Faculty of Engineering and Technology

Electrical and Computer Engineering Department

ENCS5341

MACHINE LEARNING AND DATA SCIENCE

Assignment2 - Report

Prepared by:

Manar Shawahni, 1201086.

Layan Abuershaid, 1200098.

Instructor's Name: Dr. Yazan Abu Farha

Section: 1

28 November, 2024

Contents

Description 1
1. Building Regression Models 2
1.1. Linear Models 2
1.1.1. Standard Linear Regression: 2
1.1.2. LASSO (L1 Regularization), Ridge Regression (L2 Regularization): 3
2. Closed-form solution and Gradient Descent 4
2.1.1. Closed-form solution 4
2.1.2. Linear Regression (Gradient Descent) 4
3. Nonlinear Models: 5
3.1.1. RBF Kernel Regression 5
3.1.2. Polynomial Regression 5
4. Initial Evaluation of Models Using the Validation Set 7
5. Feature Selection with Forward Selection 8
6. Applying Regularization Techniques 9
7. Hyperparameter Tuning with Grid Search 9
8. Find best model after enhance the models 10
9. Model Evaluation on Test Set 11
10. Choosing another feature as a target: 12
11. Conclusion 14

Description, including the Dataset, Preprocessing Steps, and Features Used

The dataset we used is from YallaMotors and has 6,310 entries car listings. It includes details about the car's name and brand, engine capacity, cylinder, horse power, top speed, seats, country and its price. The main objective is to predict car prices using the features such as cylinder count, top speed and seating capacity. Prices were in different currencies, so we had to convert them to USD to make them consistent. The data was not very clean, and we had to fix issues like very high or low values for horsepower and engine size to make the dataset more accurate for analysis.

To prepare the dataset for analysis we have done several preprocessing steps. Initially, for numerical features like engine capacity, horsepower, and top speed, we replaced outliers and non-numeric values with NaN, then filled missing values using the median to avoid the impact of extreme values and categorical features like brand and country were encoded using label encoding. Prices were converted to USD using exchange rates from an external API to ensure consistency. We used StandardScaler to standardize all features for better model performance. Finally, the dataset was split into training (60%), validation (20%), and test (20%) sets. The selected features included engine capacity, cylinders, horsepower, top speed, seats, brand, and country, which were used to predict the target, price_in_usd.

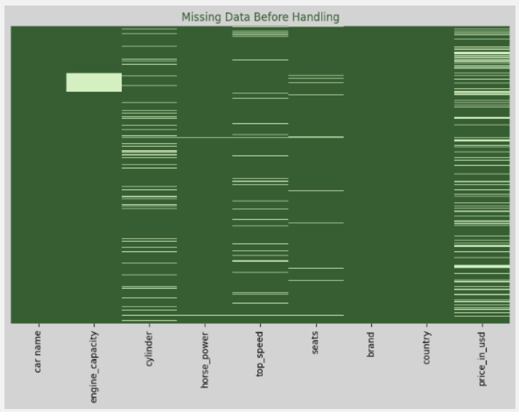


Figure 1 : Missing Data Before Handling.

Missing Data After Handling
car name
engine_capacity
cylinder
horse_power
top_speed
seats
brand
country
price_in_usd

Figure 2 : Missing Data After Handling.

Following the handling of missing data as visualized in figures above, the heatmaps before and after cleaning, it is clear that all missing values were successfully addressed.

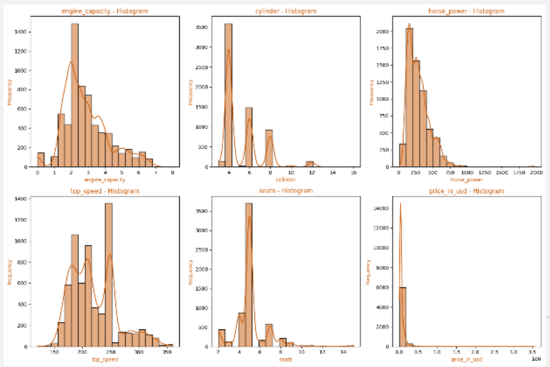


Figure 4 : Histograms.

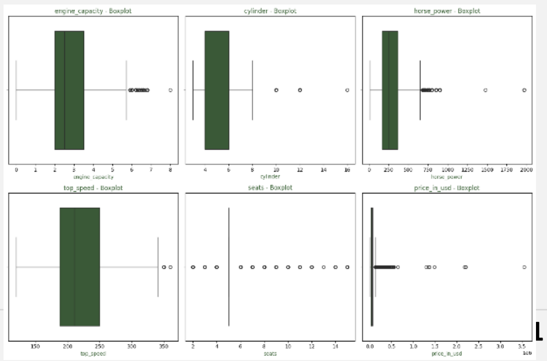
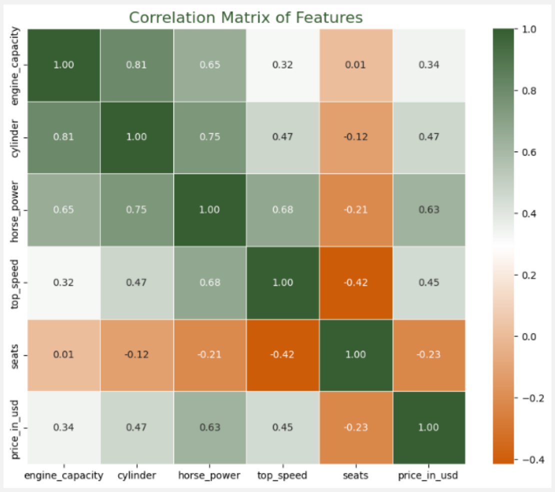


Figure 3 : Boxplots.



The histograms, boxplots and correlation matrix help to show the data distribution, detect outliers and understand the relationships between features which makes it easier to prepare the data for the model.

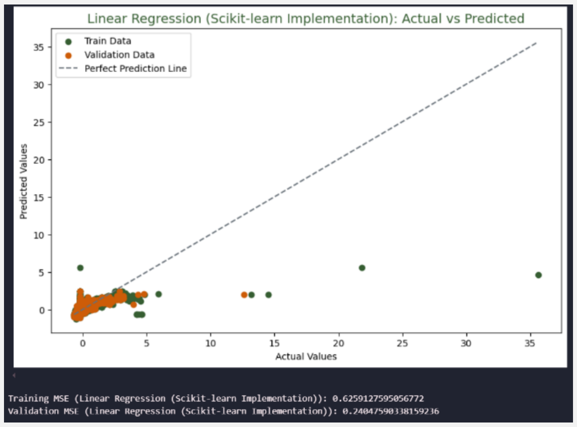
Figure 5 : Correlation Matrix of Features.

