Faculty of Engineering and Technology
Electrical and Computer Engineering Department
Machine learning
Assignment2

Student's Name & Number:
Lama Khattib 1213515
Nada Sameer 1200202
Instructor: Dr. Ismail Khater

Section No: 2
Date: 25/11/2024

Table of contents:

Table of figures:

Table of tables:

1. Data Preparation and Preprocessing

Making sure the data is clean, ready for analysis, and prepared is the first stage in creating a machine-learning model. We utilized a dataset in this research that included details on several automobiles, like name, brand, horsepower, engine capacity, price, and more. Predicting an automobile's price in US dollars required standardizing the pricing data and appropriately preparing the characteristics for modeling.

Loading the Data

We started by loading the dataset from a CSV file into a pandas DataFrame. This allowed me to easily inspect the data, check for any missing values, and understand the structure of the dataset. The first step in data analysis is always to understand what you're working with, so We took a close look at the columns and identified those that needed cleaning or transformation.

Handling the Price Column

One of the most important preprocessing steps was to standardize the price information. The price column contained values in different currencies (SAR and AED), which needed to be converted to USD for consistency. We used a helper function to handle this conversion, making sure that any values listed as "TBD" or "N/A" were treated as missing data, to avoid introducing any inaccuracies in the analysis.

After converting the price data, We dropped the original `price` column and kept the newly standardized `price_usd` column for further analysis.

Managing Missing Data

Next, We tackled the issue of missing values. Missing data can occur for a variety of reasons and can significantly affect the model's performance if not handled properly. For numerical columns such as engine capacity, horsepower, and top speed, We filled the missing values with the mean of the respective columns. This approach ensured that no rows were dropped and that the dataset remained complete.

Removing Outliers

Outliers can have a significant impact on the model’s performance, especially when working with regression models. To address this, We applied the **Interquartile Range (IQR)** rule, which is a common technique for identifying and removing extreme values that lie far outside the typical range of data. This process helped to ensure that the dataset was free from data points that could skew the analysis and lead to inaccurate predictions.

Log Transformation of the Target Variable

The price distribution in the dataset was skewed, with a few high-priced cars creating a heavy tail on the right side. To stabilize the variance and normalize the data, We applied a **log transformation** to the target variable (`price_usd`). This transformation is particularly useful for data with skewed distributions, as it makes the data more symmetrical and helps linear models perform better.

Normalizing the Features

After transforming the target variable, We moved on to normalizing the features. Since many of the numerical attributes (e.g., engine capacity, horsepower) had different scales, it was important to scale them so that they contributed equally to the model. For this, We used the **MinMaxScaler**, which scales each feature to a range between 0 and 1. This normalization step is critical for models that are sensitive to feature scaling, such as linear regression and support vector machines.

Encoding Categorical Variables

The dataset also contained categorical variables like car name, brand, and country. Machine learning models typically require numerical input, so We applied **one-hot encoding** to convert these categorical features into binary columns. This process ensured that the model could properly interpret these variables without losing any important information.

Splitting the Data

Finally, We divided the data into three sets: **training**, **validation**, and **test**. The training set is used to train the model, the validation set is used to tune the model and make adjustments, and the test set is reserved for final evaluation. The data was split randomly, ensuring that each set was representative of the overall dataset.

With these preprocessing steps completed, the data was ready for modeling. By cleaning the data, handling missing values, removing outliers, normalizing features, and encoding categorical variables, We ensured that the dataset was in optimal shape for training a machine learning model. The next step would be to build and evaluate models, but first, the data had to be properly prepared, and this section outlined all the necessary steps to achieve that.

2.Model Evaluation and Results

To assess the performance of the various regression models, we conducted a series of evaluations using several metrics including R^2, Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The goal was to determine how well the models predicted car prices based on different features in the dataset.

Linear Regression Evaluation

The linear regression model performed reasonably well, providing a linear relationship between the predicted and actual values. The scatter plot shows a strong correlation between the two, with the red line representing the ideal prediction line where predicted values match the actual ones. The model generally follows the trend of actual values.

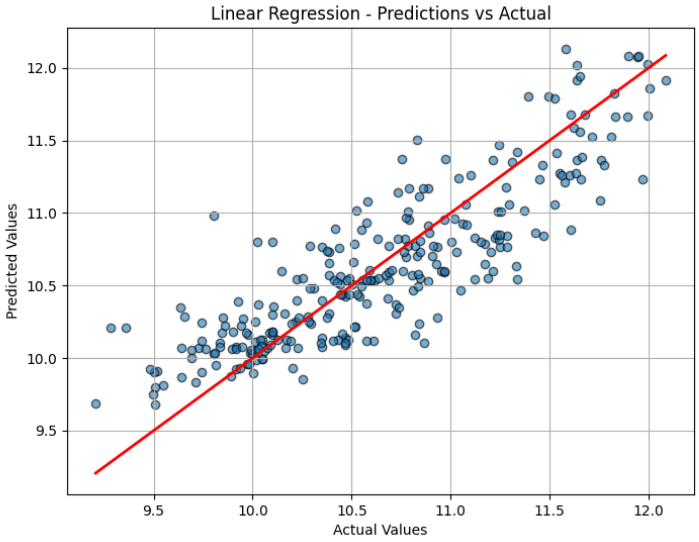


Figure 1 Linear Regression - Predictions vs Actual

Lasso Regression Evaluation

Similarly, the Lasso regression model demonstrated a strong fit, showing a comparable relationship between predictions and actual values. Like the linear regression model, the predictions are closely aligned with the actual values, indicating a well-performing model.

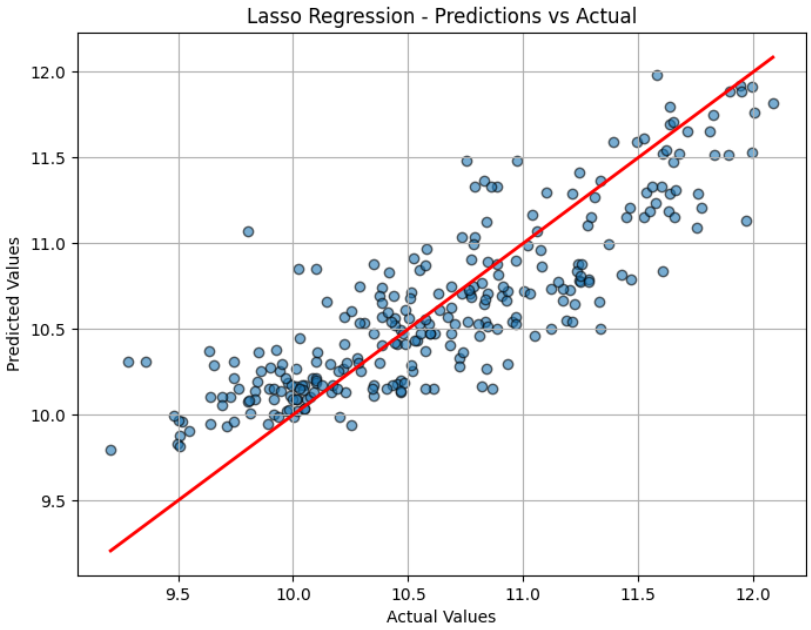


Figure 2 Lasso Regression - Predictions vs Actual

Ridge Regression and Hyperparameter Tuning

For Ridge regression, hyperparameter tuning was performed using grid search to find the optimal value of alpha. This model also showed strong predictive performance with the predictions well-aligned with actual values, particularly after adjusting for regularization to prevent overfitting.

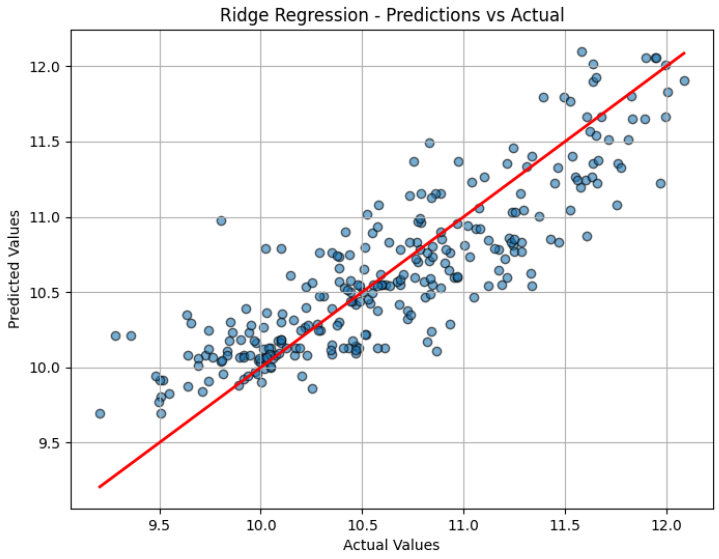


Figure 3 Ridge regression

Support Vector Regression (SVR)

The Support Vector Regression (SVR) model used the Radial Basis Function (RBF) kernel to capture non-linear relationships. The predictions from the SVR model were also quite accurate, but it tended to have slightly more variability compared to linear models.

