Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Machine Learning and Data Science - ENCS5341

Assignment #2

Prepared by: Ibaa Taleeb **ID** : 1203073

Instructor : Dr.Ismail Khater

Section: 2

Date: 22/11/2024

Abstract

This project develops and evaluates regression models to predict car prices using a dataset from Yalla Motors. I explore various models, including linear regression, Lasso, Ridge, polynomial regression, and Gaussian kernel methods, aiming to capture both linear and nonlinear relationships between car features and prices. Forward selection and regularization techniques enhance model simplicity and prevent overfitting by prioritizing relevant features. Grid search is used to optimize hyperparameters, allowing for selection of the best model based on validation metrics such as Mean Squared Error (MSE) and R-squared. Finally, the chosen model is evaluated on a test set to assess its generalization to unseen data, demonstrating the model's robustness in predictive performance. This analysis provides a systematic approach for selecting and tuning machine learning models for predictive accuracy and reliability.

Table of Contents

Table of Figures

Data Cleaning and Feature Engineering:

In this part we will clean the dataset by handling missing values, standardize all prices to a common currency (USD), encoding categorical features, normalizing and standardizing numerical features, these are crucial steps in data preprocessing to ensure the quality and reliability of analysis and model.

1. standardize all prices to a common currency: in the code, we define a dictionary with conversion rates from several currencies to USD, then strip out the currency symbol and any commas, then converts the numeric value to USD using the corresponding rate and return [NaN] if the currency is not unique.

	car name	engine_capacity	cylinder	horse_power	top_speed	seats	brand	country	price
0	Fiat 500e 2021 La Prima	0.0	N/A, Electric	Single	Automatic	150	fiat	ksa	92679.062041
1	Peugeot Traveller 2021 L3 VIP	2.0	4	180	8 Seater	8.8	peugeot	ksa	37955.250000
2	Suzuki Jimny 2021 1.5L Automatic	1.5	4	102	145	4 Seater	suzuki	ksa	26671.950000
3	Ford Bronco 2021 2.3T Big Bend	2.3	4	420	4 Seater	7.5	ford	ksa	92679.062041
4	Honda HR-V 2021 1.8 i-VTEC LX	1.8	4	140	190	5 Seater	honda	ksa	92679.062041

Figure 1: standardize all prices to a common currency

2.Missing Value Strategies: we can apply multiple strategies (mean/median imputation, dropping rows) to fill the missing values, columns with low missing values then drop rows and using .mean() , .median() to calculate the mean and median to fill the missing values, check the null values so we will get 0 null value.

At first, we convert the engine_capacity, cylinder, horse_power, and top_speed columns to numeric values, coercing any non-numeric values to [NaN] using pd.to_numeric(). to ensure that these columns contain only valid numerical data. Then, the seats column is cleaned by replacing common non-numeric values with NaN and extracts the numeric part of the seats column, converting it to a float for ensures that the seats column contains only numerical values representing the number of seats.

After that we fill the missing values using fill_with_mode function, which replaces missing values (NaN) in the dataset with the most frequent value (mode) within each group. The groups are formed based on the car name and brand columns, ensuring that missing values are filled with the most common value for each specific car or brand. This was done by applied separately to each group, first by car name and then by brand. After that drop rows with null values.

	car name	engine_capacity	cylinder	horse_power	top_speed	seats	brand	country	price
0	Fiat 500e 2021 La Prima	0.0	4.0	100.0	182.0	150.0	fiat	ksa	21200.00
1	Peugeot Traveller 2021 L3 VIP	2.0	4.0	180.0	205.0	8.0	peugeot	ksa	37955.25
2	Suzuki Jimny 2021 1.5L Automatic	1.5	4.0	102.0	145.0	4.0	suzuki	ksa	26671.95
3	Ford Bronco 2021 2.3T Big Bend	2.3	4.0	420.0	160.0	7.0	ford	ksa	52135.65
4	Honda HR-V 2021 1.8 i-VTEC LX	1.8	4.0	140.0	190.0	5.0	honda	ksa	25740.45