1. Building Regression Models

1.1. Linear Models

In this section, we implemented and evaluated three linear regression models: standard Linear Regression, Lasso Regression (L1 Regularization), and Ridge Regression (L2 Regularization). These models aim to establish a relationship between the features and target variable while addressing overfitting through regularization techniques. Each model was assessed using performance metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R²).

1.1.1. Standard Linear Regression:



```
Training MSE (Linear Regression (Scikit-learn Implementation)): 0.6259127595056772
Validation MSE (Linear Regression (Scikit-learn Implementation)): 0.24047590338159236
```

We implemented Standard Linear Regression using Scikit-learn's `LinearRegression` class to train the model and predict on both training and validation datasets.

Figure 6 : Linear Regression: Actual vs Predicted.

The Linear Regression model gave good results as shown the figure above, the training MSE was **0.625913144**, which shows the model learned the patterns in the training data but still had some error. The validation MSE was **0.24045803**, meaning the model worked better on unseen data and did not overfit. In the scatter plot above, most predictions were close to the perfect prediction line especially for smaller values. However, for higher values the model made some errors, so we need to try more advanced models or regularization to improve the performance, which we will do next.

1.1.2. LASSO (L1 Regularization), Ridge Regression (L2 Regularization):

For LASSO (L1 Regularization) and Ridge Regression (L2 Regularization), we used Scikit-learn's Lasso and Ridge classes to train the models with a regularization parameter (alpha) of 0.1.

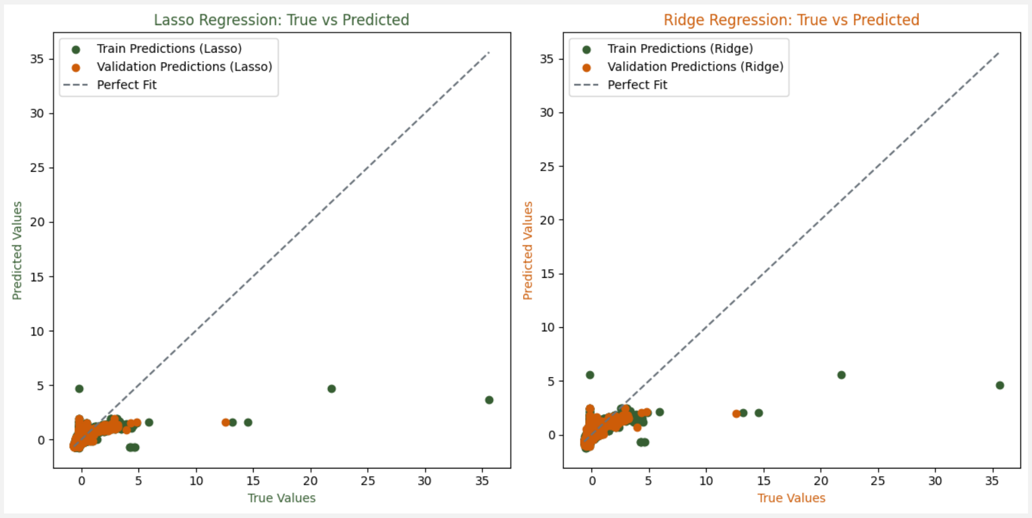
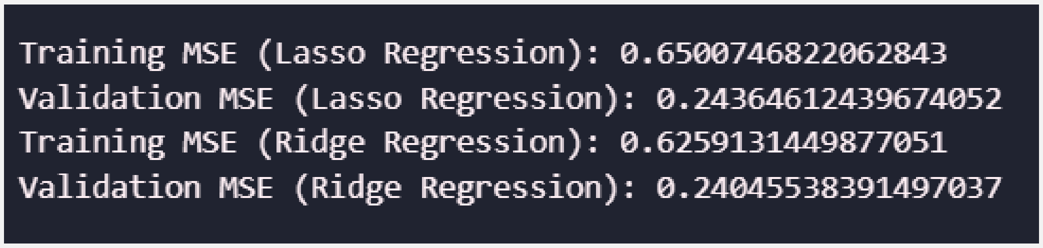


Figure 7: True vs Predicted.



```
Training MSE (Lasso Regression): 0.6500746822062843
Validation MSE (Lasso Regression): 0.24364612439674052
Training MSE (Ridge Regression): 0.6259131449877051
Validation MSE (Ridge Regression): 0.24045538391497037
```

Figure 8: Ridge Regression and LASSO MSE results.

The results above for both LASSO and Ridge Regression show how regularization helps improve the models and as above, the Ridge Regression performed better than LASSO with a lower training MSE (**0.6259131**) and validation MSE (**0.240553**), which are very close to the results of the other linear models. This means that Ridge works well while controlling large coefficients.

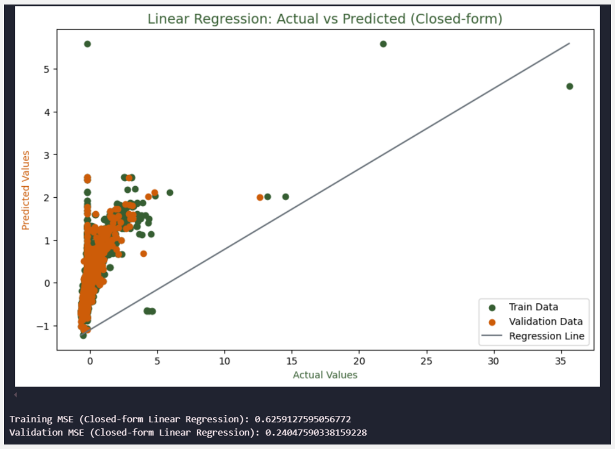
LASSO had higher errors, with a training MSE of **0.65007** and validation MSE of **0.243646** and this means LASSO didn't fit the data as well as Ridge. However, LASSO is useful because it can shrink some feature coefficients to zero, which helps in selecting the most important features. Overall, Ridge Regression gave better results in this case.

While these results show the benefits of regularization, but more optimization can enhance the models. In Part 5, we used Grid Search to find the best value for the regularization parameter (alpha) for LASSO and Ridge.

2. Closed-form solution and Gradient Descent

2.1.1. Closed-form solution

We used here matrix calculations to create the linear regression model without using any external libraries for fitting. This method finds the best parameters (θ) by solving the normal equation directly.