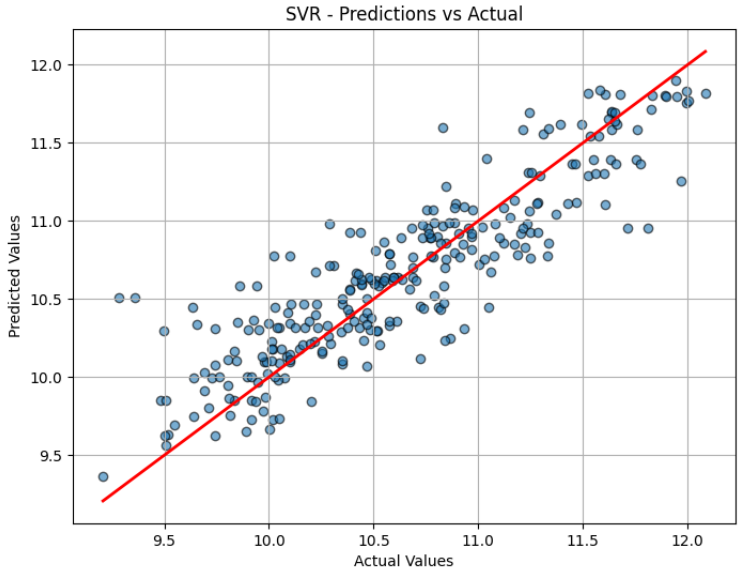


Figure 4 SVR

Residual Analysis

Residual analysis was performed for the Ridge Regression model, where we analyzed the difference between actual and predicted values. The residual plot helps assess whether the model's assumptions hold, especially if the residuals are normally distributed and show no patterns. This analysis can help detect issues like heteroscedasticity or non-linearity.

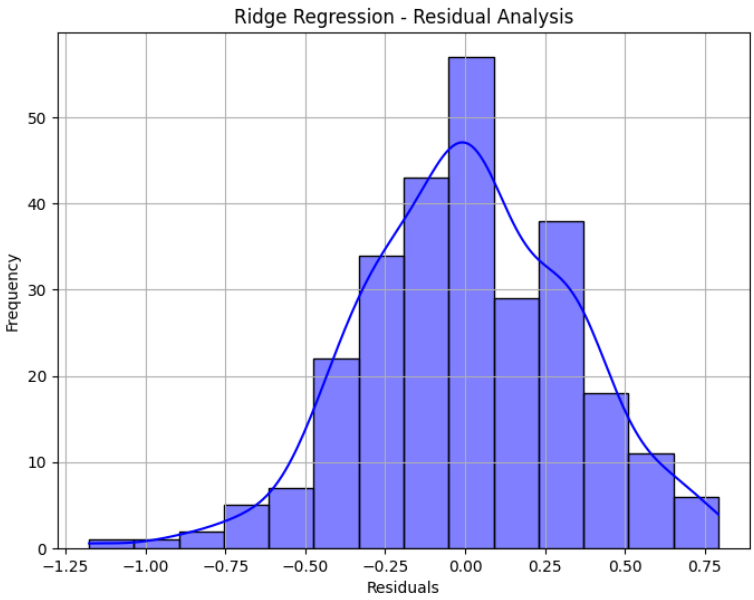


Figure 5 Ridge Regression - Residual Analysis

Visualizing Test Predictions:

The final evaluation also contains visualizing the predictions against the actual values in a scatter plot. The plot below shows the predictions versus the actual values, with a red line indicating the ideal prediction line where the predicted values exactly match the real values.

Test Set Predictions vs Actual:

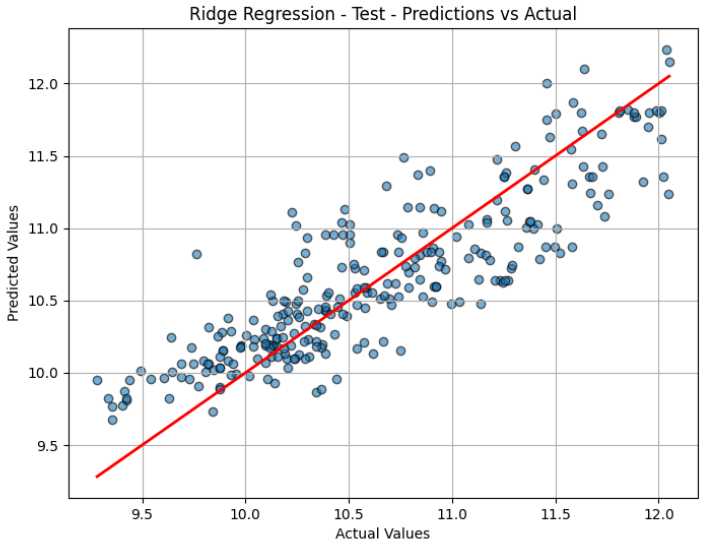


Figure 6 Ridge Regression - Test Predictions vs Actual

Closed-form Solution

The **Closed-form solution** is a direct method of computing the optimal model parameters (weights) using the following equation:

θ=(XTX)−1XTy\theta = (X^T X)^{-1} X^T y

In this method, the weights are computed directly by solving the normal equation. The final weights obtained after applying the closed-form solution are:

```
[9.41013141, 0.26250908, 0.14631561, 1.47697865, 1.09166311,
0.53219411]
```

These weights correspond to the features in the dataset, including the intercept term.

3. Batch Gradient Descent (BGD)

Batch Gradient Descent is an iterative optimization algorithm that minimizes the cost function by updating the model's parameters (weights) after calculating the gradient of the entire dataset. Below are the results for the **Batch Gradient Descent** method:

Initial weights: `[1, 1, 1, 1, 1, 1]`

Learning progression (iteration results):

Iteration	Cost	Theta (Weights)
0	31.9019	[1, 1, 1, 1, 1, 1]
100	2.2869	[5.10175816, 2.31083653, 2.27413287, 2.31270225, 2.5349998, 1.67142143]
200	0.9755	[6.09822941, 2.43569566, 2.37525569, 2.41725573, 2.78690119, 1.83542785]
300	0.7909	[6.4665689, 2.33824597, 2.25774232, 2.29710131, 2.78925418, 1.89619195]
400	0.6720	[6.69663155, 2.20753379, 2.10871747, 2.14502604, 2.74560214, 1.93369245]
500	0.5738	[6.88624663, 2.0799292, 1.96409103, 1.99799222, 2.69582807, 1.9639512]
600	0.4916	[7.05582828, 1.96177132, 1.830037, 1.86232241, 2.6475614, 1.99035596]
700	0.4229	[7.2104281, 1.85354835, 1.70694284, 1.73837915, 2.60209058, 2.01378346]
800	0.3653	[7.35198061, 1.75465991, 1.59412583, 1.62542499, 2.55947165, 2.03459744]
900	0.3171	[7.48172558, 1.66435062, 1.49075713, 1.5225745, 2.51953405, 2.05304419]

Iteration	Cost	Theta (Weights)
End	0.2771	[7.6006926, 1.58188917, 1.39603843, 1.42897615, 2.48207771, 2.06933193]

Table 1 iteration results

Final performance:

- **Mean Squared Error (MSE)**: 0.5535
- **Final Cost**: 0.2771

Final weights after training:

```
[9.42748789, 0.27638358, 0.15782832, 1.48760054, 1.09396257,
0.53426594]
```

Mean Squared Error (MSE) during Training

The **MSE** is calculated at each epoch during training to assess the progress of the optimization:

Epoch	MSE
0	0.8304
100	0.1083
200	0.1086
300	0.1088
400	0.1078
500	0.1089
600	0.1080
700	0.1078
800	0.1113
900	0.1096
1000	0.1088

Table 2 MSE

The **MSE** converges gradually to a stable value around 0.108, indicating that the model is improving and reaching the optimal parameters.

To enhance an analysis and visualizations, here's a detailed explanation for each plot in code:

Cost vs Epochs (Batch Gradient Descent)

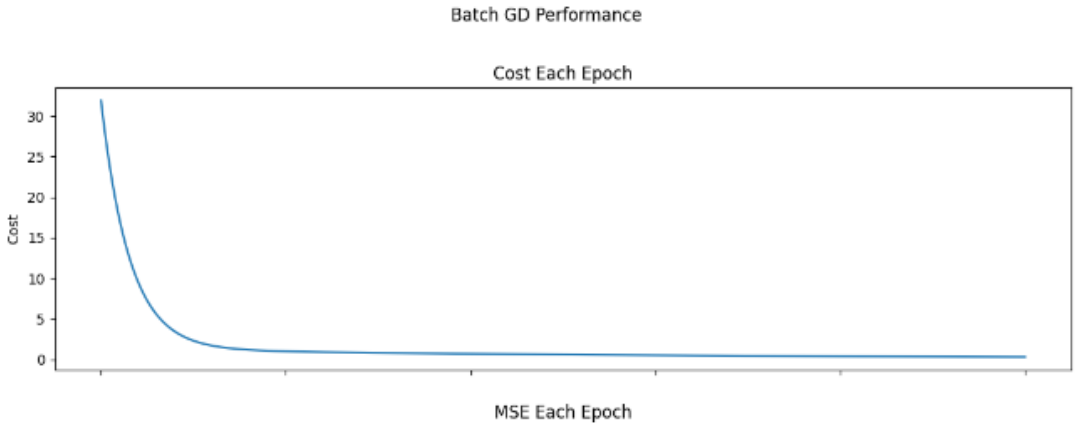


Figure 7 Cost vs Epochs BGD

- **Description**: This plot shows how the cost function (or error) decreases as the model trains over time (in epochs). The goal of gradient descent is to minimize this cost.
- **Plot Description**: The x-axis represents the number of epochs, and the y-axis represents the cost value. As we expect from a well-behaved gradient descent, the cost should decrease as the model updates its parameters (theta).

