جامعة بيرزيت

BIRZEIT UNIVERSITY

ENCS5341, MACHINE LEARNING AND DATA SCIENCE

Assignment 2

Students Names:

Yafa Naji	**1200708**
Noor Shamali	**1200016**

Instructor: Dr. Yazan Abu Farha

Section: #3

Date: 28/11/2024

Table of Contents

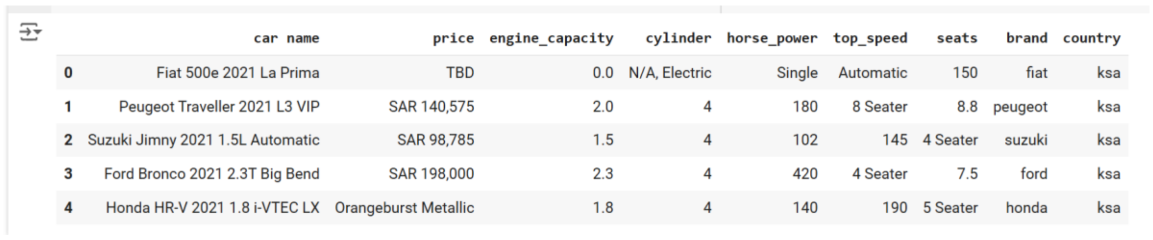
Step 1: Preprocessing the dataset

1.1 Data set description

The dataset used for this project comes from the YallaMotors website, which has 6,308 rows and 9 columns. The main objective is to predict car prices using the following features:

- **Car Name**: Contains the vehicle name and additional details.
- **Price**: Including prices in different currencies must be standardized.
- **Engine Capacity**: Represents engine displacement values.
- **Cylinder**: Shows the number of cylinders, though some values are missing or irregular.
- **Horse Power**: Indicates the power output of the car engine.
- **Top Speed**: The maximum speed that a vehicle can attain.
- **Seats**: Displays seating capacity, often categorical.
- **Brand**: Specify the car brand.
- **Country**: Specify the country of origin.

The figure below shows the top 5 rows in the dataset before being processed:



	car name	price	engine_capacity	cylinder	horse_power	top_speed	seats	brand	country
0	Fiat 500e 2021 La Prima	TBD	0.0	N/A, Electric	Single	Automatic	150	fiat	ksa
1	Peugeot Traveller 2021 L3 VIP	SAR 140,575	2.0	4	180	8 Seater	8.8	peugeot	ksa
2	Suzuki Jimny 2021 1.5L Automatic	SAR 98,785	1.5	4	102	145	4 Seater	suzuki	ksa
3	Ford Bronco 2021 2.3T Big Bend	SAR 198,000	2.3	4	420	4 Seater	7.5	ford	ksa
4	Honda HR-V 2021 1.8 i-VTEC LX	Orangeburst Metallic	1.8	4	140	190	5 Seater	honda	ksa

Figure 1: raw car dataset

1.2 Cleaning and changing data

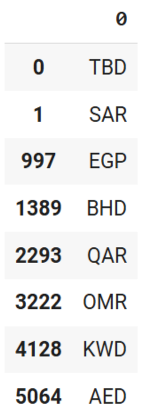
Currency Standardization:

Car prices are stated in several currencies. This makes the dataset inconsistent. By converting all values to common currency (USD), we made the target variables suitable for regression modeling to ensure consistency. This step helps prevent mistakes that may occur from mixing currency formats during training.

Steps taken:

1. Extract unique currency identifiers from the 'price' column
   - Regular expressions are used to extract unique currency symbols.
   - Match large currency abbreviations (e.g. SAR, AED)

The figure below shows unique currency identifiers extracted from the dataset. These identifiers were used to map the currency conversion rates:

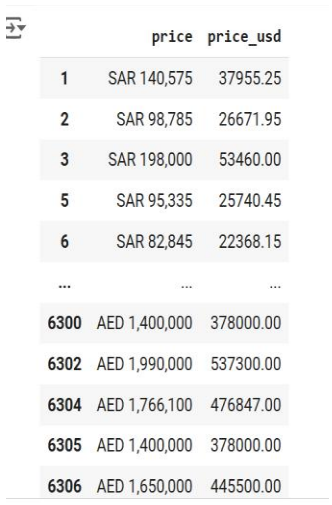


	0
0	TBD
1	SAR
997	EGP
1389	BHD
2293	QAR
3222	OMR
4128	KWD
5064	AED

Figure 2:unique currency identifiers extracted from the dataset

2. Converting to USD:
    - A default exchange rate (e.g. SAR → 0.27 USD, AED → 0.27 USD, etc.) is used for each currency.
    - Remove the currency symbol and comma from the value column.
    - Convert clean strings to numeric values. This ensures that invalid data is marked as NaN.

3. Handling Unrecognized or Invalid Data:
    - Rows with invalid currency or invalid value formats are discarded from the data set.



	price	price_usd
1	SAR 140,575	37955.25
2	SAR 98,785	26671.95
3	SAR 198,000	53460.00
5	SAR 95,335	25740.45
6	SAR 82,845	22368.15
...	...	...
6300	AED 1,400,000	378000.00
6302	AED 1,990,000	537300.00
6304	AED 1,766,100	476847.00
6305	AED 1,400,000	378000.00
6306	AED 1,650,000	445500.00

Figure 3:The price column before and after preprocessing

The standardized prices are now stored in a new column (price_usd), and any unrecognized or invalid data was removed.

1.3 Handling Missing Values

Missing values in the dataset were carefully corrected to maintain data quality and ensure compatibility for model training:

1. Numerical properties:
   - Missing values in columns engine_capacity, cylinders, horsepower, and top_speed are filled with their respective median values.
   - The median was chosen to efficiently handle outliers while preserving the central tendency of the data.
   - Inconsistent or non-numeric entries in numeric columns are converted to NaN to ensure a uniform approach to handling missing values across datasets.
2. Category features:
   - For category columns such as Seat, Brand, Country, etc., missing values are replaced with the mode that represents the value that appears most frequently in each respective column. By using these strategies The data set will be free of missing values.
   - This method ensures that the most common category is used to replace the missing data, preserving the original distribution and patterns within the dataset. By doing so, it prevents the introduction of anomalies or biases that could arise from arbitrarily assigning or leaving these values null, maintaining the dataset's overall integrity and consistency for analysis.

Column	**Missing Values**
car name	**0**
price	**0**
engine_capacity	**0**
cylinder	**0**
horse_power	**0**
top_speed	**0**
seats	**0**

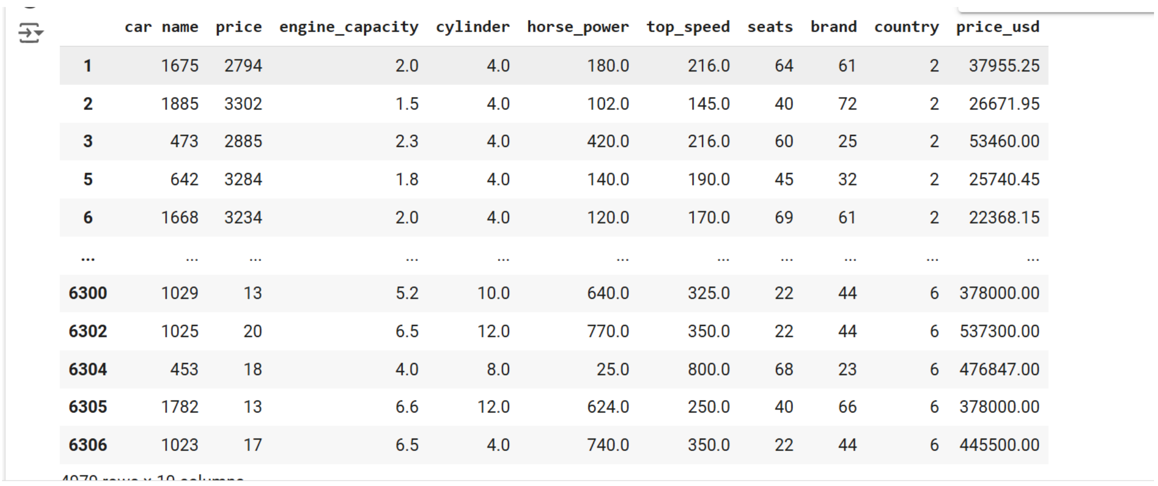
Figure 4:The result after handling missing values

1.4 Encoding Categorical Features

In the dataset, categorical features such car name, brand, country are encoded using label encoding. This procedure converts non-numeric category data to numeric values. This makes it suitable for regression models.

Machine learning models require numeric input for processing. Categorical data encoding ensures that all features are in a compatible format. So that the model can be interpreted and used efficiently during training.