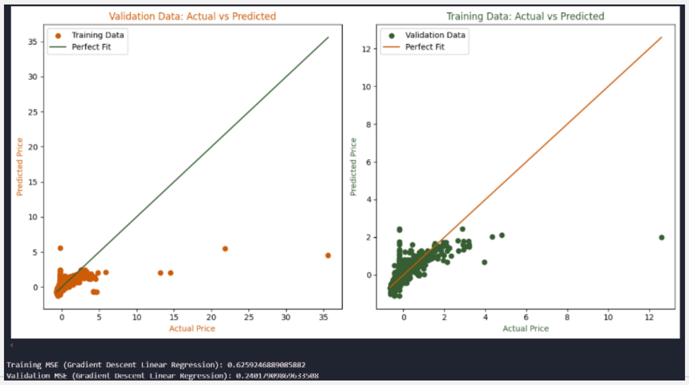
```
Training MSE (Closed-form Linear Regression): 0.6259127595056772
Validation MSE (Closed-form Linear Regression): 0.24047590338159228
```

Figure 9: Linear Regression: Actual vs Predicted (Closed-form).

The training MSE is 0.62591314, indicating that the model effectively fits the training data. The validation MSE is 0.240458, showing that the model generalizes well to unseen data without overfitting. The Actual vs Predicted plot shows that most predictions align well with the regression line for smaller values, while larger values show some deviation. This consistency with the previous implementation confirms the correctness of the Closed-form solution and its suitability for linear problems in this dataset. Further refinement or exploring non-linear models may still help reduce the errors for higher values. By comparing the Linear Model with the Closed-form solution we observed a very small difference, but the Closed-form solution shows better performance.

2.1.2. Linear Regression (Gradient Descent)

In this method we used an iterative way to find the best parameters (θ) for the linear regression model. Instead of solving equations directly, the model updates the parameters step by step using a learning rate and the gradient of the error.



```
Training MSE (Gradient Descent Linear Regression): 0.6259246889085882
Validation MSE (Gradient Descent Linear Regression): 0.240179009869633508
```

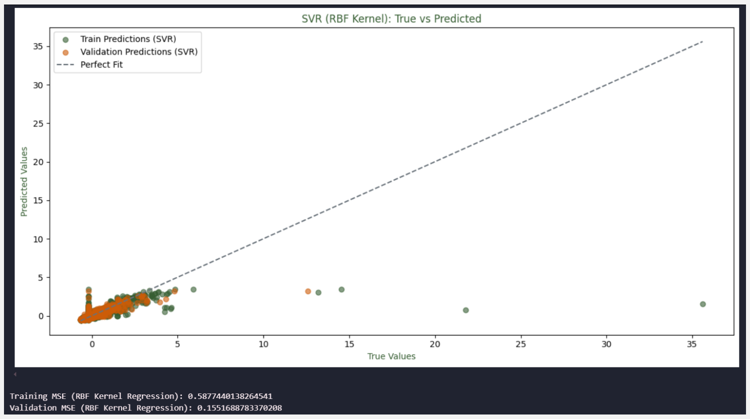
Figure 10 : Actual vs Predicted.

The training MSE was 0.625925and the validation MSE was 0.2401612, which are very close to the results of the other methods. And this shows that the model learned well from the training data and did not overfit when tested on new data. The Actual vs Predicted plots for both training and validation data show that most predictions align well with the perfect fit line, although there are still some errors for larger values. Overall, the Gradient Descent method worked effectively, and the small difference in MSE indicates consistent performance across all linear model implementations.

3. Nonlinear Models:

3.1.1. **RBF Kernel Regression** (using Support Vector Regression):

we used SVR with the RBF kernel to model nonlinear patterns in the data. The model was trained using the scikit-learn library with parameters C=1 and epsilon=0.1. After training, we made predictions for both the training and validation sets.



```
Training MSE (RBF Kernel Regression): 0.5877440138264541
Validation MSE (RBF Kernel Regression): 0.1551688783370208
```

Figure 11: SVR (RBF Kernel): True vs Predicted.

The results for the RBF Kernel Regression (SVR) show that it performed better than the linear models. The training MSE is 0.587787 which is lower than all the linear models, meaning it fits the training data better. The validation MSE is 0.15517 much lower than the other models, showing that it works well with new data.

The True vs Predicted plot also shows that most predictions are close to the perfect fit line, especially for smaller values. This suggests that the RBF Kernel can capture more complex nonlinear relationships in the data, making it a better choice for this dataset compared to linear models. However, there are still some errors for larger values, so more tuning might help.

3.1.2. **Polynomial Regression** (vary the polynomial degree from 2 to 10):

For Polynomial Regression, we explored degrees ranging from 2 to 10 using the scikit-learn library.

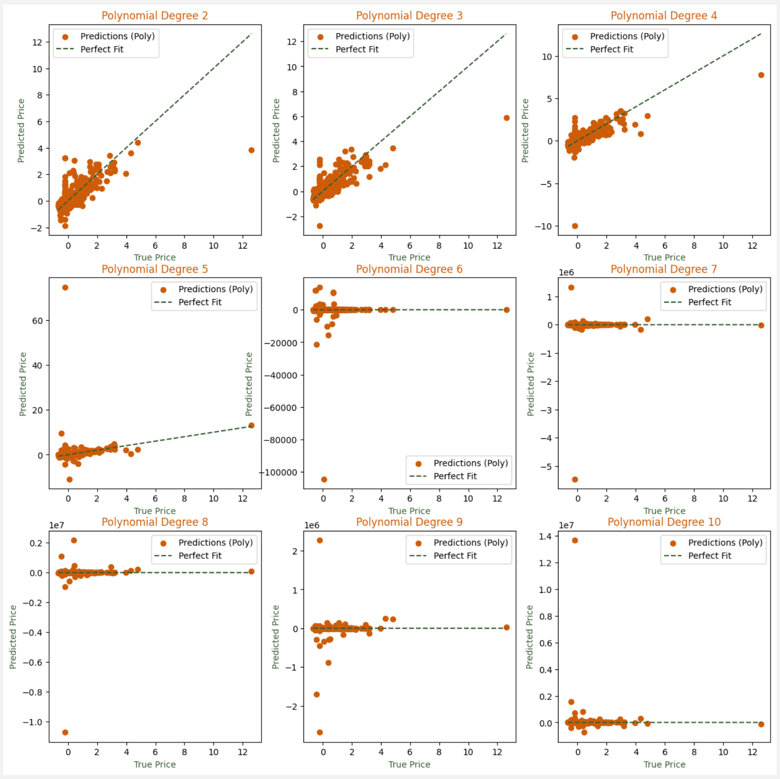
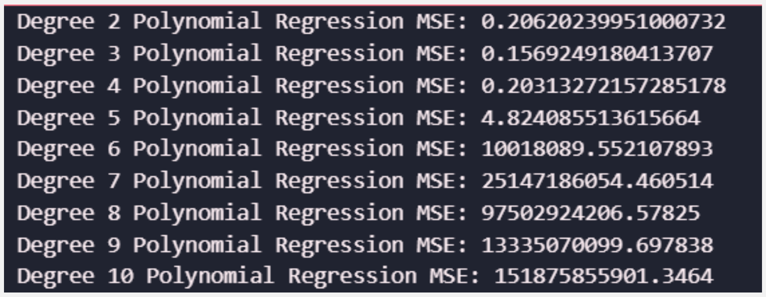