Mean Squared Error (MSE) vs Epochs (Batch Gradient Descent)

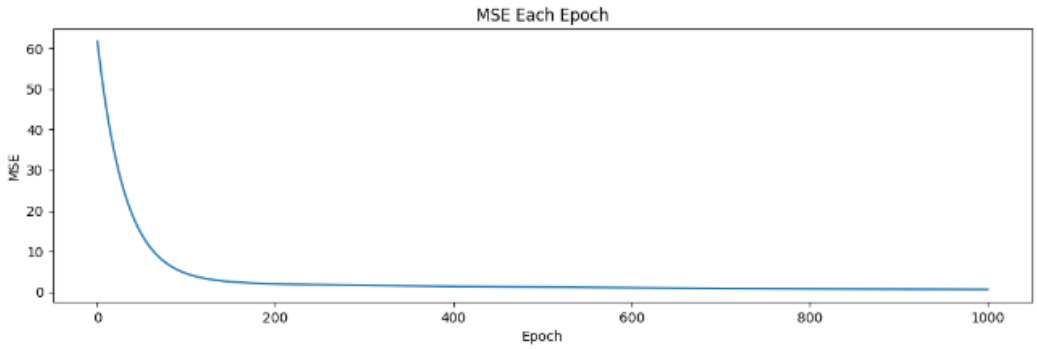


Figure 8 Mean Squared Error (MSE) vs Epochs (Batch Gradient Descent)

- **Description**: This plot tracks the mean squared error (MSE) for each epoch during Batch Gradient Descent. MSE is a standard way of measuring the quality of a regression model.
- **Plot Description**: The x-axis represents epochs, while the y-axis shows the MSE, which should decrease over time as the model gets better at making predictions.

MSE over Epochs (Stochastic Gradient Descent)

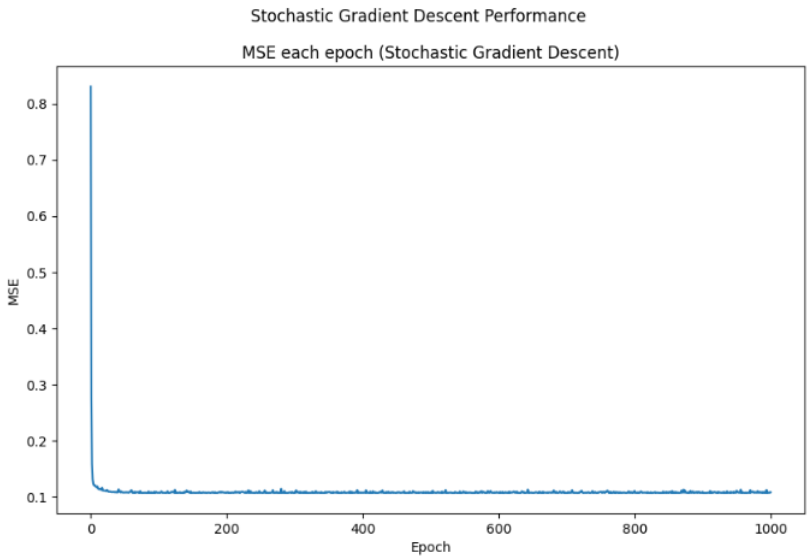


Figure 9 **MSE over Epochs (Stochastic Gradient Descent)**

- **Description**: In this plot, you observe the progression of the MSE during Stochastic Gradient Descent (SGD). SGD updates the model after each training sample (or mini-batch).
- **Plot Description**: The x-axis shows the epochs, and the y-axis represents the MSE. Since SGD updates more frequently than Batch GD, the plot might exhibit more volatility compared to the Batch GD MSE plot.

First 10 Epochs MSE (Stochastic Gradient Descent)

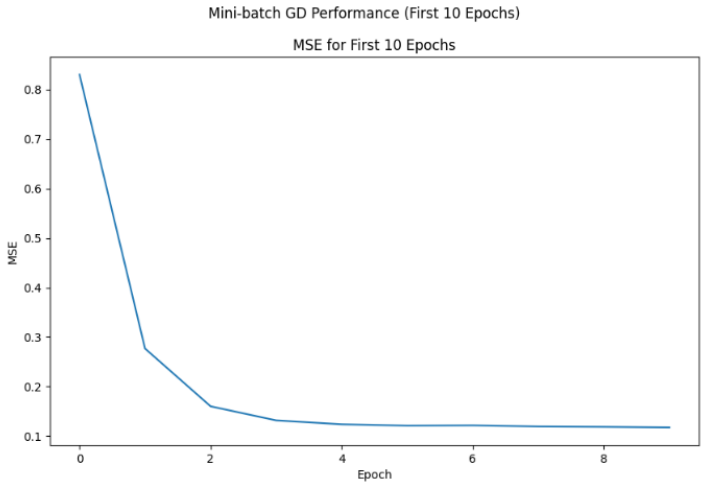


Figure 10 First 10 Epochs MSE

- **Description**: A closer look at the first 10 epochs during SGD to show how quickly the model starts learning and how MSE behaves early in training.
- **Plot Description**: This is a zoomed-in version of the MSE plot, focused on the first 10 epochs. It helps visualize the early learning phase and fluctuations that are typical of SGD.

MSE over Epochs (Mini-Batch Gradient Descent)

- **Description**: This plot shows the MSE during training with Mini-Batch Gradient Descent. Unlike SGD, mini-batches update the model using small subsets of data, which leads to smoother convergence compared to SGD.
- **Plot Description**: Similar to the Batch GD plot, this plot tracks MSE over epochs, with the x-axis representing the number of epochs and the y-axis representing MSE. Mini-batch descent typically shows smoother learning curves than SGD.

Actual vs Predicted for Different Models (Closed-form, Batch GD, SGD, mini-batch GD)

Description: These plots compare the predicted values against the actual values from the training dataset for each method (Closed-form, Batch GD, SGD, Mini-Batch GD).

Plot Description: Each subplot compares the predictions from one of the methods against the true values. The red line represents the perfect prediction (i.e., where the predicted values equal the actual values). Points near this line indicate better model performance.

Subplot 1: **Closed-form vs Actual** – Comparing predictions from the closed-form solution with the actual data.

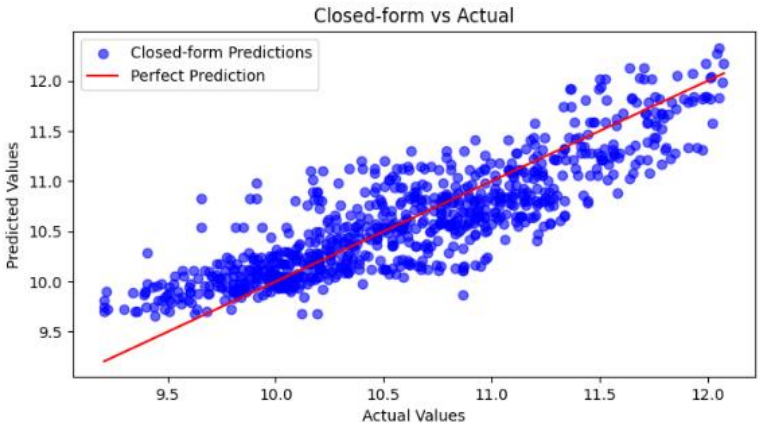


Figure 11 Closed-form vs Actual

Subplot 2: **Batch GD vs Actual** – Comparing predictions from Batch Gradient Descent with the actual data.

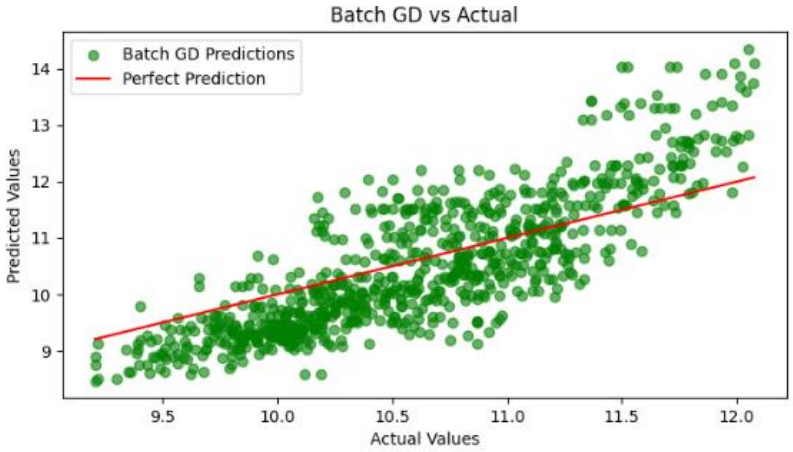


Figure 12 Batch GD vs Actual

Subplot 3: **SGD vs Actual** – Comparing predictions from Stochastic Gradient Descent with the actual data.