Label encoding assigns a unique numeric value to each category within a column. For example, brands and other categories (such as "Toyota," "Ford," "Honda"). They are replaced with the corresponding integers, such as 0, 1, and 2, respectively.



	car name	price	engine_capacity	cylinder	horse_power	top_speed	seats	brand	country	price_usd
1	1675	2794	2.0	4.0	180.0	216.0	64	61	2	37955.25
2	1885	3302	1.5	4.0	102.0	145.0	40	72	2	26671.95
3	473	2885	2.3	4.0	420.0	216.0	60	25	2	53460.00
5	642	3284	1.8	4.0	140.0	190.0	45	32	2	25740.45
6	1668	3234	2.0	4.0	120.0	170.0	69	61	2	22368.15
...	...	...	...	...	...	...	...	...	...	...
6300	1029	13	5.2	10.0	640.0	325.0	22	44	6	378000.00
6302	1025	20	6.5	12.0	770.0	350.0	22	44	6	537300.00
6304	453	18	4.0	8.0	25.0	800.0	68	23	6	476847.00
6305	1782	13	6.6	12.0	624.0	250.0	40	66	6	378000.00
6306	1023	17	6.5	4.0	740.0	350.0	22	44	6	445500.00

4970 rows × 10 columns

Figure 5:The dataset after Encoding Categorical Features

As shown in the figure above , the encoded dataset represents categorical features numerically, enabling the models to handle and interpret these variables during analysis. The transformation preserves the original categorical relationships without introducing bias.

1.5 Feature Scaling (Standardization)

Feature scaling was applied to the numerical features in the dataset using standardization. This process ensures that all numerical values are on the same scale. This is important for machine learning algorithms that are sensitive to feature size, such as regression models.

The benefits of feature scaling:

- **Uniform Scaling**: Standardization transforms all numerical features to have a mean of 0 and a standard deviation of 1. This prevents features with larger ranges (e.g., `horse_power` or `top_speed`) from dominating those with smaller ranges (e.g., `engine_capacity` or `cylinder`).

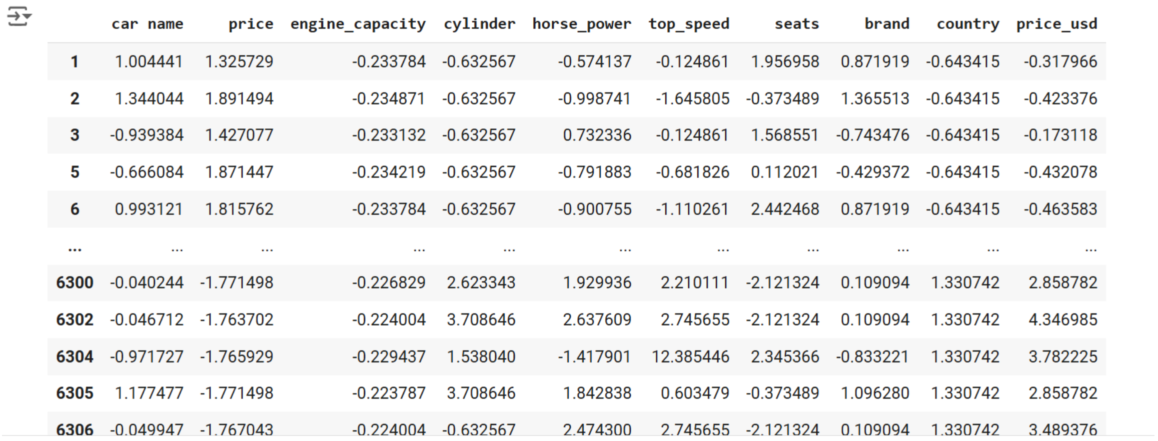
- **Improved Convergence**: Many machine learning algorithms perform better when features are scaled. Standardization enhances convergence during model training by ensuring that features contribute equally to the model's learning process.

Steps Involved:

- **Identifying Numerical Features:** Only numerical columns, which includes engine_capacity, cylinder, horse_power, top_speed, and price_usd, have been selected for standardization.
- **Applying the Scaler:** A StandardScaler was used to compute the mean and standard deviation of each numerical column. Each value was then transformed using the formula:

$$x' = (x - \mu) / \sigma$$

- **Replacing Original Values:** The standardized values replaced the original numerical data in the dataset.



	car name	price	engine_capacity	cylinder	horse_power	top_speed	seats	brand	country	price_usd
1	1.004441	1.325729	-0.233784	-0.632567	-0.574137	-0.124861	1.956958	0.871919	-0.643415	-0.317966
2	1.344044	1.891494	-0.234871	-0.632567	-0.998741	-1.645805	-0.373489	1.365513	-0.643415	-0.423376
3	-0.939384	1.427077	-0.233132	-0.632567	0.732336	-0.124861	1.568551	-0.743476	-0.643415	-0.173118
5	-0.666084	1.871447	-0.234219	-0.632567	-0.791883	-0.681826	0.112021	-0.429372	-0.643415	-0.432078
6	0.993121	1.815762	-0.233784	-0.632567	-0.900755	-1.110261	2.442468	0.871919	-0.643415	-0.463583
...	...	...	...	...	...	...	...	...	...	...
6300	-0.040244	-1.771498	-0.226829	2.623343	1.929936	2.210111	-2.121324	0.109094	1.330742	2.858782
6302	-0.046712	-1.763702	-0.224004	3.708646	2.637609	2.745655	-2.121324	0.109094	1.330742	4.346985
6304	-0.971727	-1.765929	-0.229437	1.538040	-1.417901	12.385446	2.345366	-0.833221	1.330742	3.782225
6305	1.177477	-1.771498	-0.223787	3.708646	1.842838	0.603479	-0.373489	1.096280	1.330742	2.858782
6306	-0.049947	-1.767043	-0.224004	-0.632567	2.474300	2.745655	-2.121324	0.109094	1.330742	3.489376

Figure 6:The dataset after feature scaling

• Each value in the standardized dataset represents the number of standard deviations a data point is from the mean of its respective column. For instance:

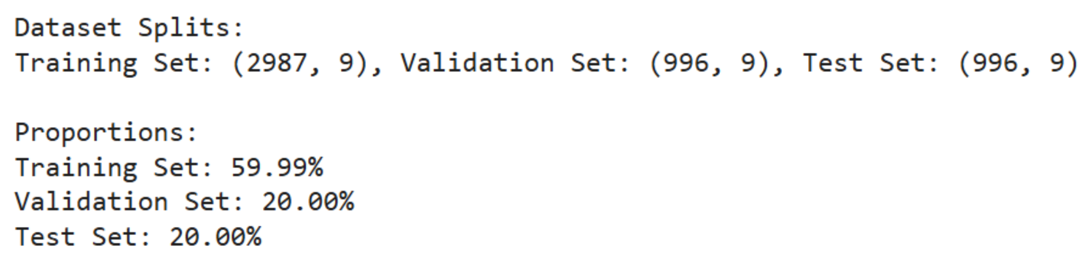
- A value of 1.32 for price in row 1 indicates that the price is 1.32 standard deviations above the mean price.
- Similarly, a value of -0.63 for cylinder in row 1 means that the cylinder count is 0.63 standard deviations below the mean.

1.6 (60-20-20) Split , 60% for training, 20% for validation, and 20% for testing

The dataset was divided into three sub-sets: training, validation, and testing sets. To achieve the following:

- **Training set (60%):** Used to train the machine learning model.
- **Validation Set (20%):** Used to tune the hyperparameters of the model during training without overfitting the test set. and to evaluate the performance of the model.
- **Test set (20%):** Used to evaluate the final performance of the model on unseen data.

The result is :



```
Dataset Splits:
Training Set: (2987, 9), Validation Set: (996, 9), Test Set: (996, 9)

Proportions:
Training Set: 59.99%
Validation Set: 20.00%
Test Set: 20.00%
```

Figure 7:Data Splitting result

The proportions closely match the intended 60-20-20 split. This balance ensures sufficient data for training while retaining enough data for model evaluation and parameter tuning.

Step 2: Building Regression Models

2.1Regression Linear Model

2.1.1 Linear Regression without Regularization

2.1.1.1Gradient Descent

The gradient descent update rule for linear regression is expressed as:

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

Figure 8:Gradient Descent Equation

Where:

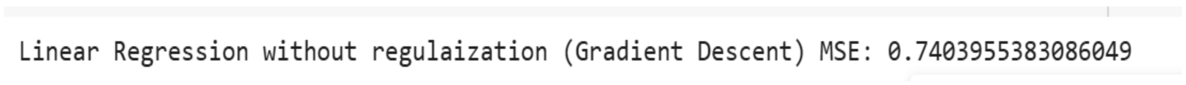
- θ :Weight vector (including the bias term).
- α: Learning rate, which controls the step size for updates.
- m: Number of training samples.
- X: Feature matrix with rows representing samples and columns representing features.
- y: Target vector.