Figure 12: Polynomial Regression plots Results.



```
Degree 2 Polynomial Regression MSE: 0.20620239951000732
Degree 3 Polynomial Regression MSE: 0.1569249180413707
Degree 4 Polynomial Regression MSE: 0.20313272157285178
Degree 5 Polynomial Regression MSE: 4.824085513615664
Degree 6 Polynomial Regression MSE: 10018089.552107893
Degree 7 Polynomial Regression MSE: 25147186054.460514
Degree 8 Polynomial Regression MSE: 97502924206.57825
Degree 9 Polynomial Regression MSE: 13335070099.697838
Degree 10 Polynomial Regression MSE: 151875855901.3464
```

Figure 13: Polynomial Regression Results.

As shown in the figures above, the result:

- **Degree 2**: The MSE is **0.2062** shows a good fit prediction and it captures the data well without overfitting.
- **Degree 3**: The MSE drops further to **0.1569249**, indicating the best performance among all degrees.
- **Degree 4**: The MSE increases slightly to **0.2031327**, indicating the model starts to lose its generalization ability and starts overfitting.
- **Degree 5 and higher (5 to 10)**: The MSE increases rapidly, with Degree 6 and above having very high errors (**10,018,089**) and this happens because the models overfit the training data which make them perform badly on the validation set.

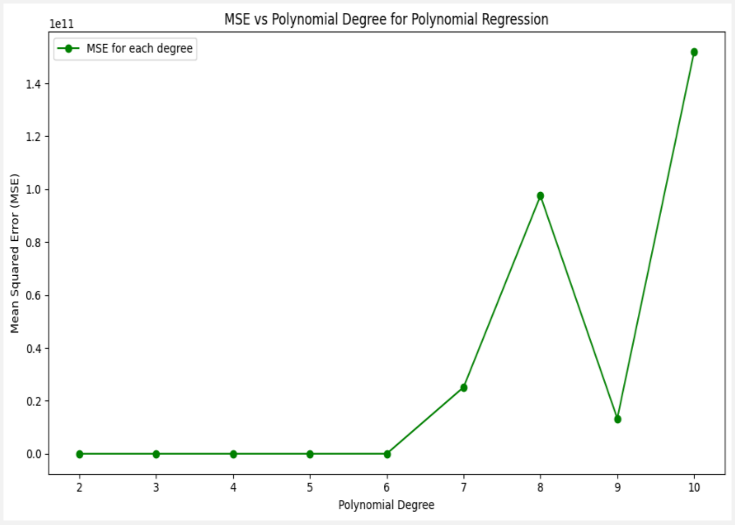
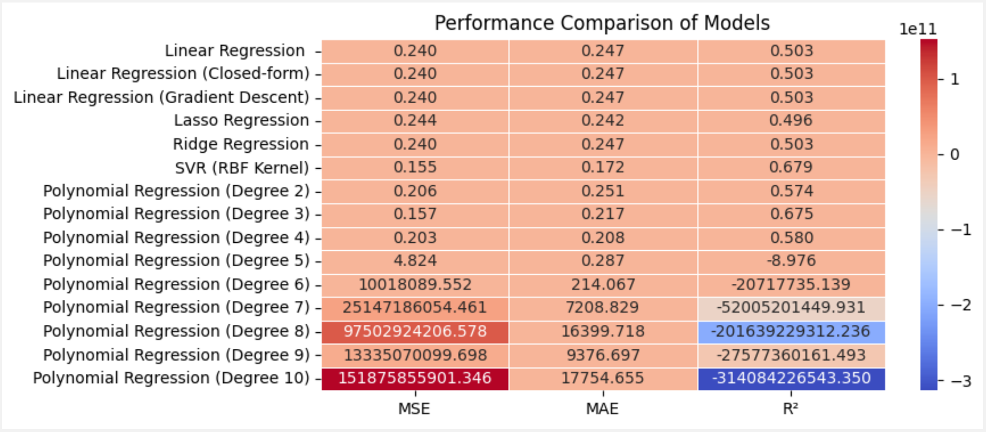


Figure 14 : MSE vs Polynomial Degree.

The graph shows that the MSE remains low for lower polynomial degrees (2-4) but increases sharply for higher degrees (5-10), showing overfitting as the model complexity grows.

To enhance the performance of both SVR and Polynomial Regression, we used Part 6 (Hyperparameter Tuning with Grid Search) to improve their performance.

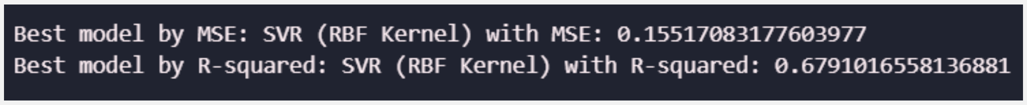
4. Initial Evaluation of Models Using the Validation Set



Performance Comparison of Models

	MSE	MAE	R²
Linear Regression	0.240	0.247	0.503
Linear Regression (Closed-form)	0.240	0.247	0.503
Linear Regression (Gradient Descent)	0.240	0.247	0.503
Lasso Regression	0.244	0.242	0.496
Ridge Regression	0.240	0.247	0.503
SVR (RBF Kernel)	0.155	0.172	0.679
Polynomial Regression (Degree 2)	0.206	0.251	0.574
Polynomial Regression (Degree 3)	0.157	0.217	0.675
Polynomial Regression (Degree 4)	0.203	0.208	0.580
Polynomial Regression (Degree 5)	4.824	0.287	-8.976
Polynomial Regression (Degree 6)	10018089.552	214.067	-20717735.139
Polynomial Regression (Degree 7)	25147186054.461	7208.829	-52005201449.931
Polynomial Regression (Degree 8)	97502924206.578	16399.718	-201639229312.236
Polynomial Regression (Degree 9)	13335070099.698	9376.697	-27577360161.493
Polynomial Regression (Degree 10)	151875855901.346	17754.655	-314084226543.350

Figure 15: Summarizing the Performance of Each Regression Model.



```
Best model by MSE: SVR (RBF Kernel) with MSE: 0.15517083177603977
Best model by R-squared: SVR (RBF Kernel) with R-squared: 0.6791016558136881
```

Figure 16: Best model by MSE.