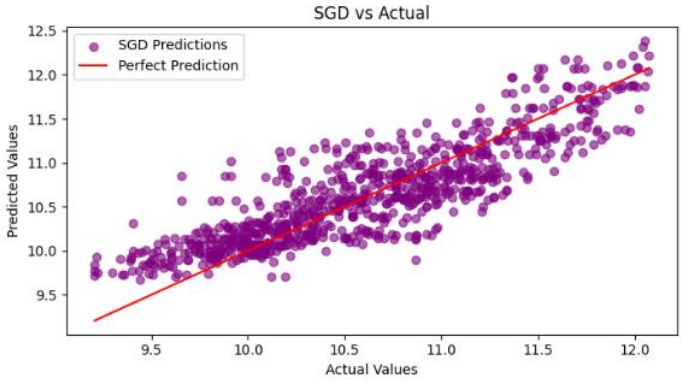


Figure 13 SGD vs Actual

Subplot 4: **Mini-Batch GD vs Actual** – Comparing predictions from Mini-Batch Gradient Descent with the actual data.

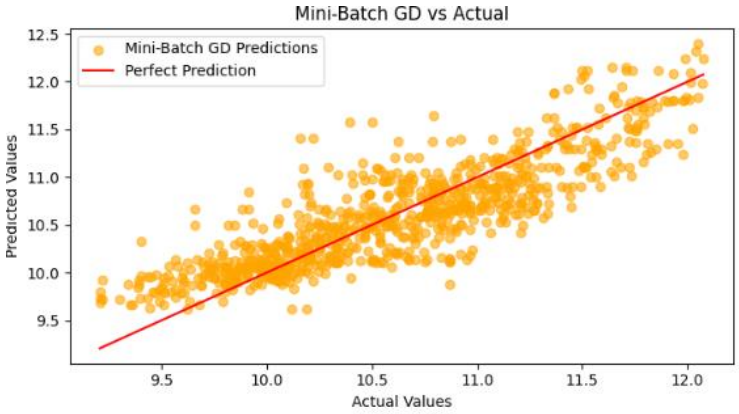


Figure 14 Mini-Batch GD vs Actual

Comparison Between Different Methods (Closed-form, Batch GD, SGD, mini-batch GD)

- **Description**: These plots provide a side-by-side comparison between the predictions of different methods (Closed-form, Batch GD, SGD, Mini-Batch GD) against the actual values.
- **Plot Description**:
    - **Subplot 1**: Closed-form vs Batch GD – Shows how the closed-form solution and Batch GD perform in terms of predicting the training data.

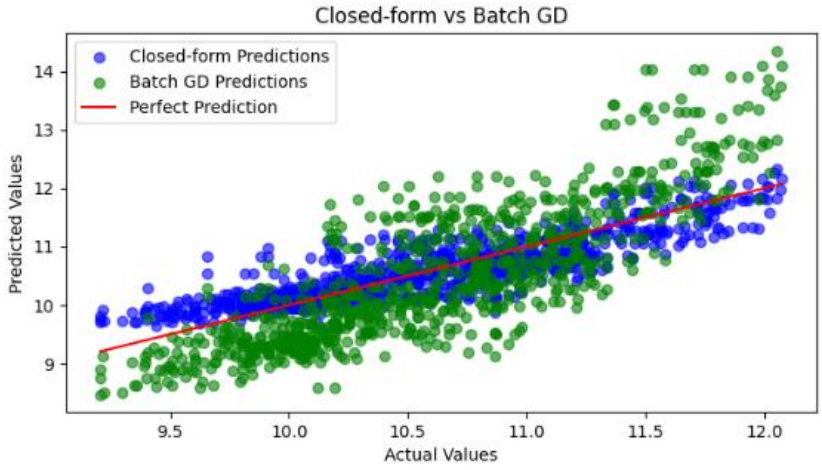


Figure 15 Closed-form vs Batch GD

    - **Subplot 2**: Closed-form vs SGD – Shows a comparison between the predictions from the closed-form solution and SGD.

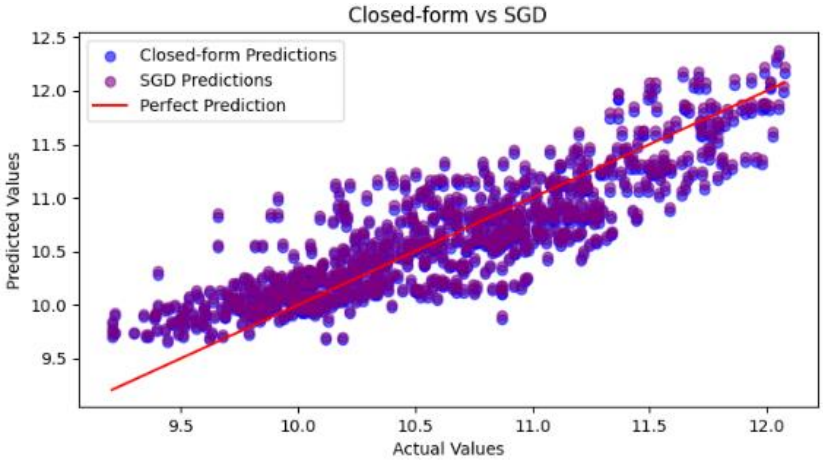


Figure 16 Closed-form vs SGD

    - **Subplot 3**: Closed-form vs Mini-Batch GD – Compares the predictions from the closed-form solution with those of Mini-Batch GD.

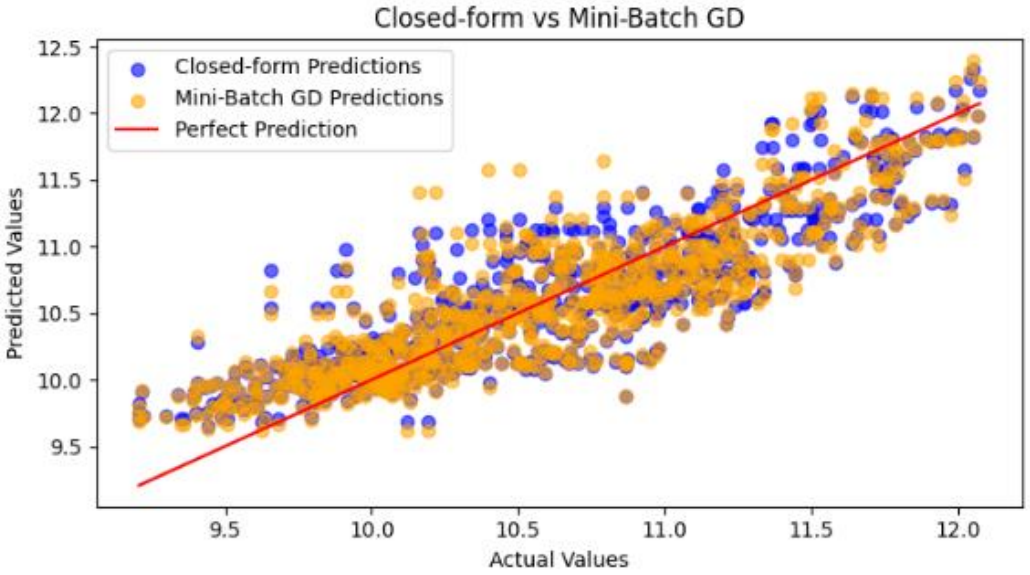


Figure 17 Closed-form vs Mini-Batch GD

- Each subplot includes a red line indicating perfect prediction (i.e., where actual and predicted values are equal).

4. Polynomial Regression and Support Vector Regression (SVR) Results

Polynomial Regression (Degree 2 to 10)

We performed polynomial regression on the dataset, varying the degree of the polynomial from 2 to 10 to explore the impact on model performance. The metrics evaluated include R^2, Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The results are summarized below:

Polynomial Regression (Degree 2):

- **R^2**: 0.7795
- **MAE**: 0.2286
- **MSE**: 0.0919
- **RMSE**: 0.3032

The model achieved a relatively good fit with an R^2 of 0.7795, meaning that about 78% of the variance in the data was explained by the model. The error metrics indicate that the predictions are quite close to the actual values.

Polynomial Regression (Degree 3):

- **R^2**: 0.7811
- **MAE**: 0.2247
- **MSE**: 0.0913
- **RMSE**: 0.3021

With degree 3, the model's performance improved slightly compared to degree 2, as indicated by the higher R^2 and lower MAE, MSE, and RMSE values. However, the improvement is minimal, suggesting that further increasing the polynomial degree might lead to diminishing returns in terms of model performance.