Figure 2:dataset after cleaning and filling missing values

Figure 3 shows that the dropped rows equal 58, the reduction in rows (58 rows) is relatively small (0.009 wich is less than 1% of the total dataset), so the impact on overall analysis or modeling will not be affected.

```
Number of rows befor handling errors and missing values: 6308
Number of rows after handling errors and missing values: 6250
```

Figure 3:Number of rows in the dataset

Figure 2 below shows the dataset columns after cleaning and handle missing values.

```
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   car name        6308 non-null   object 
 1   engine_capacity  6308 non-null   float64
 2   cylinder        6308 non-null   float64
 3   horse_power     6308 non-null   float64
 4   top_speed       6308 non-null   float64
 5   seats           6308 non-null   float64
 6   brand           6308 non-null   object 
 7   country         6308 non-null   object 
 8   price           6308 non-null   float64
dtypes: float64(6), object(3)
memory usage: 443.7+ KB
```

Figure 4: Convert to numerical and handle missing values

For example, if we search about fiat in brand column, we can see all car names information from brand fiat look at figure 5,before cleaning the dataset we can see that Fiat 500e 2021 La Prima information are null values as shown in figure 6,after cleaning and fill using car name and brand name,we can see that missing values are filled with the most common value for each specific car or brand as shown in figure 7.

	car name	price	engine_capacity	cylinder	horse_power	top_speed	seats	brand	country	price_usd
0	Fiat 500e 2021 La Prima	TBD	0.0	N/A, Electric	Single	Automatic	150	fiat	ksa	NaN
298	Fiat 500 2021 1.4L Hatch POP	SAR 71,185	1.4	4	100	182	4 Seater	fiat	ksa	19219.95
336	Fiat 500X 2021 1.4T City Cross	SAR 66,500	1.4	4	140	190	5 Seater	fiat	ksa	17955.00
357	Fiat 500 2021 Convertible 1.4L Pop	SAR 82,685	1.4	NaN	100	182	4 Seater	fiat	ksa	22324.95
1008	Fiat 500e 2021 La Prima	TBD	Cylinders	Drive Type	Horsepower (bhp)	Top Speed (Km/h)	Seating Capacity	fiat	egypt	NaN

Figure 5:Fiat brand

	car name	price	engine_capacity	cylinder	horse_power	top_speed	seats	brand	country	price_usd
0	Fiat 500e 2021 La Prima	TBD	0.0	N/A, Electric	Single	Automatic	150	fiat	ksa	NaN
1008	Fiat 500e 2021 La Prima	TBD	Cylinders	Drive Type	Horsepower (bhp)	Top Speed (Km/h)	Seating Capacity	fiat	egypt	NaN
1394	Fiat 500e 2021 La Prima	TBD	0.0	N/A, Electric	Single	Automatic	150	fiat	bahrain	NaN
2302	Fiat 500e 2021 La Prima	TBD	0.0	N/A, Electric	Single	Automatic	150	fiat	qatar	NaN
3219	Fiat 500e 2021 La Prima	TBD	0.0	N/A, Electric	Single	Automatic	150	fiat	oman	NaN

Figure 6:car name "Fiat 500e 2021 La Prima" befor cleaning

	car name	engine_capacity	cylinder	horse_power	top_speed	seats	brand	country	price
0	Fiat 500e 2021 La Prima	0.0	4.0	100.0	182.0	150.0	fiat	ksa	21200.0
1008	Fiat 500e 2021 La Prima	0.0	4.0	100.0	182.0	150.0	fiat	egypt	21200.0
1394	Fiat 500e 2021 La Prima	0.0	4.0	100.0	182.0	150.0	fiat	bahrain	21200.0
2302	Fiat 500e 2021 La Prima	0.0	4.0	100.0	182.0	150.0	fiat	qatar	21200.0
3219	Fiat 500e 2021 La Prima	0.0	4.0	100.0	182.0	150.0	fiat	oman	21200.0

Figure 7: car name "Fiat 500e 2021 La Prima" after cleaning

3.Feature Encoding: using label Encoding to encode categorical features (car name, brand and country) into a numerical format suitable for analysis, making it easier to process and analyze these categorical features in the model. The figure 8 below shows the encoded columns, for example the brand 'audi' was encoded to become 3.