This formula iteratively updates the weights (θ) by minimizing the Mean Squared Error (MSE), measuring the difference between predicted (Xθ) and actual (y) values. Gradient descent adjusts weights to find the best-fitting line in the feature space.

• **Learning Rate (α)**: Set to 0.01 for a balance between convergence speed and stability.

• **Iterations**: Limited to 1,000 to ensure sufficient updates.

Result:



```
Linear Regression without regulaization (Gradient Descent) MSE: 0.7403955383086049
```

Figure 9:Linear Regression without regulaization (Gradient Descent) MSE

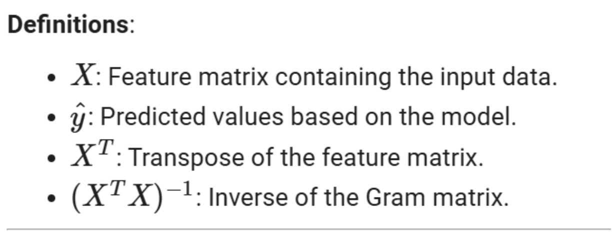
- The validation MSE is 0.7404, indicating a relatively high value. This implies that the model does not sufficiently represent the patterns in the data, resulting in large prediction errors on the validation set.
- A high MSE indicates that the model is too simple and cannot predict the target value effectively. Underfitting occurs when the model cannot accurately capture the complex relationship between the attribute and the target variable.
- Linear regression models without regularization may not be able to capture more complex patterns in the data. especially If there is a non-linear interaction

2.1.1.2Closed Form Solution

The closed-form solution for linear regression computes the optimal weights (θ) directly using the following formula:

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

Figure 10:Closed Form Solution Equation

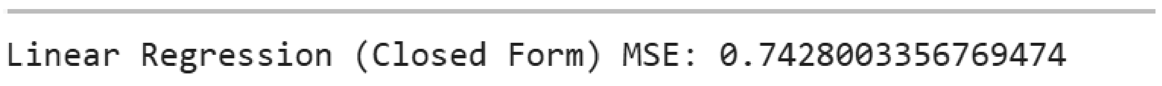


Definitions:

- X: Feature matrix containing the input data.
- $\hat{y}$: Predicted values based on the model.
- X^T: Transpose of the feature matrix.
- $(X^TX)^{-1}$: Inverse of the Gram matrix.

This formula calculates the weight vector θ that minimizes the Mean Squared Error (MSE) without iterative updates, as in gradient descent.

Result:



```
Linear Regression (Closed Form) MSE: 0.7428003356769474
```

Figure 11:Linear Regression (Closed Form) MSE

This High MSE indicates that the model captures common patterns in the data, but its high value reflects potential **underfitting**, suggesting that the model may not fully capture the relationships in the data.

Comparison with Gradient Descent:

- **Closed-Form Solution MSE**: 0.74280
- **Gradient Descent MSE**: 0.74040
- **Difference**: 0.00240 (a small but noticeable difference)

The gradient descent MSE is barely decrease than the closed-form solution MSE.

This may additionally indicate that gradient descent has higher treated nuances in the dataset, potentially due to regularization effects, numerical stability, or iterative great-tuning.

Note:

Closed-Form Solution is efficient for small to medium-sized datasets due to its direct computation of weights in a single step.

Gradient Descent scales better for large datasets because it avoids direct matrix inversion, instead updating weights iteratively.

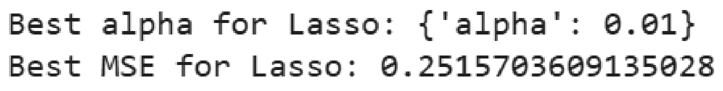
2.1.2 Linear Model using LASSO

Lasso is a technique for regression that adds an L1 regularization. It is different from normal linear regression because it applies a penalty term to the loss function to shrink less relevant coefficients towards zero, implicitly performing feature selection.

Thus, LASSO is especially useful for high-dimensional datasets when some features could be irrelevant or redundant.

2.1.2.1 Lasso Hyperparameter Tuning

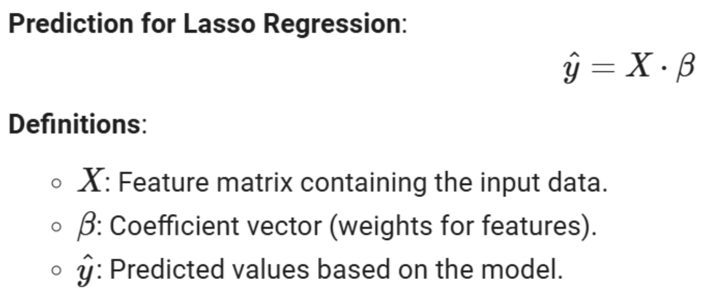
For best performance in Lasso regression, a grid search is performed to determine the most appropriate regularization parameters (Lasso). Using a medium α value [0.001,0,01,0,1,1,10,100], the model was evaluated by 5-fold cross-validation to minimize negative mean square error (MSE). This approach allows the selected α to strike a balance between regulization and accuracy. avoiding overfitting or underfitting , the optimal α for Lasso was found to be 0.010 with a corresponding MSE of 0.25160, indicating its ability to effectively regularize the model while a strong predictive performance can be preserved.



```
Best alpha for Lasso: {'alpha': 0.01}
Best MSE for Lasso: 0.2515703609135028
```

Figure 12:Best alpha and MSE for Lasso

2.1.2.2Gradient Descent



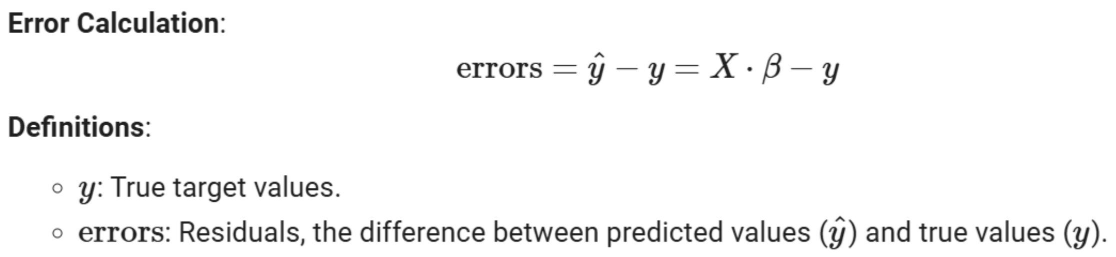
Prediction for Lasso Regression:

$$\hat{y} = X \cdot \beta$$

Definitions:

- X: Feature matrix containing the input data.
- β: Coefficient vector (weights for features).
- $\hat{y}$: Predicted values based on the model.

Figure 13:Prediction for Lasso Regression Equation



Error Calculation:

$$\text{errors} = \hat{y} - y = X \cdot \beta - y$$

Definitions:

- y: True target values.
- errors: Residuals, the difference between predicted values ($\hat{y}$) and true values (y).

Figure 14:Error Calculation Equation

Gradient Calculation with L1 Regularization:

$$\frac{\partial J(\beta)}{\partial \beta_j} = \frac{2}{m} X^T (X \cdot \beta - y) + \alpha \cdot \text{sign}(\beta_j)$$

Definitions:

- $J(\beta)$: Lasso loss function.
- m: Number of samples in the dataset.
- $\frac{2}{m} X^T (X \cdot \beta - y)$: Gradient of the Mean Squared Error (MSE) component.
- α: Regularization strength (controls the L1 penalty term).

Sign Function:

The sign function, $\text{sign}(\beta_j)$, is defined as:

$$\text{sign}(\beta_j) = \begin{cases} +1 & \text{if } \beta_j > 0 \\ -1 & \text{if } \beta_j < 0 \\ 0 & \text{if } \beta_j = 0 \end{cases}$$

Figure 15:Gradient Calculation with L1 Regularization

Gradient Descent Update Rule:

$$\beta_j := \beta_j - \eta \cdot \frac{\partial J(\beta)}{\partial \beta_j}$$

Definitions:

- β_j: Coefficient for feature j.
- η: Learning rate, controls the step size for updating coefficients.
- $\frac{\partial J(\beta)}{\partial \beta_j}$: Gradient of the Lasso loss function with respect to β_j.