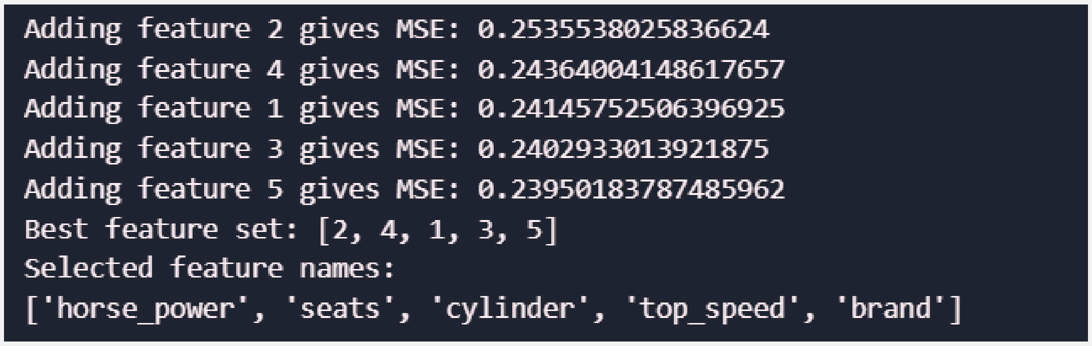
From the results above, the **SVR (RBF Kernel)** was the best model, with the lowest MSE (**0.155**) and highest R-squared (**0.679**), showing excellent handling of nonlinear relationships.

We noticed also that Linear models including Scikit-learn, Closed-form and Gradient Descent performed similarly with MSE ~ **0.240** and R-squared ~ **0.503** while Ridge slightly outperformed Lasso due to better regularization.

Polynomial Regression (Degree 3) also performed well, with MSE **0.157** and R-squared **0.675**, but higher degrees (5 and above) overfitted, resulting in poor generalization.

Overall, SVR was the best choice, followed by Polynomial Regression (Degree 3).

5. Feature Selection with Forward Selection



```
Adding feature 2 gives MSE: 0.2535538025836624
Adding feature 4 gives MSE: 0.24364004148617657
Adding feature 1 gives MSE: 0.24145752506396925
Adding feature 3 gives MSE: 0.2402933013921875
Adding feature 5 gives MSE: 0.23950183787485962
Best feature set: [2, 4, 1, 3, 5]
Selected feature names:
['horse_power', 'seats', 'cylinder', 'top_speed', 'brand']
```

Figure 17: Feature Selection with Forward Selection.

The Forward Feature Selection method improved the model step by step by adding the most important features, so that by adding Feature 2 first it resulted in an MSE of 0.25355. Next, adding Feature 4 reduced the MSE to 0.24364, followed by Feature 1 which further improved the MSE to 0.24146. Then, feature 3 and Feature 5 gave smaller but steady improvements and the final MSE became 0.2395 after all features were added.

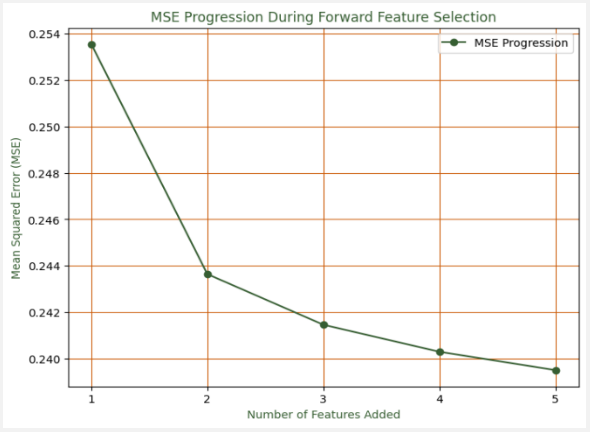


Figure 18 : MSE values vs number of features added.

The MSE progression plot shows that the biggest improvement happened with the first two features, while the improvements got smaller after adding more features.

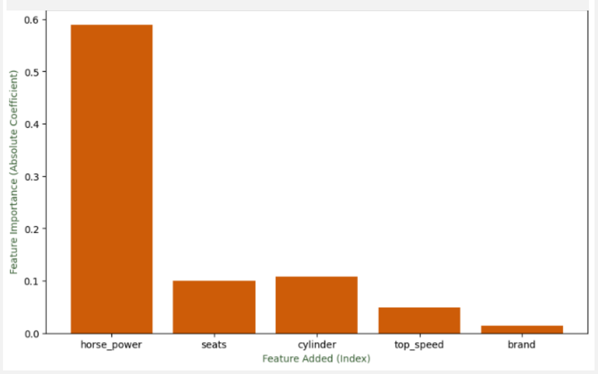
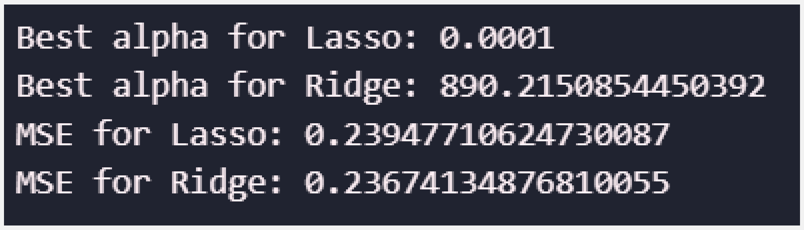


Figure 19 : feature importances with feature names.

The figure shows that horse_power had the biggest impact followed by Feature 1 and Feature 4. Feature 3 and Feature 5 had smaller effects. So, the selected feature set ['horse_power', 'seats', 'cylinder', 'top_speed', 'brand'] resulted in a model that is simple and performs well.

6. Applying Regularization Techniques



```
Best alpha for Lasso: 0.0001
Best alpha for Ridge: 890.2150854450392
MSE for Lasso: 0.23947710624730087
MSE for Ridge: 0.23674134876810055
```

Figure 20: Regularization Techniques results.

The results show that Ridge regression performed better than LASSO, with a lower MSE of 0.2367 compared to 0.2395. The best alpha values were 890.215 for Ridge and 0.0001 for LASSO which means Ridge needed stronger regularization while LASSO used very small regularization. Ridge worked better because it reduced all coefficients without removing any features, while LASSO may have removed some useful features by setting their coefficients to zero.