	engine_capacity	cylinder	horse_power	top_speed	seats	price	car_name_encoded	country_encoded	brand_encoded
0	0.0	4.0	100.0	182.0	150.0	21200.00	548	2	21
1	2.0	4.0	180.0	205.0	8.0	37955.25	1953	2	55
2	1.5	4.0	102.0	145.0	4.0	26671.95	2206	2	65
3	2.3	4.0	420.0	160.0	7.0	52135.65	558	2	22
4	1.8	4.0	140.0	190.0	5.0	25740.45	795	2	29

Figure 8:dataset after encoding

4. Normalization : applying Min-Max scaling the data becomes consistent and suitable for models that may be sensitive to feature scaling, improving the analysis accuracy and performance. As we can see in the below figure that the values between 0 and 1.

$$v' = \frac{v - min_F}{max_F - min_F}(new\_max_F - new\_min_F) + new\_min_F$$

	engine_capacity	cylinder	horse_power	top_speed	seats	price	car_name_encoded	country_encoded	brand_encoded
0	0.000000	0.076923	0.017467	0.091176	0.573643	0.004621	0.218762	0.333333	0.295775
1	0.000296	0.076923	0.033347	0.125000	0.023256	0.009288	0.779641	0.333333	0.774648
2	0.000222	0.076923	0.017864	0.036765	0.007752	0.006145	0.880639	0.333333	0.915493
3	0.000341	0.076923	0.080985	0.058824	0.019380	0.013237	0.222754	0.333333	0.309859
4	0.000267	0.076923	0.025407	0.102941	0.011628	0.005886	0.317365	0.333333	0.408451

Figure 9: Normalization the numerical features

5.Standrization: we applied the StandardScaler() to standardize the numerical columns in the dataset. Standardization involves transforming the features, so that they have a mean of 0 and a standard deviation of 1. We standardized the data because many machine learning algorithms assume features are on a similar scale. So, without standardization, features with larger numerical ranges can dominate the model, leading to poor performance or slow convergence, especially for gradient-based algorithms. By standardizing the data, the model can treat all features equally, improving performance and ensuring that the optimization process runs more efficiently. The standardization is done by using the below equation.

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

	engine_capacity	cylinder	horse_power	top_speed	seats	price	car_name_encoded	country_encoded	brand_encoded
0	-0.231457	-0.710244	-1.070244	-0.845235	6.523206	-0.437432	-0.928564	-0.647444	-0.777183
1	-0.227787	-0.710244	-0.623176	-0.353877	0.019975	-0.275611	1.035795	-0.647444	0.905306
2	-0.228704	-0.710244	-1.059067	-1.635681	-0.163215	-0.384584	1.389520	-0.647444	1.400156
3	-0.227236	-0.710244	0.718028	-1.315230	-0.025822	-0.138657	-0.914583	-0.647444	-0.727698
4	-0.228154	-0.710244	-0.846710	-0.674328	-0.117417	-0.393580	-0.583228	-0.647444	-0.381304

Figure 10:Standerized dataset

Building Regression Models:

6.Linear Models:

This part evaluates linear regression models (Linear, Lasso, and Ridge) to predict car prices based on cleaned dataset. First, splitting the data into training (60%), validation (20%), and test sets (20%). Each model is initially trained and evaluated on the training and validation sets, with Lasso and Ridge, also applying regularization to reduce overfitting. A Grid Search (GS) is conducted to find the optimal regularization parameters (alpha) for Lasso and Ridge, figure 11 show the best alpha parameter, minimizing mean squared error through cross-validation. Finally, the models with optimized alpha values are re-trained and evaluated, providing improved accuracy for predicting car prices by reducing irrelevant feature influence. Figures 11 show the Validation Performance with Optimized Alpha.

```
Best lambda for LASSO: 0.01
Best lambda for Ridge: 100

Validation Metrics:
LASSO: MSE = 0.24085863173419397, MAE = 0.278896699706039
Ridge: MSE = 0.24178397978124627, MAE = 0.28062844231483725
Linear Regression: MSE = 0.2425109579750796, MAE = 0.2833959665503344

Test Metrics:
LASSO: MSE = 0.3686404041822717, MAE = 0.276759687436954
Ridge: MSE = 0.3705122647011442, MAE = 0.2790877285088804
Linear Regression: MSE = 0.36925471458111087, MAE = 0.28199119435111775

Validation R² Scores:
LASSO: 0.5824644723631667
Ridge: 0.5808603542865511
Linear Regression: 0.5796001161893853

Test R² Scores:
LASSO: 0.5129445729568234
Ridge: 0.5104714316135486
Linear Regression: 0.5121329331847062
```

