model	1120196809	0.67	25547
Non-Linear Models			
RBF (Standard Gaussian Kernel)	404463873	0.88	11834
Polynomial of degree 2	2294483370	**0.33**	21835
Polynomial of degree 3	174990061616	-50.16	29205
Polynomial of degree 4	10171657663960	-2972.99	119023
Polynomial of degree 5	542159076289135	-158515.41	780671
Polynomial of degree 6	2.56+16	-7478532.59	5275388

Figure 9-Evaluation of Models with Validation Set

After evalauting different models on the validation set, using the higher R^2 score we conlcuded that the RBF model best interperts and generalize the dataset. So we chose it as a starting point and applied the feature extraction on it.

The idea of feature extraction is using the forward selection we want to choose a subset of predictors through *iteratively* building an optimal subset of predictors by optimizing a fixed model evaluation metric each time.

We started with an empty set then looped to all features, for each we looped again for all the un-added features and we kept adding them to the subset and buildingand evaluating the model, as long as the R^2 score kept increasing. We also made sure to have a set of more than 2 to be able to correctly vapture complex features. For the RBF model (we built it also incoprainting ridge regularization technique here reather than simple linear model), we were able to extrac the following features:

{'country_egypt', 'country_qatar', 'country_kuwait', 'country_uae', 'country_ksa', 'brand', 'horse_power', 'seats', 'engine_capacity', 'cylinder', , 'cylinder'}

Thus we were able to decrease dimensionality to from 12 to 10 while retaining a high R^2 score of 0.88.

Reporting Final Performance

To report final performance we used the test set to evaluate and predict using the RBF- feature extracted model and the results were as follows:

Mean Squared Error: 2759327298.0359077

R^2 Score: 0.6784635877572609

Mean Absolute Error: 13761.175712056913

We have a value R^2 score of 67% meaning our model can genralize up to 67%. Below is a sample of the predicted values:

Ford Expedition EL 2021 3.5L EcoBoost LTD	58384	50993.03775
Jaguar F-Pace 2021 2.0T Portfolio (250 PS)	61745	47453.39239
Audi A4 2021 40 TFSI Sport (190 HP)	43160	39463.74321
Bentley Continental GT Convertible 2021 W12	286000	322348.2541
Bentley Continental GT Convertible 2021 V8	262400	252008.2844
Geely Azkarra 2021 1.5TD Comfort (2WD)	18860	33678.08473
Kia Sorento 2021 3.3L Base (FWD)	25480	40319.55013
Toyota Hilux 2021 2.7L Double Cab GL M/T (4x4)	23616	21814.26158
Jeep Cherokee 2021 3.2L Limited	33750	37577.88886
Porsche Panamera 2021 4 E-Hybrid	125624	136341.4468
Maserati Quattroporte 2021 3.0T V6 S GranLusso	132512	125703.1522
Mercedes-Benz SL-Class 2021 SL 500	140400	141120.6453
Bentley Bentayga 2021 4.0L V8	229500	259580.4731
Ford Figo 2021 1.5L Trend H/B	15088	17170.82787
Ford Figo 2021 1.5L Trend H/B	17160	18214.77937
Audi A4 2021 40 TFSI S-line Plus (190 HP)	19530	20388.2156
Mitsubishi Pajero 2021 3.8L GLS 5 Door high	32250	30464.68051
Mitsubishi Pajero 2021 3.8L GLS 5 Door high	35100	37700.76874
Audi A5 Coupe 2021 45 TFSI quattro Sport (252 HP)	57200	51533.97017
BMW 2 Series Convertible 2021 228i	52250	53504.18257
Toyota 86 2021 VT	29520	31573.286
Nissan Altima 2021 2.5 SL	34147.71	35719.36487
Lexus CT 2021 200h Platinum	46800	26683.63466
Maserati Quattroporte 2021 3.0T V6 S	137800	125698.2784

Figure 10: Sample Predicted Values

Conclusion

In this assignment, different regression models are successfully implemented to predict the car prices using the provided dataset. Linear regression is a simple baseline model. Lasso Regression improves upon the linear regression idea and prevent overfitting. Ridge regression is also a regularization technique used for solving overfitting and multicollinearity issues. Nonlinear regression keeps the math of linear regression, but extend to more general functions. While higher-degree polynomials overfit the training data, degree 2 polynomial regression showed the best overfitting performance. Using Gaussian kernels, RBF Regression effectively represented non-linear relationships. and it was our code's best mode. It provides the validation set's highest R-squared score. Grid Search is applied to find the best hyperparameters for each model (e.g., λ for regularized models). Feature selection is used to select the best features that improve the model performance.