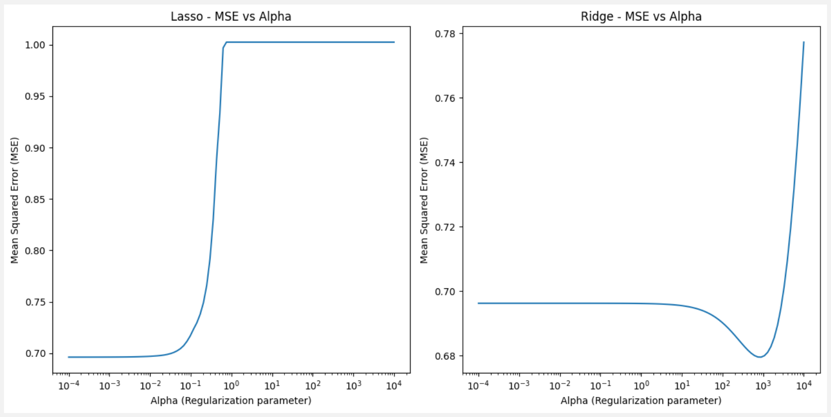
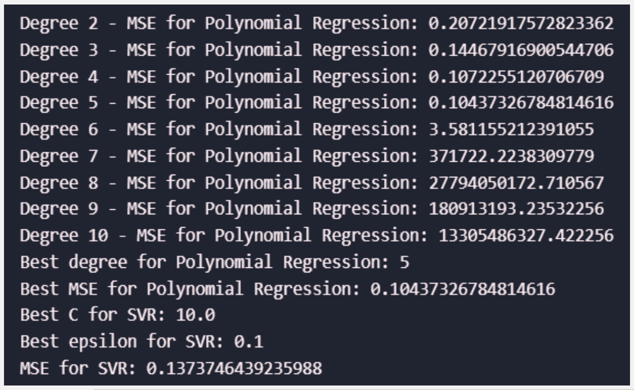


Figure 21: MSE vs Alpha.

The MSE plots show that when the regularization became too high, both models performed worse. LASSO's error increased very quickly with high alpha but Ridge's error increased more slowly.

7. Hyperparameter Tuning with Grid Search



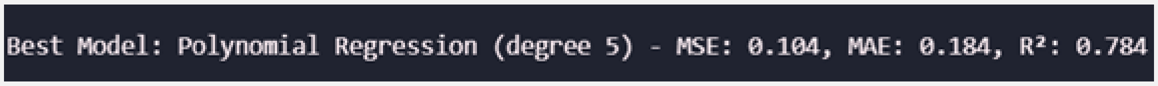
```
Degree 2 - MSE for Polynomial Regression: 0.20721917572823362
Degree 3 - MSE for Polynomial Regression: 0.14467916900544706
Degree 4 - MSE for Polynomial Regression: 0.1072255120706709
Degree 5 - MSE for Polynomial Regression: 0.10437326784814616
Degree 6 - MSE for Polynomial Regression: 3.581155212391055
Degree 7 - MSE for Polynomial Regression: 371722.2238309779
Degree 8 - MSE for Polynomial Regression: 27794050172.710567
Degree 9 - MSE for Polynomial Regression: 180913193.23532256
Degree 10 - MSE for Polynomial Regression: 13305486327.422256
Best degree for Polynomial Regression: 5
Best MSE for Polynomial Regression: 0.10437326784814616
Best C for SVR: 10.0
Best epsilon for SVR: 0.1
MSE for SVR: 0.1373746439235988
```

Figure 22 : Hyperparameter Tuning results.

The results show that Polynomial Regression worked best at degree 5 with the lowest MSE of 0.10437. However, when using degrees higher than 5, the MSE increased a lot due to overfitting. For example, at degree 8, the MSE became extremely high at 27.79 billion.

For SVR, the best parameters found were C=10.0 and epsilon=0.1, with an MSE of 0.13737. Even though this MSE is a bit higher than Polynomial Regression's best result, SVR was more stable and did not overfit.

8. Find best model after enhance the models



```
Best Model: Polynomial Regression (degree 5) - MSE: 0.104, MAE: 0.184, R²: 0.784
```

Figure 23: Best Model.

From all the models we tested, **Polynomial Regression with degree 5** was the best. It gave the lowest MSE 0.104, the lowest MAE 0.184 and the highest R² 0.784 and this means it explained 78% of the data variance while keeping the prediction error low. We also noticed in the heatmaps and bar plots that it performed better than all other models in the comparison.

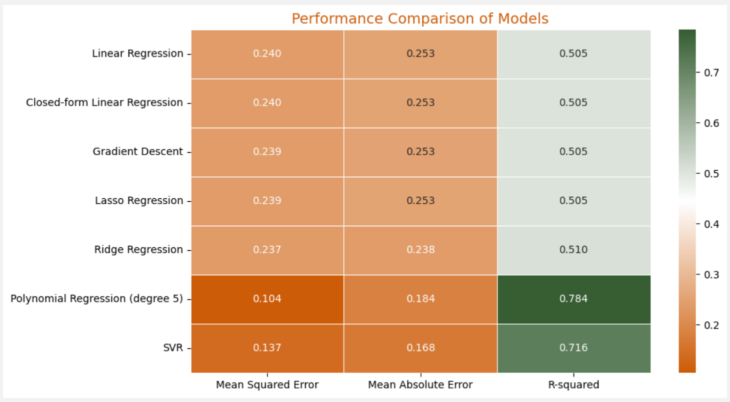


Figure 24: Performance Comparison of Models.

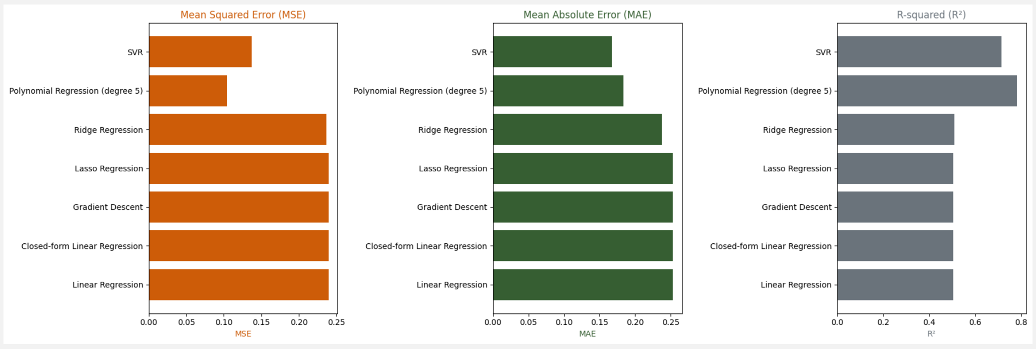
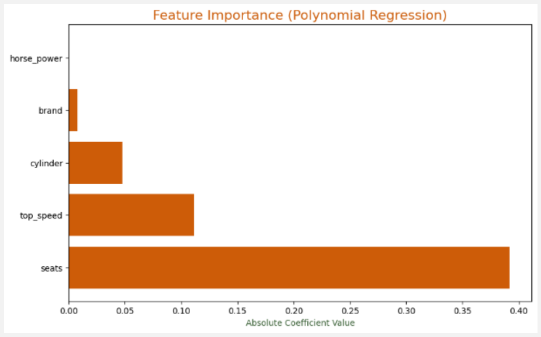


Figure 25: Performance Comparison Visualization.