Figure 11: Linear Regression models performance

From the above results, we can see that LASSO model appears to be the most effective model, particularly for generalization on unseen data, as it achieves the best balance between minimizing error and explaining variance in both validation and test sets. We can see also that Ridge provides a slightly more regularized model, which may be beneficial in cases with highly correlated features, but its performance on this dataset is very similar to that of Linear Regression, both of which have slightly higher error metrics and lower R^2 scores compared to LASSO.

7. Use the closed-form solution: The closed-form solution for linear regression (the Normal Equation) allows us to compute the optimal model parameters without iteration, unlike gradient descent. We can use this approach to find the weight vector θ that minimizes the cost function J(θ) for a linear regression model. Solution can be computed by using the below equation

$$\theta = (X^T X)^{-1} X^T y$$

In this part , the code implement and compares two approaches for solving linear regression: the Closed-Form Solution and Gradient Descent, using a subset of features determined by a forward feature selection process. The Closed-Form Solution computes the optimal weights directly using linear algebra (via the normal equation), while Gradient Descent iteratively optimizes the weights by minimizing the Mean Squared Error (MSE). Both methods are applied to the same selected features, and their performance is evaluated using metrics such as R^2, MSE, and Mean Absolute Error (MAE) on the validation set. Feature selection ensures the model only includes predictors that significantly improve performance. The comparison reveals that both methods yield nearly identical results, confirming the theoretical equivalence of the two approaches for linear regression under ideal conditions. This implementation highlights the computational efficiency of the closed-form solution for small datasets and the scalability of gradient descent for larger, high-dimensional problems.

The results in figure 13 shows that both methods are nearly identical. Since both approaches aim to minimize the same linear regression objective function. The Closed-Form Solution directly computes the optimal weights, while Gradient Descent iteratively converges to those weights. The small numerical differences arise from the iterative nature of gradient descent and potential rounding errors during matrix operations.

The selected features' indices are [2, 3, 1, 6, 4, 0]. This means the most relevant features in predicting the target variable (based on validation MSE) include

```
Closed-Form Solution (Validation):
R²: 0.5796001161893852, MSE: 0.24251095797507966, MAE: 0.28339596655033455
Gradient Descent Solution (Validation):
R²: 0.5795666794849605, MSE: 0.24253024619930103, MAE: 0.2833902428096397
Selected Features (Indices): [2, 3, 1, 6, 4, 0]
```

Figure 12:closed-form solution results

8. Nonlinear Models:

Polynomial Regression (Degree 2 to Degree 10): we implement and evaluate polynomial regression models for various degrees (from 2 to 10) to assess how increasing polynomial complexity affects the model's performance. It transforms the training and validation data into polynomial features of the specified degree, fits a linear regression model to the transformed training data, and makes predictions on the validation set. For each degree, we calculated performance metrics: R^2 (coefficient of determination), Mean Squared Error (MSE), and Mean Absolute Error (MAE). These are stored along with the corresponding model to compare performance across polynomial degrees.

```
Polynomial Regression Metrics on Validation Set:
Degree 2: R² = 0.4947, MSE = 0.2915, MAE = 0.3170
Degree 3: R² = 0.6386, MSE = 0.2085, MAE = 0.2260
Degree 4: R² = -1011.1207, MSE = 583.8497, MAE = 1.9267
Degree 5: R² = -77498822.0797, MSE = 44705801.6683, MAE = 435.0272
Degree 6: R² = -6819106128998937.0000, MSE = 3933654654914809.0000, MAE = 2399818.9273
Degree 7: R² = -310558971830259.5000, MSE = 179148369603832.5000, MAE = 712515.8150
Degree 8: R² = -8401931992788.2900, MSE = 4846720122621.8652, MAE = 92937.1230
Degree 9: R² = -24767982452321.7930, MSE = 14287604214297.6055, MAE = 134243.3507
Degree 10: R² = -35845241079422.8438, MSE = 20677607410887.8398, MAE = 136143.5246
```