Figure 16:Gradient Descent Update Rule

Mean Squared Error (MSE):

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

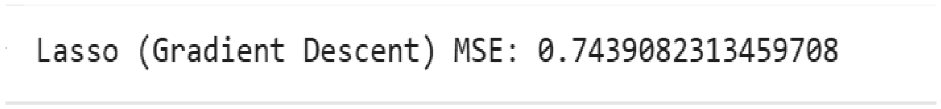
Definitions:

- y_i: True target value for sample i.
- $\hat{y}_i$: Predicted value for sample i.
- m: Number of samples in the dataset.

Figure 17:Mean Squared Error (MSE) Equation

Result:

The MSE for LASSO Gradient Descent is **0.7439**, which is slightly **higher** than the MSE for linear regression without regularization (0.7404).



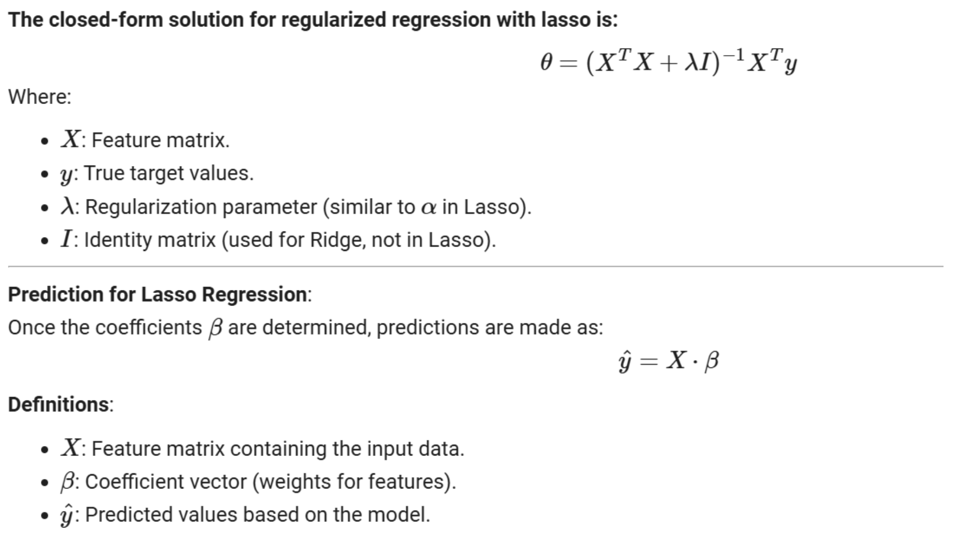
```
Lasso (Gradient Descent) MSE: 0.7439082313459708
```

Figure 18:Lasso (Gradient Descent) MSE

The increase in MSE with LASSO is due to the **L1 penalty**, which reduces the magnitude of coefficients and forces some to become exactly zero. While this simplifies the model and helps avoid overfitting, it may also eliminate some useful features, which adds a bit of error.

2.1.2.3Closed Form Solution

The closed-form solution for LASSO regression could be performed using the Least Angle Regression **(LARS)** method, which is specifically designed to solve L1-regularized regression efficiently.



The closed-form solution for regularized regression with lasso is:

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

Where:

- X: Feature matrix.
- y: True target values.
- λ: Regularization parameter (similar to α in Lasso).
- I: Identity matrix (used for Ridge, not in Lasso).

Prediction for Lasso Regression:
Once the coefficients β are determined, predictions are made as:

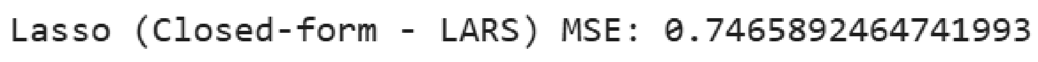
$$\hat{y} = X \cdot \beta$$

Definitions:

- X: Feature matrix containing the input data.
- β: Coefficient vector (weights for features).
- $\hat{y}$: Predicted values based on the model.

Figure 19:The closed-form solution for regularized regression with lasso

Result:



```
Lasso (Closed-form - LARS) MSE: 0.7465892464741993
```

Figure 20:Lasso (Closed-form - LARS) MSE

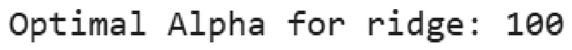
- This MSE is slightly higher than LASSO with gradient descent (0.74390) and standard linear regression (0.74040).
- The slight increase in MSE reflects the trade-offs of the L1 penalty for simplifying the model. But it loses some accuracy in the validation set.

- This may indicate that dataset do not benefit significantly from additional regularization as it may already be well-represented without it.

2.1.3 Linear Model with Ridge

2.1.3.1Ridge Hyperparameter Tuning

for ridge regression A similar grid search is performed to determine the optimal value of α, a set of alpha values. [0.001,0.01,0.1,1,10,100] It was tested using 5-fold cross-validation to confirm that the model was evaluated under different levels. of normalization This method allows for the selection of α, which improves the accuracy of MSE, ensuring that the model generalizes well to unseen data. The best α obtained from Ridge is 100. confirming that stronger regularization was necessary to stabilize the model and reduce overfitting, while achieving competitive performance.

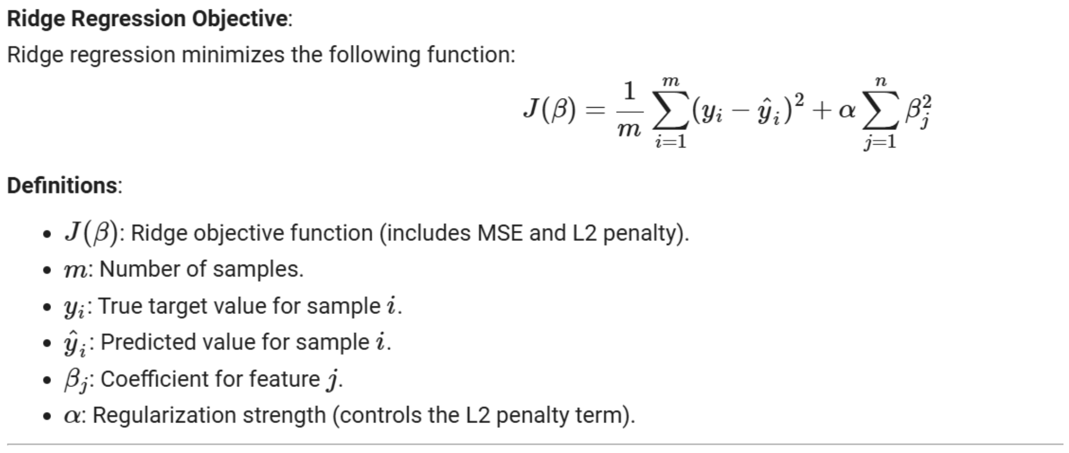


```
Optimal Alpha for ridge: 100
```

Figure 21:Optimal Alpha for ridge

2.1.3.2Gradient Descent

Ridge regression is a regularization method that incorporates an **L2 penalty** into the loss function. This technique controls overfitting by reducing the magnitude of coefficients. This ensures that the coefficient is not too large. At the same time, all model properties are preserved.



Ridge Regression Objective:

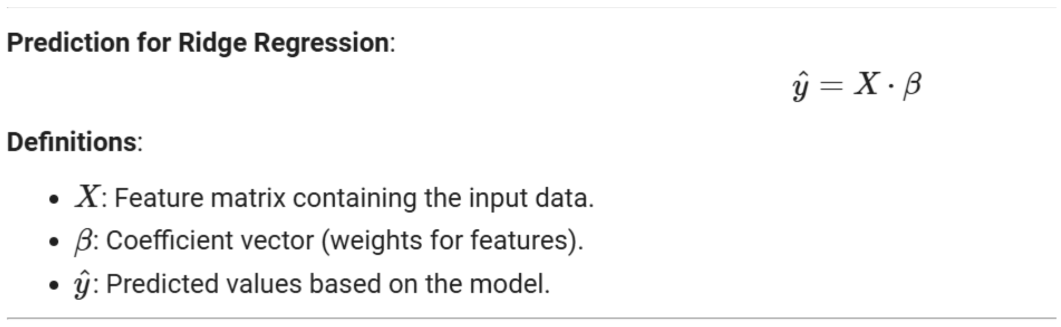
Ridge regression minimizes the following function:

$$J(\beta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \alpha\sum_{j=1}^{n}\beta_j^2$$

Definitions:

- $J(\beta)$: Ridge objective function (includes MSE and L2 penalty).
- m: Number of samples.
- y_i: True target value for sample i.
- $\hat{y}_i$: Predicted value for sample i.
- β_j: Coefficient for feature j.
- α: Regularization strength (controls the L2 penalty term).