However, this model has some drawbacks. Since it uses a higher degree, there is a chance it might overfit, especially if we use it on new data that is different from the training data. It could also be sensitive to noise or outliers. Additionally, training polynomial models with higher degrees takes more time and computing resources, which might not be ideal for very large datasets.

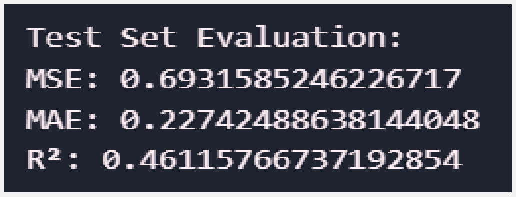
Although SVR was a strong second option with an MSE of 0.137 and an R² of 0.716, it didn't perform as well as Polynomial Regression. For simpler models like Ridge, Lasso, and standard Linear Regression methods gave similar results but were less accurate overall and explained less variance in the data. So we decided to select Polynomial Regression with degree 5.



The image shows top speed and seats are the most important features for the model.

Figure 26 : Feature Importance.

9. Model Evaluation on Test Set



```
Test Set Evaluation:
MSE: 0.6931585246226717
MAE: 0.22742488638144048
R²: 0.46115766737192854
```

Figure 28 : Model Evaluation on Test Set results.

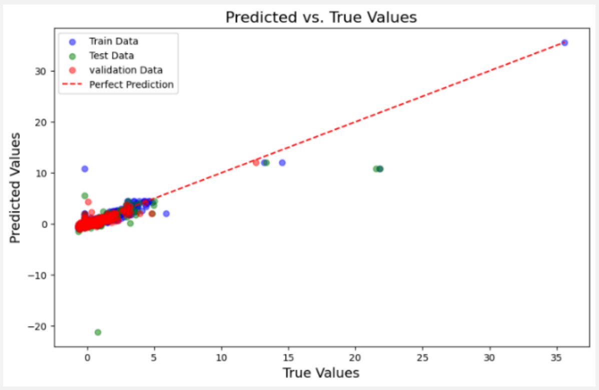


Figure 29 :Predicted vs. True Values.

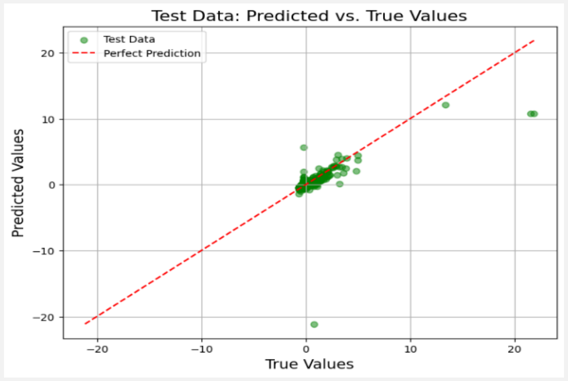


Figure 27 : Test data Predicted vs. True Values.

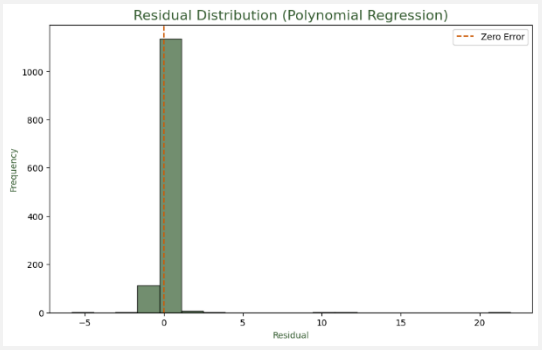
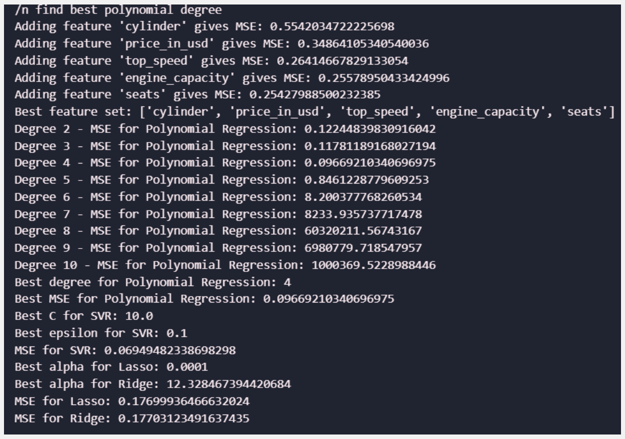


Figure 30 : Error distribution.

The test results show that the Polynomial Regression model works well for most of the data, especially for smaller values. The low MAE and the residuals close to zero mean the model gives good predictions for many data points. However, the R² result and the large errors for high values show the model has trouble with more complex relationships or extreme cases.

10.Choosing another feature as a target:



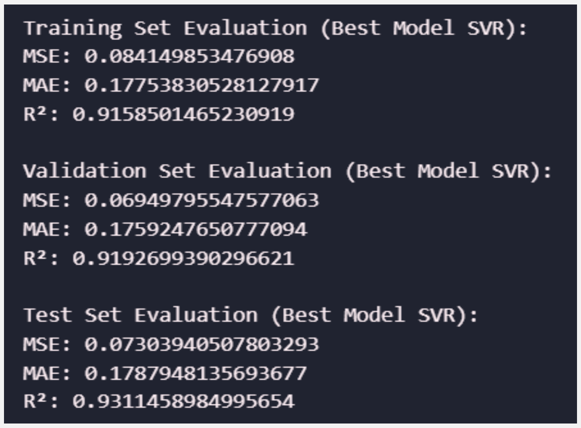
```
/n find best polynomial degree
Adding feature 'cylinder' gives MSE: 0.5542034722225698
Adding feature 'price_in_usd' gives MSE: 0.34864105340540036
Adding feature 'top_speed' gives MSE: 0.26414667829133054
Adding feature 'engine_capacity' gives MSE: 0.25578950433424996
Adding feature 'seats' gives MSE: 0.25427988500232385
Best feature set: ['cylinder', 'price_in_usd', 'top_speed', 'engine_capacity', 'seats']
Degree 2 - MSE for Polynomial Regression: 0.12244839830916042
Degree 3 - MSE for Polynomial Regression: 0.11781189168027194
Degree 4 - MSE for Polynomial Regression: 0.09669210340696975
Degree 5 - MSE for Polynomial Regression: 0.8461228779609253
Degree 6 - MSE for Polynomial Regression: 8.200377768260534
Degree 7 - MSE for Polynomial Regression: 8233.935737717478
Degree 8 - MSE for Polynomial Regression: 60320211.56743167
Degree 9 - MSE for Polynomial Regression: 6980779.718547957
Degree 10 - MSE for Polynomial Regression: 1000369.5228988446
Best degree for Polynomial Regression: 4
Best MSE for Polynomial Regression: 0.09669210340696975
Best C for SVR: 10.0
Best epsilon for SVR: 0.1
MSE for SVR: 0.06949482338698298
Best alpha for Lasso: 0.0001
Best alpha for Ridge: 12.328467394420684
MSE for Lasso: 0.17699936466632024
MSE for Ridge: 0.17703123491637435
```