Figure 13: Polynomial Regression (Degree 2 to Degree 10)

According to the result showed above, we can see that Polynomial Regression with degree 3 provides the optimal balance between bias and variance, resulting in the best generalization to validation data. Models with degrees greater than 3 are severe overfitting.

Gaussian Kernel (RBF) Regression (Alpha 0.1 to Alpha 100):in this part, we implement a Support Vector Regression (SVR) model using the Radial Basis Function (RBF) kernel to predict price. The "rbf_regression" function used to train the SVR model on the training data and evaluates its performance on the validation set by calculating three key metrics: R^2, Mean Squared Error (MSE), and Mean Absolute Error (MAE).

The results in figure 15 below show that RBF Regression model on the validation set demonstrate good performance. Where R^2 score = 0.675 indicates that the model explains about 67.5% of the variance in price, MSE (0.188), and the MAE (0.190), both of MSE and MAE are low. This means that the RBF SVR model performs well in predicting the price, likely benefiting from its ability to model nonlinear relationships effectively.

```
RBF Regression R² Score on Validation Set: 0.6747757783870858
RBF Regression MSE on Validation Set: 0.18760813353501693
RBF Regression MAE on Validation Set: 0.1904737796567862
```

Figure 14:Gaussian Kernel (RBF) Regression

9. Model Selection Using Validation Set

Based on the validation metrics shown in figure 16 below, the RBF Regression model demonstrates the best performance among all models. It achieves the highest R^2 score (0.675), indicating it explains the largest proportion of variance in the validation data. It also has the lowest MSE (0.188) and MAE (0.190), reflecting lower prediction errors compared to the linear regression, LASSO, Ridge, and polynomial regression models.

The linear models (LASSO, Ridge, and basic Linear Regression) have similar performances, with R^2 scores around 0.58 and slightly higher MSE and MAE values. Among these, LASSO marginally outperforms Ridge and Linear Regression.

Polynomial regression shows inconsistent performance, with a peak R^2 score of 0.639 for degree 3. However, higher degrees lead to extremely poor results due to overfitting, evident from the negative R^2 scores and massive MSE values.

```
Validation Metrics:
LASSO: MSE = 0.24085863173419397, MAE = 0.278896699706039
Ridge: MSE = 0.24178397978124627, MAE = 0.28062844231483725
Linear Regression: MSE = 0.2425109579750796, MAE = 0.2833959665503344

Test Metrics:
LASSO: MSE = 0.3686404041822717, MAE = 0.276759687436954
Ridge: MSE = 0.3705122647011442, MAE = 0.2790877285088804
Linear Regression: MSE = 0.36925471458111087, MAE = 0.28199119435111775

Validation R² Scores:
LASSO: 0.5824644723631667
Ridge: 0.5808603542865511
Linear Regression: 0.5796001161893853

Test R² Scores:
LASSO: 0.5129445729568234
Ridge: 0.5104714316135486
Linear Regression: 0.5121329331847062
```

```
Polynomial Regression Metrics on Validation Set:
Degree 2: R² = 0.4947, MSE = 0.2915, MAE = 0.3170
Degree 3: R² = 0.6386, MSE = 0.2085, MAE = 0.2260
Degree 4: R² = -1011.1207, MSE = 583.8497, MAE = 1.9267
Degree 5: R² = -77498822.0797, MSE = 44705801.6683, MAE = 435.0272
Degree 6: R² = -6819106128998937.0000, MSE = 3933654654914809.0000, MAE = 2399818.9273
Degree 7: R² = -310558971830259.5000, MSE = 179148369603832.5000, MAE = 712515.8150
Degree 8: R² = -8401931992788.2900, MSE = 4846720122621.8652, MAE = 92937.1230
Degree 9: R² = -24767982452321.7930, MSE = 14287604214297.6055, MAE = 134243.3507
Degree 10: R² = -35845241079422.8438, MSE = 20677607410887.8398, MAE = 136143.5246
```

```
RBF Regression R² Score on Validation Set: 0.6747757783870858
RBF Regression MSE on Validation Set: 0.18760813353501693
RBF Regression MAE on Validation Set: 0.1904737796567862
```