Figure 22:Ridge Regression Objective



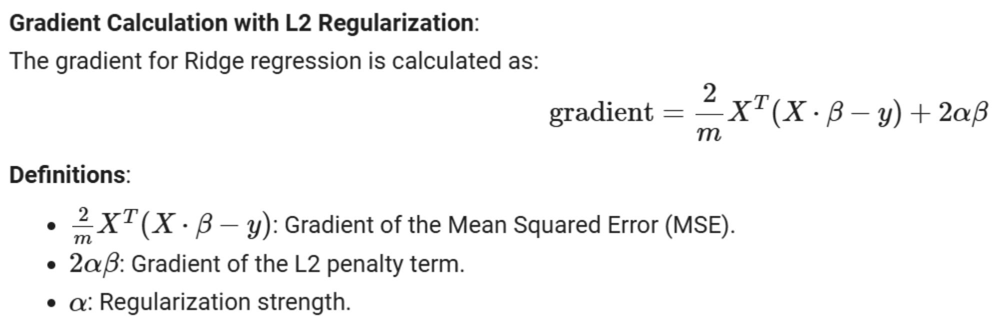
Prediction for Ridge Regression:

$$\hat{y} = X \cdot \beta$$

Definitions:

- X: Feature matrix containing the input data.
- β: Coefficient vector (weights for features).
- $\hat{y}$: Predicted values based on the model.

Figure 23:Prediction for Ridge Regression



Gradient Calculation with L2 Regularization:
The gradient for Ridge regression is calculated as:

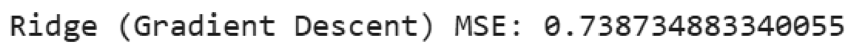
$$\text{gradient} = \frac{2}{m} X^T (X \cdot \beta - y) + 2\alpha\beta$$

Definitions:

- $\frac{2}{m} X^T (X \cdot \beta - y)$: Gradient of the Mean Squared Error (MSE).
- $2\alpha\beta$: Gradient of the L2 penalty term.
- α: Regularization strength.

Figure 24:Gradient Calculation with L2 Regularization

Result:



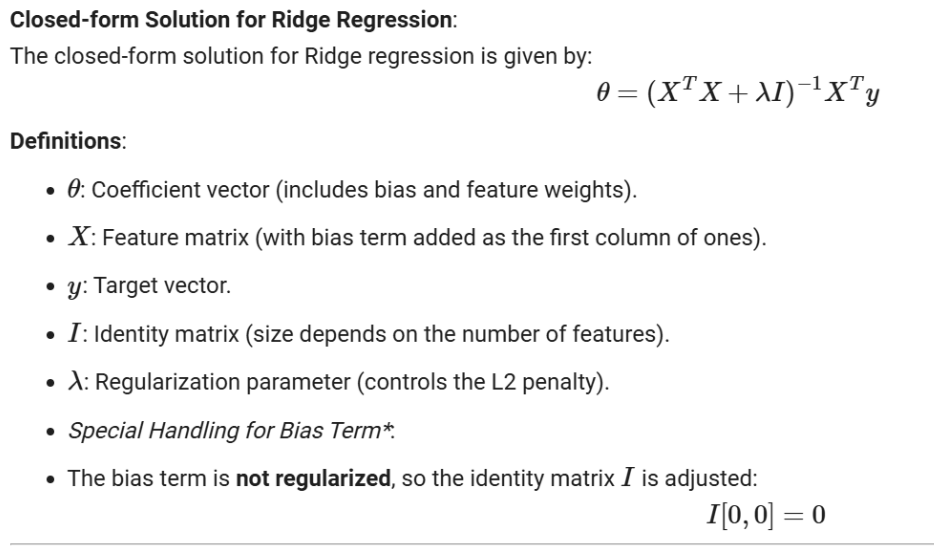
```
Ridge (Gradient Descent) MSE: 0.738734883340055
```

Figure 25:Linear Model with Ridge (Gradient Descent) MSE

- **Validation MSE**: 0.73870, slightly lower than LASSO (0.74390) and regularized linear regression (0.74040).
- It effectively reduces overfitting by shrinking the coefficients without forcing any to become zero.
- The inclusion of the L2 penalty stabilizes the model, especially when features are correlated.

2.1.3.3Closed Form Solution

Ridge regression with closed form solution includes L2 regularization in the loss function. This method calculates the optimal coefficient (θ) directly without iterative optimization by solving the following formula.



Closed-form Solution for Ridge Regression:

The closed-form solution for Ridge regression is given by:

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

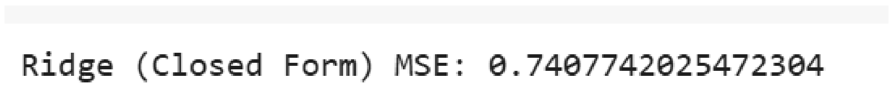
Definitions:

- θ: Coefficient vector (includes bias and feature weights).
- X: Feature matrix (with bias term added as the first column of ones).
- y: Target vector.
- I: Identity matrix (size depends on the number of features).
- λ: Regularization parameter (controls the L2 penalty).
- *Special Handling for Bias Term**:
- The bias term is **not regularized**, so the identity matrix I is adjusted:

$$I[0,0] = 0$$

Figure 26:Closed-form Solution for Ridge Regression

Result:



```
Ridge (Closed Form) MSE: 0.7407742025472304
```

Figure 27:Ridge (Closed Form) MSE

The **Ridge (Closed Form) MSE: 0.7408** indicates that using the optimized regularization parameter (λ=100) improved the model's performance, bringing the MSE closer to the unregularized linear regression (MSE=0.7404) and slightly better than the earlier Ridge closed-form solution (MSE=0.7428).

2.1.4 Comparison of Regression Linear Model

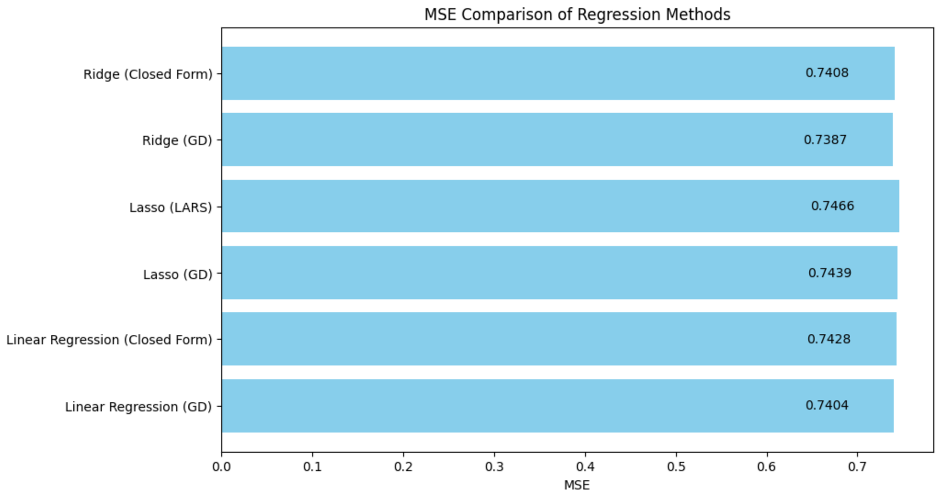


Figure 28: Comparison of Regression Linear Model

The figure above shows the **MSE comparison between regression methods**:

1. **Linear Regression**:
   - Gradient Descent and Closed-Form methods have approximately similar MSE values, indicating minimal differences in their performance due to the simplicity of the model and lack of regularization.
   - **MSE**:
     - Gradient Descent: **0.7404**
     - Closed-Form: **0.7428**
2. **Lasso Regression**:
   - Both Gradient Descent and LARS (Least Angle Regression) implementations show slightly higher MSE than linear regression methods, reflecting the trade-off introduced by the **L1 penalty**.
   - **MSE**:
     - Lasso (GD): **0.7439**
     - Lasso (LARS): **0.7466**
3. **Ridge Regression**:
   - The Ridge methods achieve lower MSE compared to Lasso, reflecting the effectiveness of **L2 regularization** in stabilizing coefficients and reducing overfitting.
   - **MSE**:
     - Ridge (GD): **0.7387** (lowest MSE overall)
     - Ridge (Closed Form): **0.7408**

Discussion of the result :

- **Best Performing Method**: Ridge (GD) achieved the lowest MSE (**0.7387**), demonstrating the impact of L2 regularization and iterative optimization.
- **Lasso vs. Ridge**: Ridge regression is often better than Lasso because of its ability to minimize all factors. Instead of eliminating some coefficients.
- **Linear Regression:** Although the linear regression method is simple, But linear regression methods lack the power of regularization. This makes them often overfitting in more complex data sets.