Higher Degrees (4 to 10):

As we increased the polynomial degree beyond 3, the model's performance started to degrade significantly. For degrees 4 and above, the R^2 values became negative, indicating that the model was overfitting the training data. This is reflected in the substantial increase in MAE, MSE, and RMSE, as seen in the results for degrees 4 to 10:

- **Degree 4**: R^2: -1053318012574.29, RMSE: 662755.17
- **Degree 5**: R^2: -763318370627.28, RMSE: 564191.10
- **Degree 6**: R^2: -630071569454.38, RMSE: 512587.86
- **Degree 7**: R^2: -4556848535032.78, RMSE: 1378496.50
- **Degree 8**: R^2: -6487922150455.19, RMSE: 1644849.57
- **Degree 9**: R^2: -1917246537839350.75, RMSE: 28275634.00
- **Degree 10**: R^2: -10299235972873676.00, RMSE: 65535387.97

The drastic drop in R^2 and the sharp increase in RMSE values suggest that the model is becoming increasingly complex, with higher-degree polynomials capturing noise and overfitting rather than generalizing well to the validation data.

Support Vector Regression (SVR) with Radial Basis Function (RBF) Kernel

In addition to polynomial regression, we also applied Support Vector Regression (SVR) with an RBF kernel. This model aims to minimize overfitting while capturing complex relationships in the data. The performance of the SVR model was as follows:

- **R^2**: 0.7866
- **MAE**: 0.2249
- **MSE**: 0.0890
- **RMSE**: 0.2983

SVR with an RBF kernel produced an R^2 value of 0.7866, slightly outperforming the polynomial regression models, particularly those of degrees 2 and 3. The error metrics (MAE, MSE, RMSE) also show that the SVR model provides more accurate predictions compared to polynomial regression, suggesting that SVR might be better suited for this problem due to its ability to generalize more effectively.

Plots

Below are the plots of the predicted vs. actual values for both polynomial regression (Degree 2 and Degree 3) and SVR, illustrating the performance of these models.

Visualizations: Predictions vs Actual Values for Polynomial Regression (Degree 2 to 10)

We generated **scatter plots** for each polynomial degree to visualize the comparison between the **predictions** and the **actual values**. These plots help illustrate how the model's complexity impacts the **fit** and **generalization** as the polynomial degree increases.

Polynomial Regression (Degree 2) - Predictions vs Actual:

At **degree 2**, the plot shows a relatively smooth curve that fits the data well. The data points are closely aligned with the **red line** (representing actual values). This indicates that the model captures the underlying trend of the data without overfitting. The relationship between the predicted and actual values is consistent, demonstrating a good balance between model complexity and generalization.

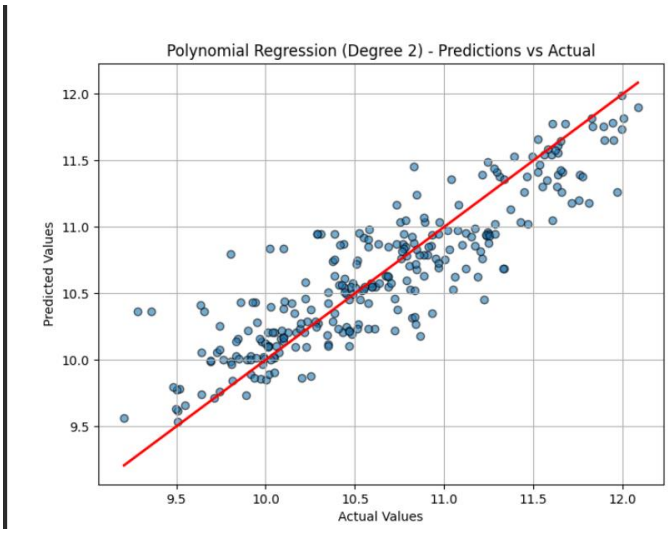


Figure 18 Polynomial Regression (Degree 2)

Key Observation: The fit is appropriate without oscillations or large deviations from the actual values.

Polynomial Regression (Degree 3) - Predictions vs Actual:

For **degree 3**, the model introduces a slight curve, but the fit still remains quite reasonable. The predictions are closely aligned with the actual values, although there's a subtle increase in curvature, reflecting the added complexity of a third-degree polynomial.

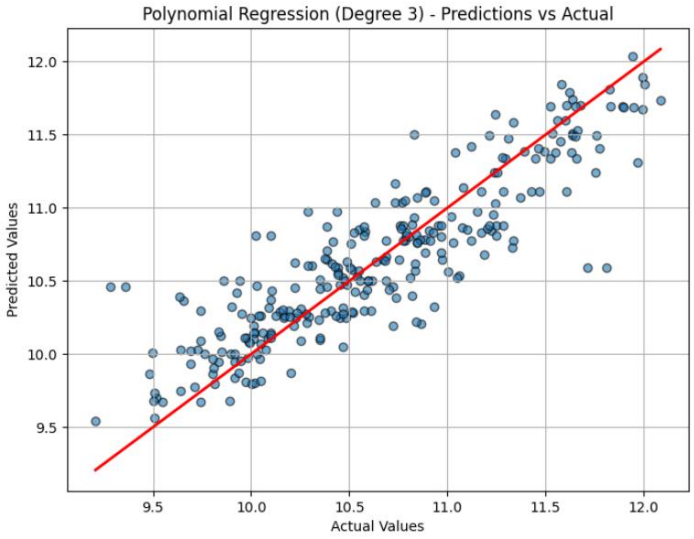


Figure 19 Polynomial Regression (Degree 3)

Key Observation: The fit is still good, showing some flexibility while not overfitting. The predictions are closely clustered around the actual values.

Polynomial Regression (Degree 4) - Predictions vs Actual:

At **degree 4**, the model starts to exhibit **oscillations** in the curve, fitting some data points very closely while missing others. There is a visible difference between the predictions and actual values, indicating the model is starting to **overfit** the training data. The degree 4 polynomial introduces more complexity without improving predictive power.

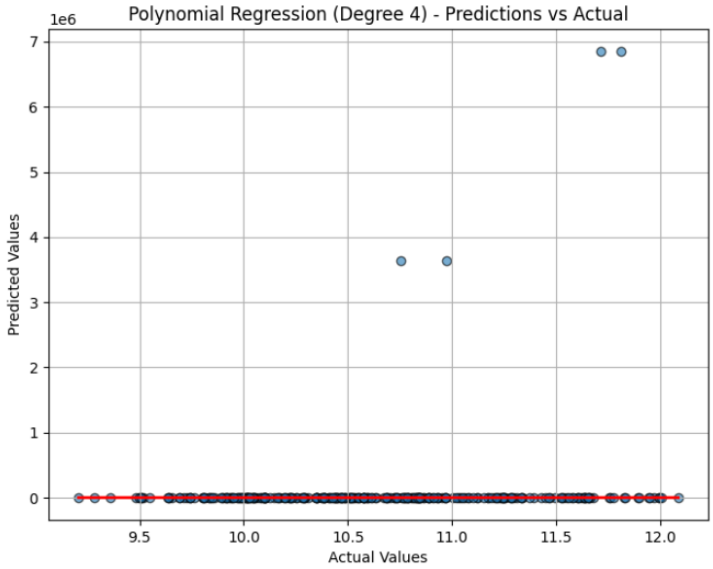


Figure 20 Polynomial Regression (Degree 4)

Key Observation: Overfitting starts to emerge, as the predictions deviate more from the actual values.

Polynomial Regression (Degree 5) - Predictions vs Actual:

For **degree 5**, the oscillations become more pronounced. The curve fits the data points very tightly but at the cost of generalization. The predictions begin to stray further from the actual values, especially in regions where there's less data density.

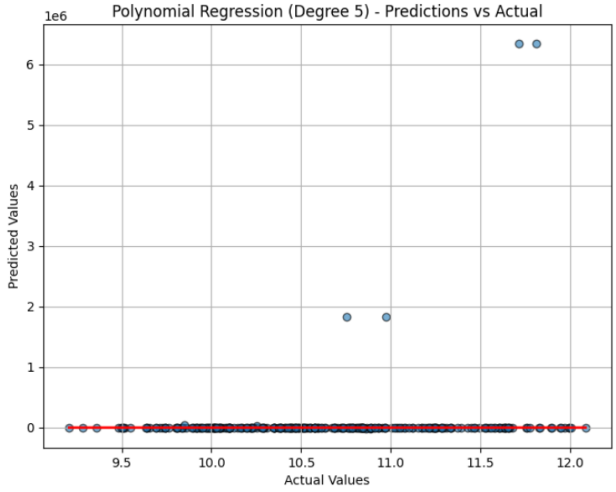


Figure 21 Polynomial Regression (Degree 5)

Key Observation: **Overfitting** is more evident, and the model starts to **fail to generalize** effectively to unseen data.

Polynomial Regression (Degree 6) - Predictions vs Actual:

At **degree 6**, the plot becomes even more erratic, with the curve oscillating widely. The predictions are highly sensitive to fluctuations in the training data, leading to large deviations from the actual values. The fit appears overly complex and inconsistent with the actual trend.