Figure 15: Model Selection Using Validation Set

10. Feature Selection with Forward Selection

We implement a forward feature selection process for regression using a Linear Regression model. Which it iteratively selects features from a set of candidate features that minimize the Mean Squared Error (MSE) on a validation set. First, Starting with an empty set of selected features, it evaluate the impact of adding each remaining feature, temporarily fitting a model with the new feature and calculating the validation MSE. The feature that results in the lowest MSE is added to the model, and this process repeats until no further improvement is observed or a specified maximum number of features is reached.

```
Selected Features (in order): ['horse_power', 'top_speed', 'cylinder', 'country_encoded', 'seats', 'engine_capacity']
Feature Scores (MSE at each step):
Feature: horse_power, Validation MSE: 0.2908143458370722
Feature: top_speed, Validation MSE: 0.2432926091700604
Feature: cylinder, Validation MSE: 0.24130810574077377
Feature: country_encoded, Validation MSE: 0.24089879224045357
Feature: seats, Validation MSE: 0.2406929924238634
Feature: engine_capacity, Validation MSE: 0.2406839058678388
```

Figure 16: Feature Selection with Forward Selection

The results show that the feture selection identified six features horse_power, top_speed, cylinder, country_encoded, seats, and engine_capacity as the most significant predictors based on their ability to minimize validation MSE at each step. Starting with horse_power, which had the largest impact on reducing MSE (from an initial high to 0.2908), subsequent features were added in order of their contribution to further reducing the error. Then when reach to add engine_capacity resulted in the lowest validation MSE (0.2407) and there is no further improvement is observed, indicating that these six features collectively provide the best predictive performance for the given model.

10 Hyperparameter Tuning with Grid Search

The hyperparameter tuning using grid search has successfully identified the optimal regularization strengths (alpha) for both Lasso and Ridge regression models. By exploring a range of alpha values, the grid search helps improve model performance by reducing overfitting and ensuring optimal bias-variance trade-off. The best alpha values were selected based on the lowest mean squared error (MSE) in cross-validation, which results in more robust models. The evaluation of the optimized models on the validation set shows improvements in model performance, with lower MSE and higher R^2 values, confirming that the hyperparameter tuning process effectively enhances the regression models.

```
Training Performance:
Ridge Regression - MSE: 0.00025, MAE: 0.0078, R2: 0.5985

Validation Performance:
Ridge Regression - MSE: 0.00019, MAE: 0.0078, R2: 0.2915
```

```
Degree 9:
  Validation MSE: 74294.04
  Validation MAE: 215.21
  Validation R²: 0.56
```

Figure 17:Hyperparameter Tuning with Grid Search & Polynomial Regression with degree 9

11. Model Evaluation on Test Set:

We selected the RBF Regression model as the best-performing model based on the validation set metrics, we evaluate its performance on the unseen test set to determine how well it generalizes to new data. The evaluation is conducted using key metrics: R^2, Squared Error (MSE), and Mean Absolute Error (MAE).

R^2 =0.662: This indicates that the model explains approximately 66.2% of the variance in the test data, which is consistent with its performance on the validation set (R^2 =0.675) The slight drop in R^2 reflects minimal overfitting, suggesting strong generalization.

MSE = 0.196: The mean squared error on the test set is marginally higher than on the validation set (MSE = 0.188), indicating that the model maintains low prediction error for unseen data.

MAE = 0.200: The mean absolute error remains close to the validation set performance (MAE= 0.190), further confirming the model's robustness.

In conclusion, The RBF Regression model demonstrates excellent generalization capability, as the test set metrics are closely aligned with the validation set metrics. This shows that the model effectively avoids overfitting while capturing the underlying patterns in the data.