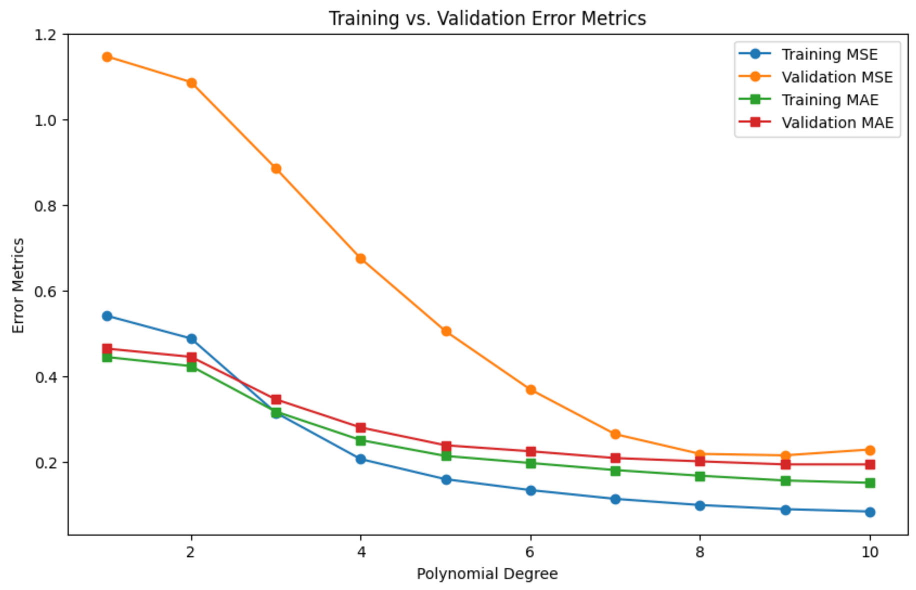
Using the regression models (Linear Regression, Lasso, and Ridge), it is clear that all models show signs of underfitting, as MSE values indicate an inability to capture the complexity of the underlying relationships in the dataset. This suggests that the relationship between the feature and the target variable is likely to be non-linear. And these linear models may not be enough to accuratly model the data. To improve prediction efficiency and manage data processing more efficiently, it is necessary to switch to a non-linear model or to improve the architecture by changing the existing non-linear properties. An example of a suitable nonlinear model is polynomial regression and standard Gaussian kernel, known here as a Radial Basis Function (RBF).

2.2Regression Non-Linear Model

2.2.1 Polynomial Regression without Regulization

The purpose of the code is to apply polynomial transformations (Polynomial Features) on the data and examine how the performance of the model is affected by varying polynomial degrees. In order to standardize the data, the algorithm first applies polynomial transformations for degrees 1 through 10 before scaling these features using StandardScaler. The Adam Optimizer is then used to train the model, and the Mean Squared Error (MSE) is used as the loss function. Three primary measures—Mean Squared Error (MSE), Mean Absolute Error (MAE), and R2—are used to assess the model's performance. These metrics show how well the model predicts the actual values.

Result:



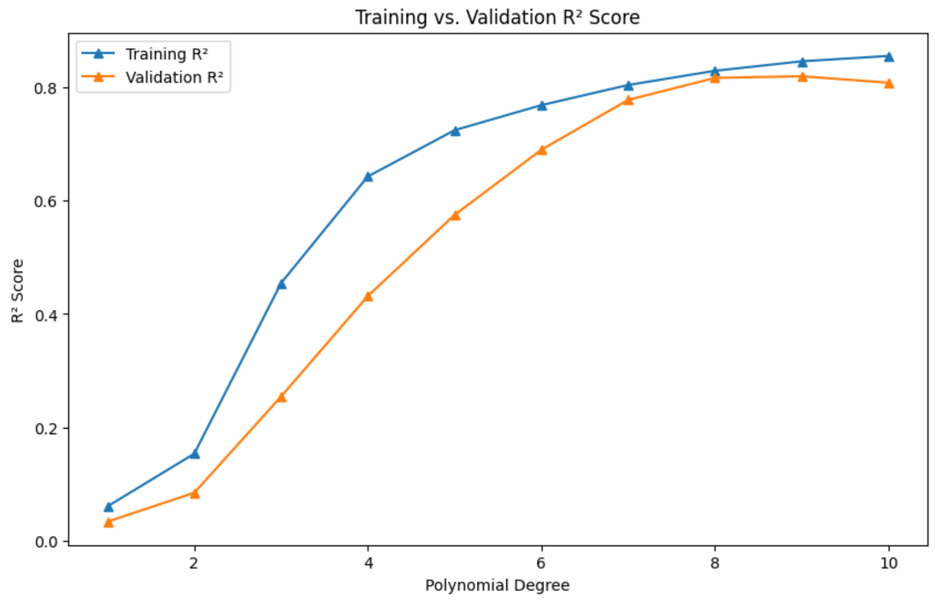


Figure 29: Results for Polynomial Regression (Without Regularization)

According to the results, the model performs noticeably better on the training set as the polynomial degree grows, with R2 values rising and MSE and MAE values falling. Higher degrees (between 7 and 10) suggest possible overfitting because the model performs exceptionally well on the training set but does not significantly improve on the validation set. That is, instead of learning the fundamental relationship between the features and the target, the model starts to learn random details from the training data.

According to these findings, polynomial degrees ranging from 4 to 7 are the most efficient, offering the optimum trade-off between training and validation performance; higher degrees, on the other hand, can be more prone to overfitting.

2.2.2. *Radial Basis Function (RBF)*

Using a polynomial kernel, we implemented Support Vector Regression (SVR) and examined the impact of various polynomial degrees and values for the parameters C and epsilon. To enhance model performance, PolynomialFeatures was used to modify the features, and StandardScaler was then used for scaling. The validation data was used to assess the model after it had been trained on the training data.

Result:

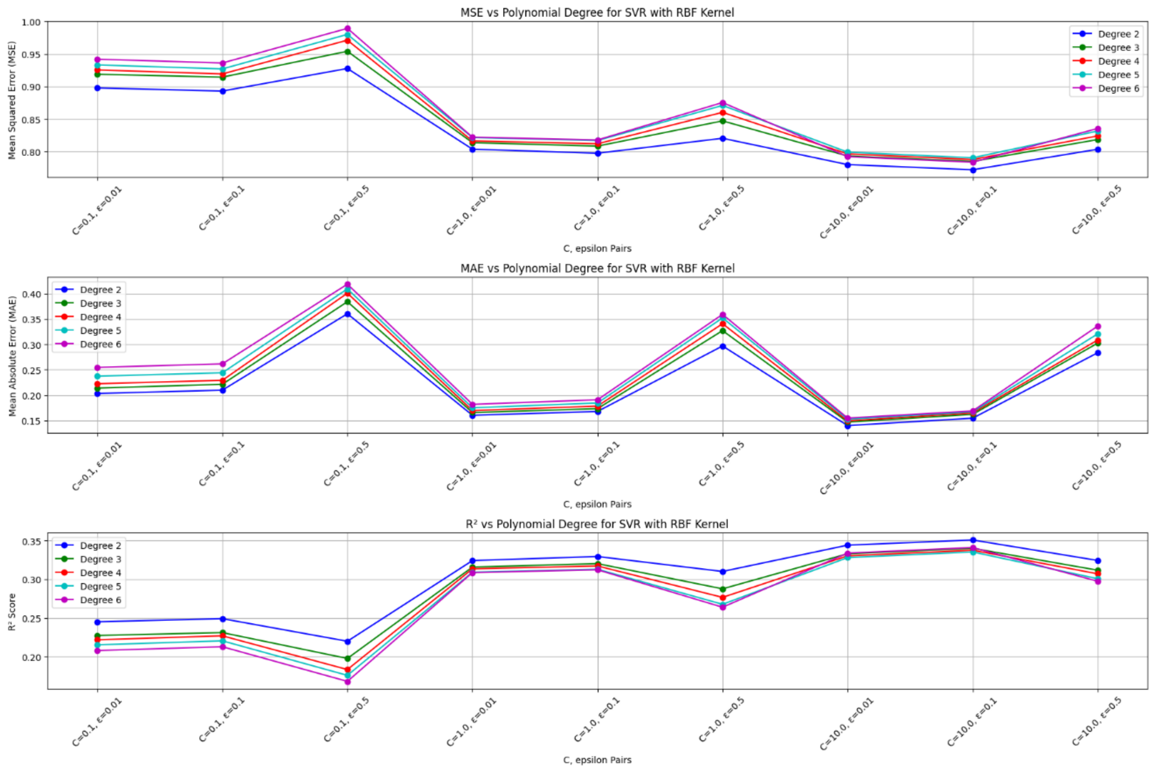


Figure 30: Results of RBF

The findings demonstrate how variations in C and epsilon values, as well as adjustments to the polynomial kernel's degree, affected model performance. Three important metrics were computed: R2 (Coefficient of Determination), MAE (Mean Absolute Error), and MSE (Mean Squared Error). The findings demonstrated that although more polynomial degrees led to greater complexity, they did not always perform better than lesser degrees. Performance was often better with larger values for C than with lower ones.