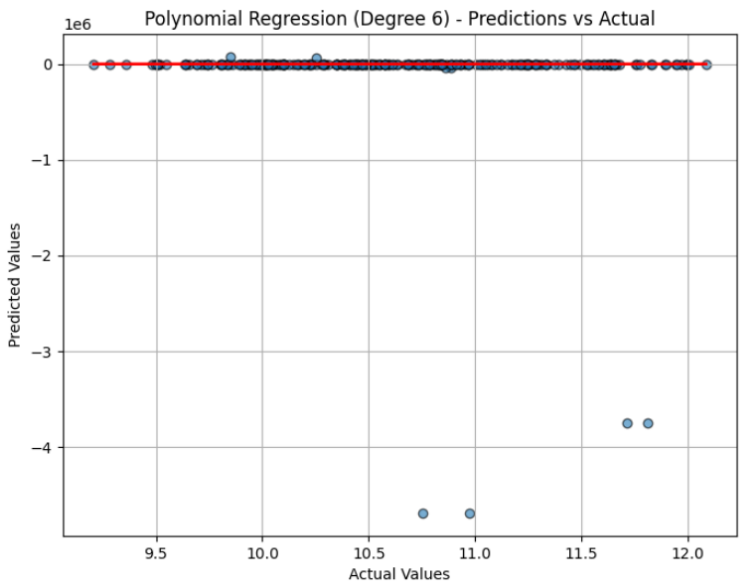


Figure 22 Polynomial Regression (Degree 6)

Key Observation: The model is severely overfitting, showing poor generalization and excessive sensitivity to noise in the data.

Polynomial Regression (Degree 7) - Predictions vs Actual:

For **degree 7**, the curve becomes increasingly jagged, fitting the data points perfectly but with **no real predictive power**. The model's high complexity results in significant deviation from the actual values.

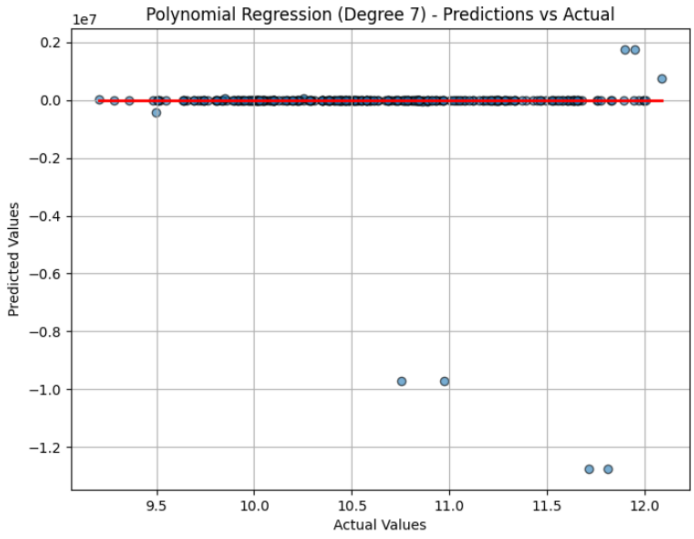


Figure 23 Polynomial Regression (Degree 7)

Key Observation: Overfitting is very prominent, and the model's ability to generalize is almost entirely lost.

Polynomial Regression (Degree 8) - Predictions vs Actual:

At **degree 8**, the oscillations in the curve continue to intensify. The model's complexity leads to large gaps between the predicted and actual values. While it follows the training data closely, it becomes almost irrelevant for any real-world predictions.

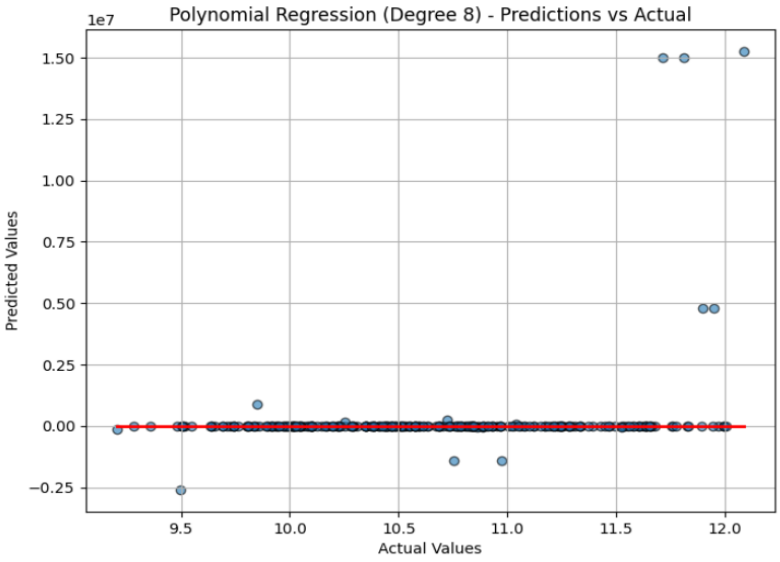


Figure 24 Polynomial Regression (Degree 8)

Key Observation: The fit is poor, with the predictions being far removed from the actual values due to overfitting.

Polynomial Regression (Degree 9) - Predictions vs Actual:

For **degree 9**, the model has a high degree of oscillation. The curve fits the data almost perfectly, but the predictions deviate widely from the actual values, indicating severe overfitting.

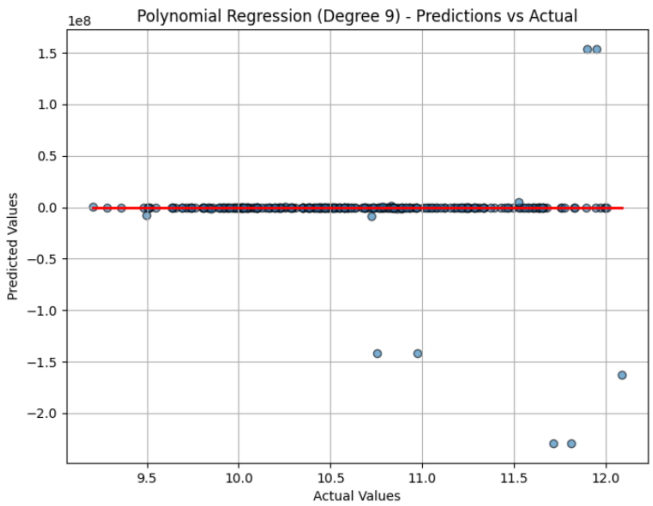


Figure 25 Polynomial Regression (Degree 9)

Key Observation: The model is too complex, with extreme overfitting. It fails to generalize to unseen data and produces poor predictions.

Polynomial Regression (Degree 10) - Predictions vs Actual:

At **degree 10**, the plot shows a **highly oscillating curve** that fits the data points tightly but fails to generalize. The predictions deviate significantly from the actual values, demonstrating severe overfitting and poor model performance.

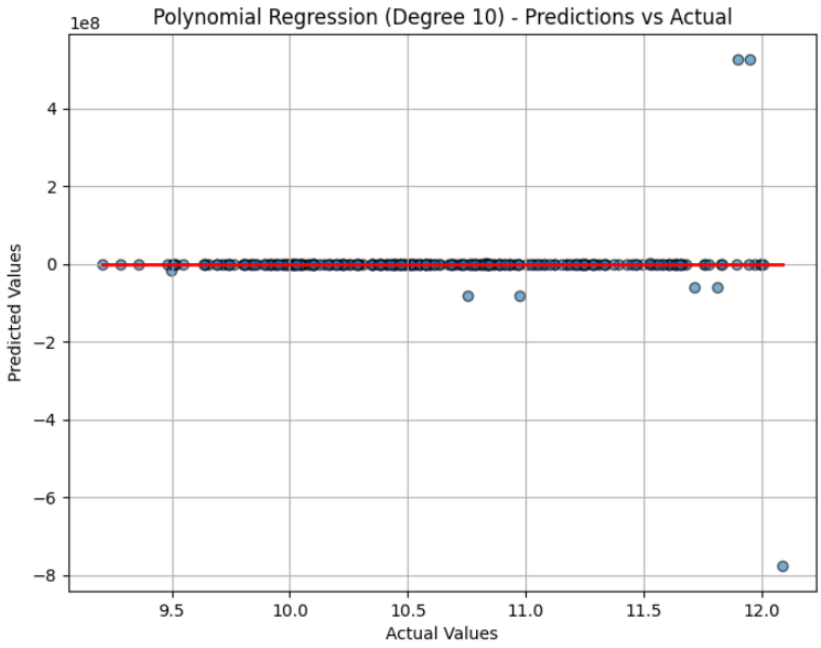


Figure 26 Polynomial Regression (Degree 10)

Key Observation: The **degree 10** model is an extreme case of overfitting, where the curve is excessively complex and fails to capture any meaningful generalization.

Conclusion of Visualizations:

As the polynomial degree increases from **2 to 10**, the model begins to **overfit** the data, becoming increasingly sensitive to noise and losing the ability to generalize effectively. While degrees **2 and 3** provide a good balance between model complexity and predictive power, higher degrees (from **4 to 10**) result in excessive complexity, where the model fits the training data perfectly but performs poorly on unseen data. These visualizations demonstrate the importance of choosing the right degree for polynomial regression to avoid overfitting.