Figure31 : Models after choosing horse power.

we chose "horse power" as a new target, MSE reduced to **0.25427** for feature seats. Polynomial Regression worked best with degree 4, achieving an MSE of **0.09669**, but SVR performed even better with a lower MSE of **0.0695** which shows it can handle "horse power" patterns very well. Lasso and Ridge Regression, however, had higher MSEs of **0.1770**, making them less effective for this target.

When comparing to "price" as the target, SVR performed strongly for both cases but showed better results for "horse power" MSE of 0.0695 vs. 0.137. Polynomial Regression stayed reliable for both targets but performed slightly better for "price" MSE of 0.104.

- Model Evaluation on Test Set:



```
Training Set Evaluation (Best Model SVR):
MSE: 0.084149853476908
MAE: 0.17753830528127917
R²: 0.9158501465230919

Validation Set Evaluation (Best Model SVR):
MSE: 0.06949795547577063
MAE: 0.1759247650777094
R²: 0.9192699390296621

Test Set Evaluation (Best Model SVR):
MSE: 0.07303940507803293
MAE: 0.1787948135693677
R²: 0.9311458984995654
```

Figure 32: Evaluate the model.

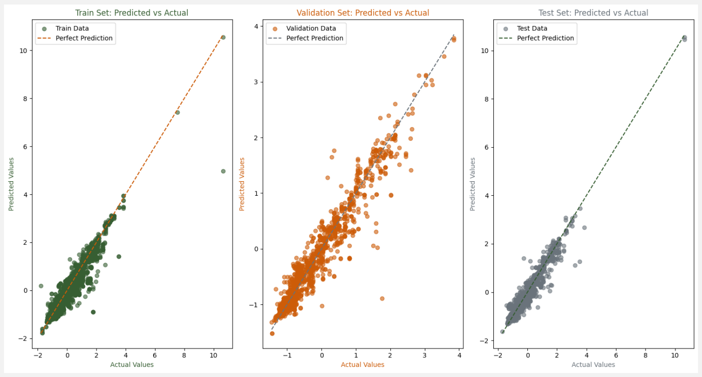


Figure 33: predicted vs actual values for train, validation, and test sets.

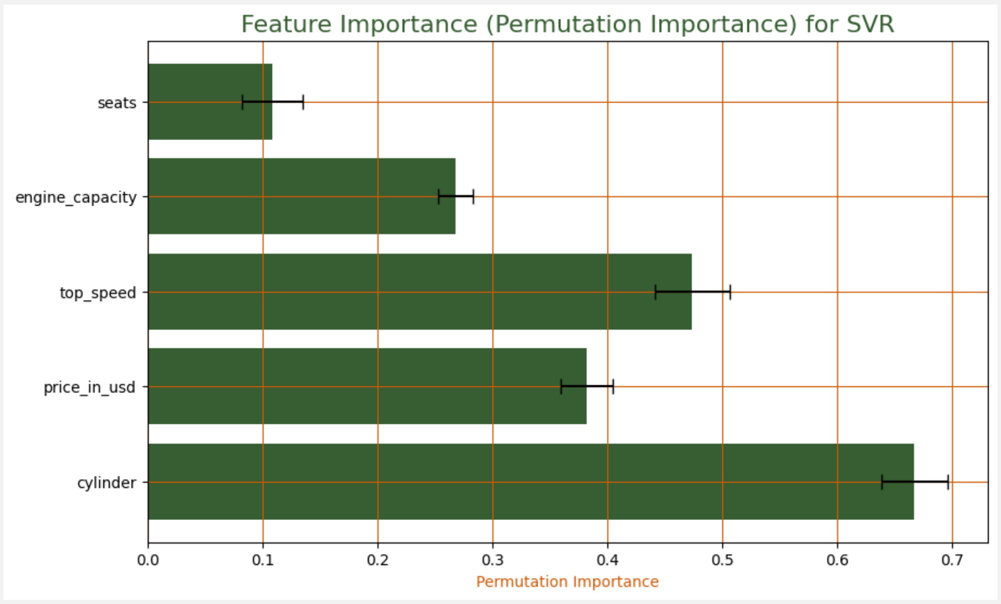


Figure 34: Feature importance.

The SVR model shows very good performance, Low MSE and MAE with high R^2 values mean the model is accurate and works well on unseen data. The "Predicted vs. Actual" plots show predictions are close to the perfect line, with small errors. The permutation importance chart explains which features are most important, and the residual plot shows most errors are near zero, but there are a few outliers. Overall, SVR is a strong and reliable model for this target variable.

We divided the work in this project as follows: Manar handled dataset cleaning, encoding, and preprocessing, as well as the first evaluation of models, regularization, and hyperparameter tuning. Layan worked on building the models, feature selection, final evaluation, and identifying the best model. For the optional task of choosing another feature as a target, we worked together to complete it.

11.Conclusion

In this project, we developed and evaluated various regression models to predict car prices using the YallaMotors dataset. After preprocessing the data, we applied linear models (Linear Regression, LASSO, Ridge) and nonlinear models (Polynomial Regression, RBF). Nonlinear models outperformed linear ones in capturing complex relationships.

We used forward selection for feature selection and applied LASSO and Ridge regularization to prevent overfitting, optimizing the models with grid search. The best model was chosen Polynomial Regression with degree 5 based on its validation performance and showed strong generalization on the test set.

Overall, the project demonstrated how feature selection, regularization and hyperparameter tuning improve model accuracy and robustness. Future work could explore more advanced models for further enhancement.