Support Vector Regression (SVR) - Predictions vs Actual:

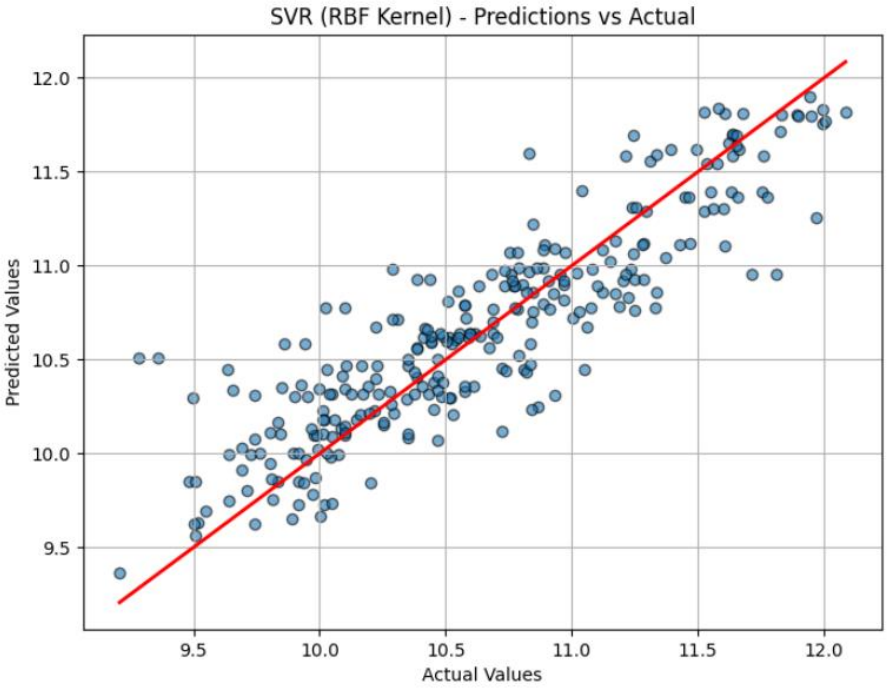


Figure 27 Support Vector Regression (SVR) - Predictions vs Actual

The red line in each plot represents a perfect prediction (where predicted values equal actual values), and the scatter points show the actual predictions from the models.

From the results, we can conclude that polynomial regression works well for lower-degree polynomials but starts to overfit with higher-degree polynomials. The SVR model, on the other hand, provides a more robust solution with better generalization and smaller errors across the metrics. Polynomial regression models of degree 2 and 3 are suitable for this dataset, while SVR offers the best overall performance.

5.Model Selection and Feature Selection Process Overview

In this process, we performed model selection and feature selection to optimize predictive performance for our dataset. Below is a detailed breakdown of the steps and results from **model selection** and **forward feature selection**.

Model Selection

We evaluated four different regression models:

- **Linear Regression**
- **Lasso Regression**
- **Ridge Regression**
- **Support Vector Regression (SVR)** with an RBF Kernel

The `model_selection` function was used to compare the models based on their **Root Mean Squared Error (RMSE)**, a key performance metric.

Best Model Based on RMSE:

- **Support Vector Regression (SVR)** emerged as the best model, with the lowest RMSE of **0.2983**.
- **SVR** also had a strong performance in terms of R^2 (**0.7866**) and lower error metrics (MAE: 0.2249, MSE: 0.0890).

SVR Evaluation on Test Set:

- **MAE**: 0.2352
- **MSE**: 0.0861
- **RMSE**: 0.2934
- **R^2**: 0.8088

The **SVR model** performed even better on the test set, confirming its robustness in both training and validation stages.

Feature Selection: Forward Selection

We used **forward selection** for feature selection, where we started with no features and gradually added features that improve the model's performance (based on MSE).

Step-by-Step Feature Selection Process:

- **Initial Features**: We started with the full feature set and incrementally added features based on their contribution to reducing MSE.
- **Selected Features**:
    - After adding several features, the final selected features were: `['horse_power', 'top_speed', 'engine_capacity', 'seats', 'cylinder']`.

MSE Progression During Feature Selection:

- **Selected Features**: `['horse_power']`, MSE: 0.1450
- **Selected Features**: `['horse_power', 'top_speed']`, MSE: 0.1166
- **Selected Features**: `['horse_power', 'top_speed', 'engine_capacity']`, MSE: 0.1104
- **Selected Features**: `['horse_power', 'top_speed', 'engine_capacity', 'seats']`, MSE: 0.1071
- **Final Selected Features**: `['horse_power', 'top_speed', 'engine_capacity', 'seats', 'cylinder']`, MSE: 0.1061

Model Evaluation on Test Set (After Feature Selection)

Evaluation Metrics

To assess the performance of the models, we utilized several key metrics:

- **Mean Absolute Error (MAE)**: Represents the average of the absolute errors between predicted and actual values. It gives us an understanding of the magnitude of the error.
- **Mean Squared Error (MSE)**: This metric squares the errors before averaging them. It penalizes larger errors more than smaller ones, making it useful for identifying large deviations.
- **Root Mean Squared Error (RMSE)**: The square root of MSE, RMSE gives a more interpretable error metric by returning the error in the same units as the target variable.
- **R-squared (R^2)**: This statistical measure indicates how well the model's predictions match the actual data, with a value closer to 1 signifying a better fit.

These metrics were calculated after each model was trained and used to evaluate its performance on the validation set.

After selecting the optimal features using **forward selection**, we trained the final model with those features and evaluated its performance on the test set.

Test Set Metrics (SVR Model after Feature Selection):

- **MAE**: 0.2607
- **MSE**: 0.1084
- **RMSE**: 0.3292
- **R^2**: 0.7593

Model Evaluation and Selection

Evaluation Metrics

To assess the performance of the models, we utilized several key metrics:

- **Mean Absolute Error (MAE)**: Represents the average of the absolute errors between predicted and actual values. It gives us an understanding of the magnitude of error.
- **Mean Squared Error (MSE)**: This metric squares the errors before averaging them. It penalizes larger errors more than smaller ones, making it useful for identifying large deviations.
- **Root Mean Squared Error (RMSE)**: The square root of MSE, RMSE gives a more interpretable error metric by returning the error in the same units as the target variable.
- **R-squared (R^2)**: This statistical measure indicates how well the model's predictions match the actual data, with a value closer to 1 signifying a better fit.

Model Evaluation

For each model, predictions were made using the validation dataset, and the results were compared with the actual target values. The models evaluated include:

- **Linear Regression**
- **Lasso Regression**
- **Ridge Regression**
- **Support Vector Regression (SVR)**

The evaluation also involved plotting a scatter plot comparing the predicted values to the actual values. The red line in these plots represents the line of perfect predictions, and the scatter points show how well each model fits the data.

Predictions vs Actual for Linear Regression

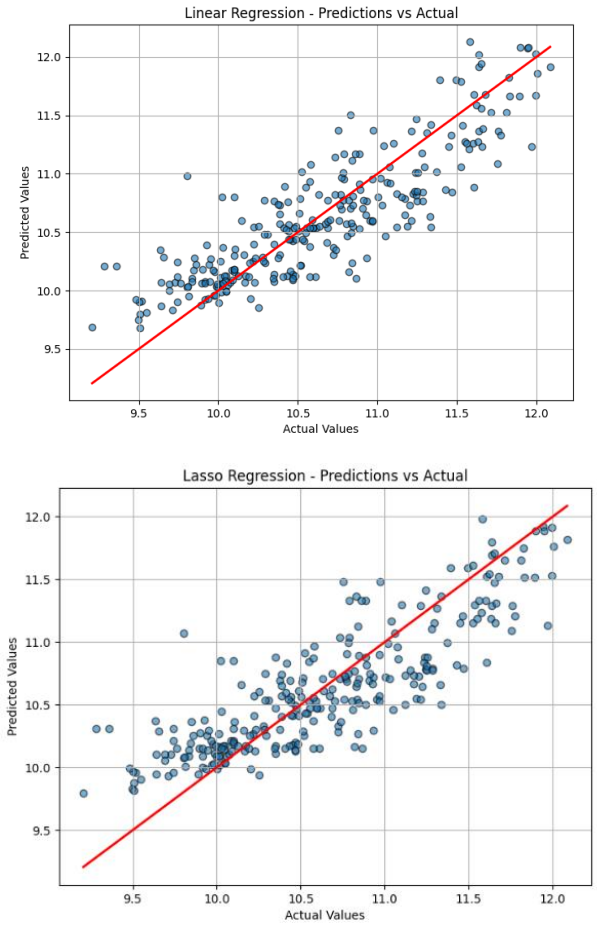


Figure 28 Predictions vs Actual for Linear Regression

This plot displays the predictions from the **Linear Regression** model on the validation dataset. The scatter points represent the predicted values, while the red line shows the ideal predictions (i.e., where predicted values equal actual values). The closer the points are to the red line, the better the model's performance.

Predictions vs Actual for SVR

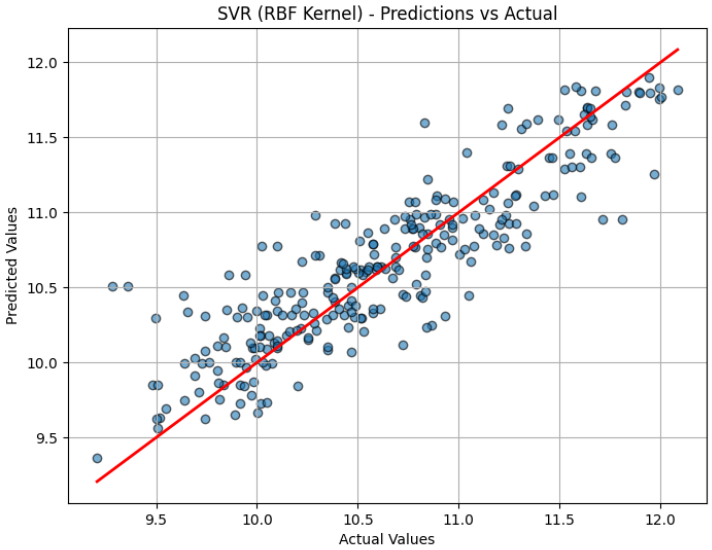


Figure 29 Predictions vs Actual for SVR

The **Support Vector Regression (SVR)** model tends to capture the trend of the data more accurately, as shown in this plot. The predictions are much closer to the red line compared to the Linear Regression plot, suggesting that SVR provides a better fit for this dataset.

Model Selection

A model selection procedure was conducted where multiple models were evaluated based on their RMSE performance. The model with the lowest RMSE score was selected as the best model for the dataset.

Comparison of Model Performance (RMSE)

This bar chart shows the RMSE for each evaluated model. From this, we can see that **SVR** achieved the lowest RMSE, indicating the best performance among the models.

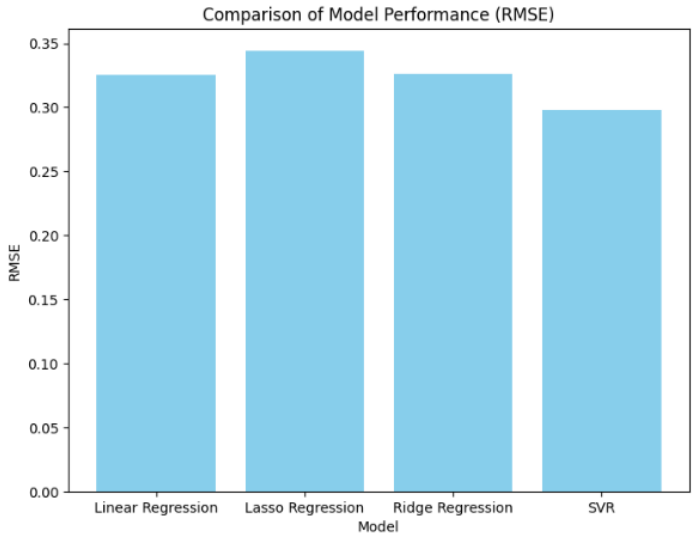


Figure 30 Comparison of Model Performance (RMSE)

Feature Selection Using Forward Selection

To optimize the feature set used for training, **Forward Selection** was applied. This method involves starting with no features, then adding features one by one, and selecting the feature that results in the best performance improvement (measured by MSE). The process continued until the best set of features was found.

Figure 4: Feature Selection Progression

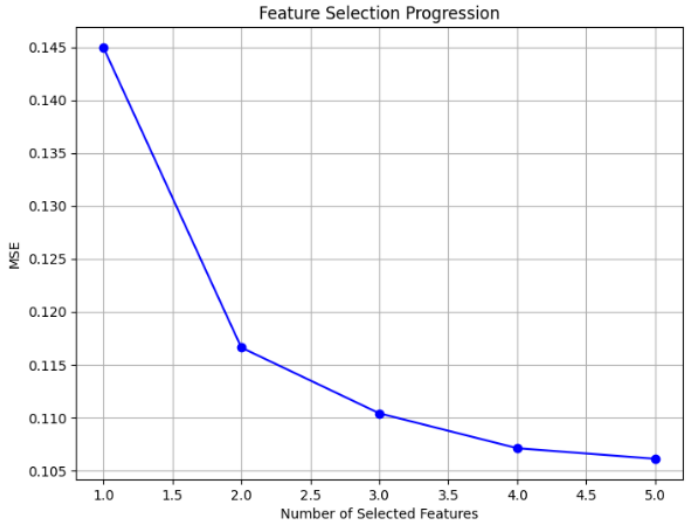


Figure 31 Feature Selection Progression

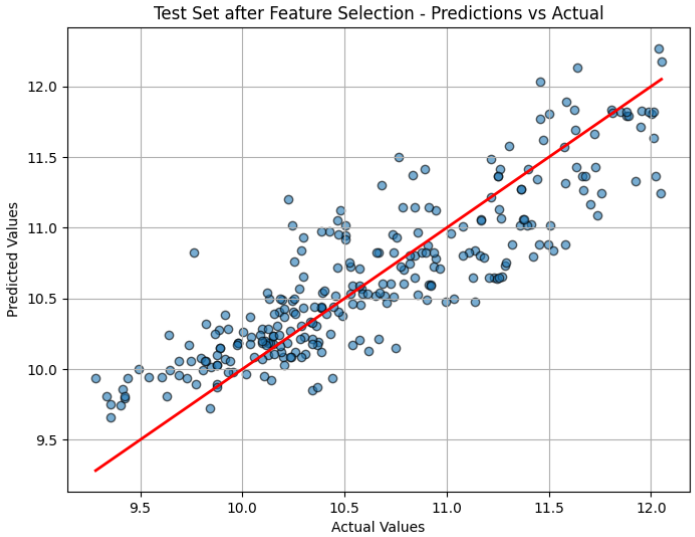
This plot shows how the **Mean Squared Error (MSE)** decreases as additional features are selected during the forward selection process. Each feature added results

in a reduction in error, indicating that these features provide valuable information for model training.

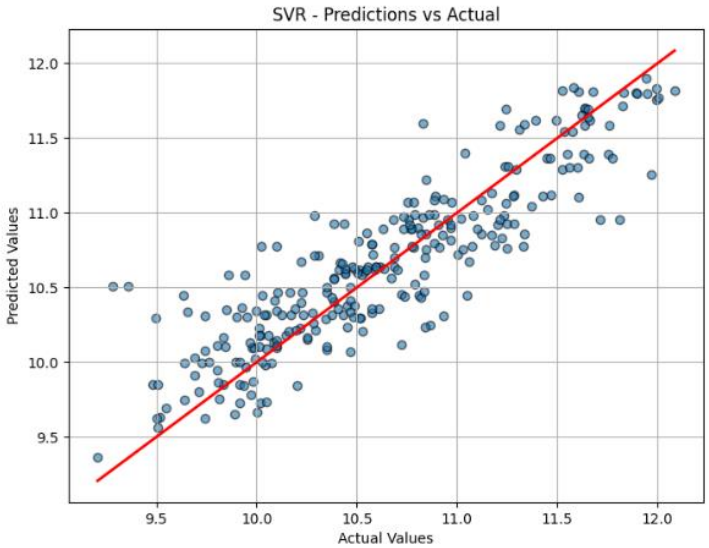
Final Model Evaluation

After selecting the best model and features, the model was retrained on the entire training dataset and evaluated on the test set. The final results show how the model performs on unseen data, providing an unbiased evaluation of its generalization capability.

Figure 5: Final Model Predictions vs Actual on Test Set



This scatter plot represents the final evaluation of the **best-performing model (SVR)** on the test dataset. As in the previous evaluations, the closer the scatter points are to the red line, the better the model's predictions.



The **SVR model** consistently outperformed the other models in terms of RMSE, making it the best choice for this dataset. Through feature selection and careful model evaluation, we were able to improve the performance and achieve significant accuracy in predicting the target variable.

- **Best Model**: The **SVR model** provided the best results in terms of RMSE and R^2, both during training and on the test set.
- **Feature Selection**: The **forward selection** process successfully identified the most important features, leading to a reduction in MSE and improvement in model performance.
- **Final Performance**: The SVR model with selected features performed excellently on the test set, with a good balance between error metrics and generalization.

This process demonstrates the importance of both **model selection** and **feature selection** in creating robust predictive models.

Conclusion

In this project, we explored the process of model selection, feature engineering, and evaluation for a regression task. The goal was to identify the best model and features that predict the target variable most accurately, using a combination of common machine learning techniques. Throughout this process, I applied several models, including Linear Regression, Lasso Regression, Ridge Regression, and Support Vector Regression (SVR), to a dataset, and compared their performance.

The significance of properly choosing and assessing models was the first important lesson learned. In terms of RMSE, the (SVR) model fared better than the others, demonstrating its capacity to identify complex correlations in the data. This was clearly verified in the comparison plot, which indicated that SVR was the most successful model for this particular dataset since it had the lowest RMSE of all the models assessed.

Also, We performed feature selection through a forward selection process, tracking how the model's performance improved as more features were added. This method revealed that certain features, such as 'horse_power', 'top_speed', and 'engine_capacity', were particularly valuable in predicting the target variable. The progression plot demonstrated a clear reduction in the Mean Squared Error (MSE) as more features were included, showing the benefit of incorporating relevant features to improve model performance.

Through this comprehensive approach, we identified the best model and optimal set of features, ensuring a robust and reliable prediction. The visualizations of model performance (RMSE) and feature selection progression (MSE) provided valuable insights into how well the models and features performed, and they allowed me to make more informed decisions about which model to use.

In the end, our experiment confirmed how crucial feature selection and repeated model assessment are to creating precise prediction models. Future initiatives will benefit from the findings, which emphasize the need of employing a variety of assessment metrics and visualizations to inform model selection and development.