2.2.3 Polynomial Regression with Ridge Regularization and Performance Evaluation

In this code, we applied polynomial regression with Ridge regularization to the training and test data. First, we transformed the data into polynomial features with degrees ranging from 1 to 10 using PolynomialFeatures, then scaled them using StandardScaler. Next, we trained the model using TensorFlow with L2 regularization (Ridge) to reduce overfitting. We used the Adam optimizer to update the weights based on MSE loss. Finally, we evaluated performance using metrics like MSE and R-squared, and plotted the results to compare performance across different polynomial degrees.

Result:

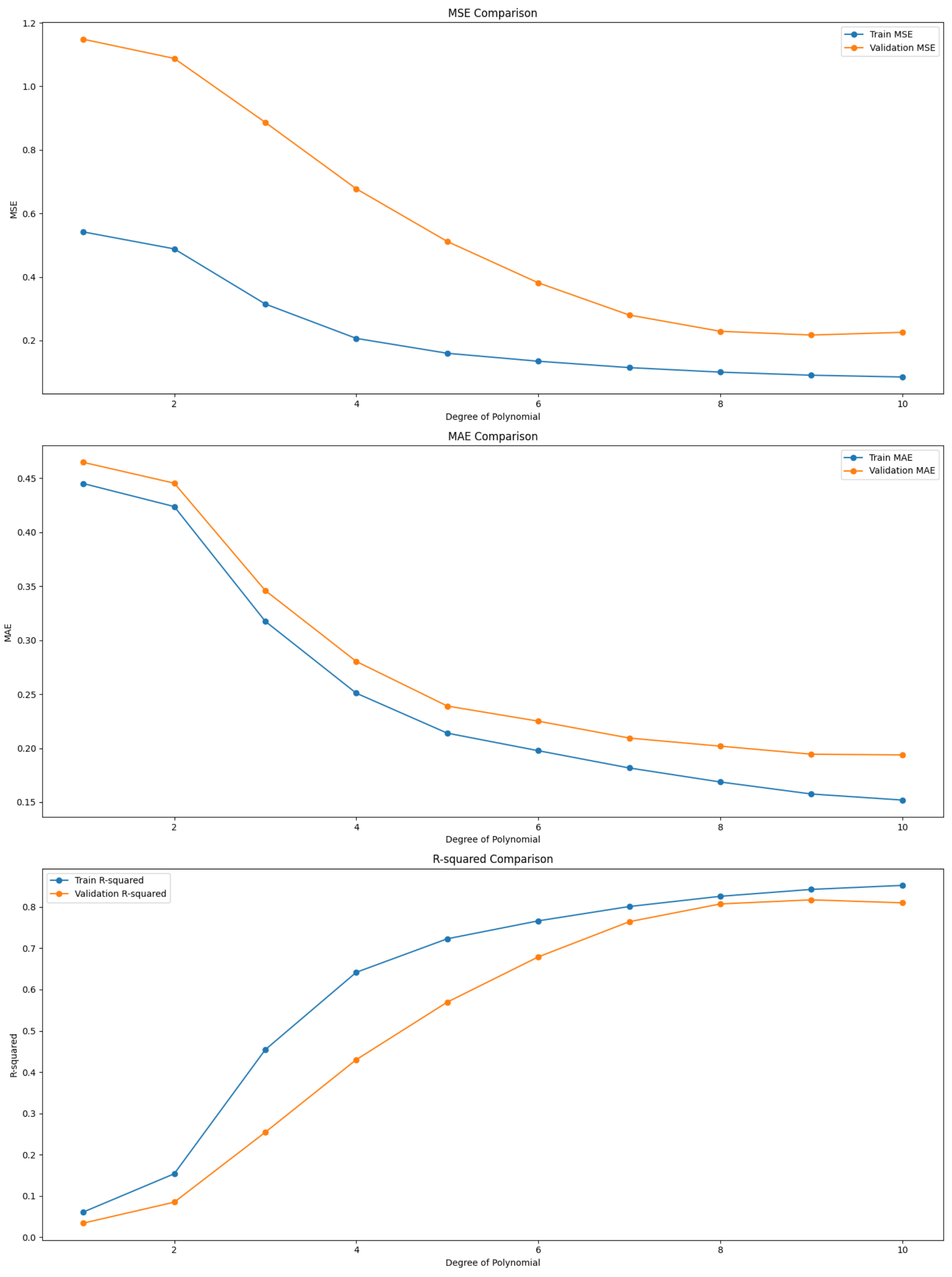


Figure 31: Results

Both the training and validation sets demonstrated a discernible improvement in model performance as the polynomial's degree grew. With low R-squared values at degree 1, the model's accuracy was poor; however, as the polynomial degree increased, these values improved. For instance, in both the training and validation sets, the model with the highest Mean Absolute Error (MAE) and the lowest Mean Squared Error (MSE) was at degree 10. Overall, the findings showed that by decreasing errors and raising prediction accuracy, increasing the polynomial degree enhanced the model's performance on both the training and validation data.

2.2.4 Impact of Polynomial Degree on Lasso Regression Model Performance

The method transforms data using polynomial features for degrees ranging from 1 to 10 and then uses a Lasso regression model (L1 regularization). Data normalization is done using StandardScaler. To avoid overfitting, the model is tweaked with early halting and trained with a modest learning rate using the Adam optimizer. For every degree, the MSE, MAE, and R2 are computed and recorded. Lastly, to show how the degree of polynomial transformation impacts the model's performance, the outcomes of these metrics are plotted across various polynomial degrees.

Result:

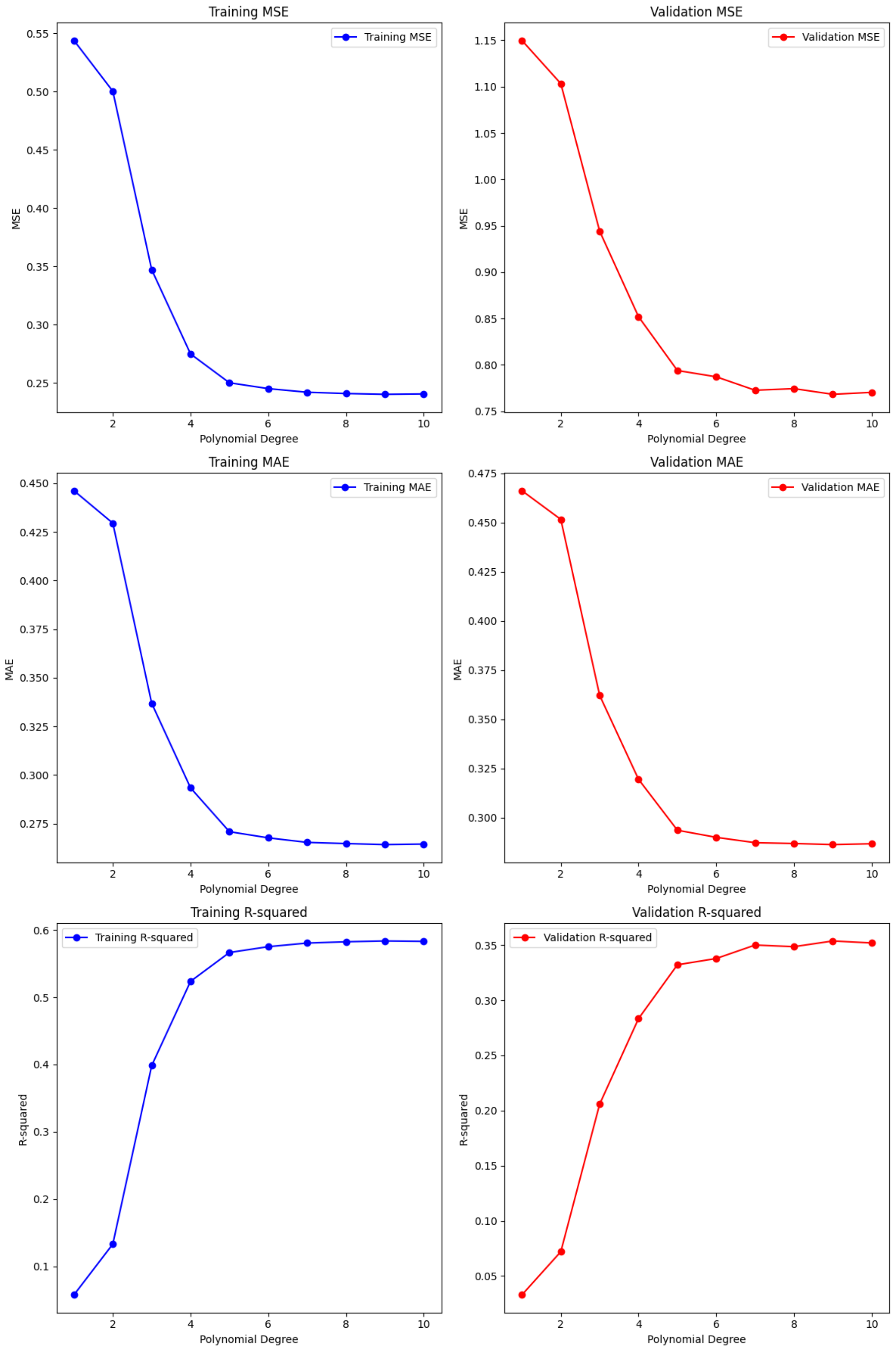


Figure 32: Results of Polynomial Degree on Lasso Regression

The findings indicate that performance gradually improves with increasing polynomial degree. R2 exhibits a discernible progressive improvement, reaching 0.5836 at degree 9, suggesting that the model explains over 58% of the variation in the data, despite MSE and MAE declining with increasing polynomial degrees. The variance between MSE and MAE in training and validation falls as the polynomial degree rises, indicating that the model has stabilized and improved in accuracy.

2.2.5 Comparison of Regression NonLinear Model

For comparison between models the results show varying performance across Polynomial Regression, Radial Basis Function (RBF), and Ridge models at different degrees**.**

- **Polynomial Regression:**
  - Degree 1: Shows the lowest performance, with an R^2 of 0.0610 on training data, and this gradually improves as the degree of the polynomial increases.
  - Degree 5: Performs well with an R^2 of 0.7241 on training data, while the test performance (Validation) is better than previous degrees, achieving an R^2 of 0.5749.
  - Degrees 6-10: Performance significantly improves as the degree increases, with a progressive rise in R^2 on both training and test data, showing better accuracy.

- **Radial Basis Function (RBF):**
  - Degree 2: Best results with C=10.0 and epsilon=0.1, achieving MSE = 0.7718 and R^2 = 0.3508.
  - Degree 3: RBF shows similar results to Degree 2, with comparable MSE, but slightly lower R^2 in most settings.
  - Degrees 4-6: Results improve slightly with settings like C=10.0 and epsilon=0.1, with R^2 improving on both training and test data.

- **Ridge Model:**
- Degrees 1-3: Shows moderate to weak performance with lower MSE and R^2 compared to the other models. Performance begins to improve at Degree 3.
- Degrees 4-6: Performance improves gradually with higher polynomial degrees, reaching an R^2 of 0.7229 on training data at Degree 5. However, test data results did not outperform other models significantly.

- **LASSO Regression:**
  - Degree 1-2: Poor performance with very low R^2, underfitting the data.
  - Degree 3-4: Improved performance, with better fitting and some generalization.
  - Degree 5-7: Strong performance, balancing training and validation well.
  - Degree 8-10: Best performance, with optimal fit and generalization.

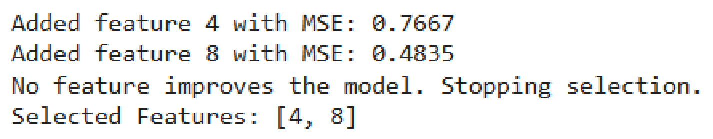
Conclusions:

- **Polynomial Regression** excels at certain degrees, especially at degrees 8-10.
- **RBF** shows good performance, particularly with C=10.0 and epsilon=0.1.
- **Ridge** is less effective compared to the other models in this context.
- **LASSO Regression** shows gradual improvement as the degree increases, with the best performance achieved at degrees 9 and 10, where it strikes a good balance between training and validation R^2, indicating strong generalization.

Step 3: Feature Selection with Forward Selection

Forward selection method to identify the most influential features of a multinomial regression model.We started with a blank model and added features gradually, one by one. At each step, the feature that minimized the mean square error (MSE) in the validation set was added. Polynomial features were generated and then scaled using StandardScaler. The model was trained using the Adam optimizer to adjust the weights. The code was iterated over training periods and the MSE was evaluated on the validation set, ultimately determining which features provided the best performance.

Result:



```
Added feature 4 with MSE: 0.7667
Added feature 8 with MSE: 0.4835
No feature improves the model. Stopping selection.
Selected Features: [4, 8]
```

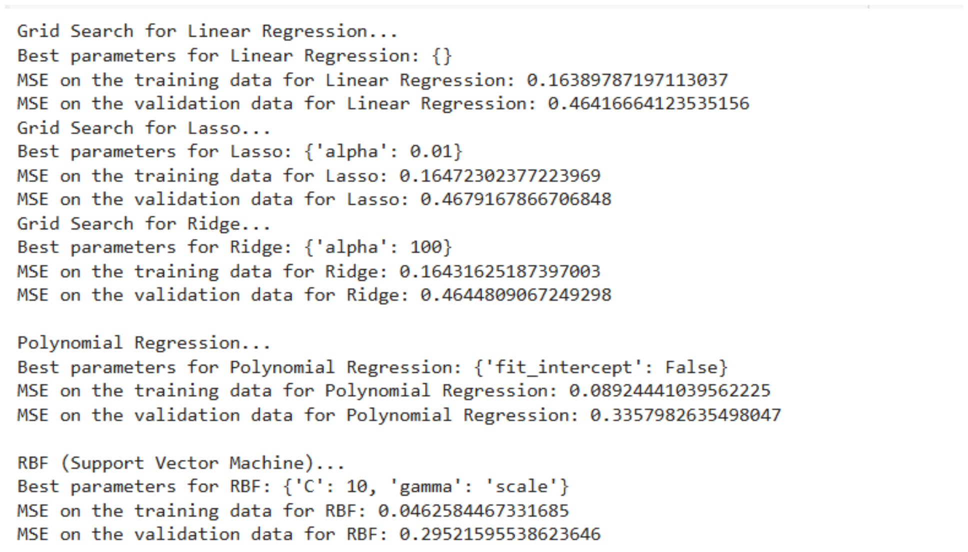
Figure 33: Results of Feature Selection

With an MSE of 0.7667, the algorithm determined that feature 4 was the initial addition. After that, it chose feature 8, which further decreased the MSE to 0.4835. The model's performance did not increase with any further features after this, therefore the selection process was terminated. This implies that, given the limitations, these two features—or their polynomial terms—have the greatest predictive power for the target variable. [4, 8] is the last feature set chosen.

Step 4: Hyperparameter Tuning with Grid Search

We searched for various hyperparameters for several regression models (such as Linear Regression, Lasso, Ridge, Polynomial Regression, and Support Vector Regression). We started by splitting the data into training and test sets using train_test_split. Then, GridSearchCV is applied to each model to find the best hyperparameters by evaluating the model's performance using MSE (Mean Squared Error) across different splits. After identifying the best hyperparameters for each model, the MSE is calculated for each model on both the training and test data.

Result:



```
Grid Search for Linear Regression...
Best parameters for Linear Regression: {}
MSE on the training data for Linear Regression: 0.16389787197113037
MSE on the validation data for Linear Regression: 0.46416664123535156
Grid Search for Lasso...
Best parameters for Lasso: {'alpha': 0.01}
MSE on the training data for Lasso: 0.16472302377223969
MSE on the validation data for Lasso: 0.4679167866706848
Grid Search for Ridge...
Best parameters for Ridge: {'alpha': 100}
MSE on the training data for Ridge: 0.16431625187397003
MSE on the validation data for Ridge: 0.4644809067249298

Polynomial Regression...
Best parameters for Polynomial Regression: {'fit_intercept': False}
MSE on the training data for Polynomial Regression: 0.08924441039562225
MSE on the validation data for Polynomial Regression: 0.3357982635498047

RBF (Support Vector Machine)...
Best parameters for RBF: {'C': 10, 'gamma': 'scale'}
MSE on the training data for RBF: 0.0462584467331685
MSE on the validation data for RBF: 0.29521595538623646
```

- **Linear Regression:** had no hyperparameters to tune, and achieved an MSE of 0.1639 on training data and 0.4642 on validation data.

- **Lasso Regression:** achieved the best result with alpha=0.01, yielding an MSE of 0.1647 on the training data and 0.4679 on the validation data.

- **Ridge Regression:** performed well with alpha=100, showing an MSE of 0.1643 on the training set and 0.4645 on the validation set.

- **Polynomial Regression:** (degree 2) performed better on the training data with an MSE of 0.0892, but its MSE on the validation set was 0.3358, indicating some overfitting.

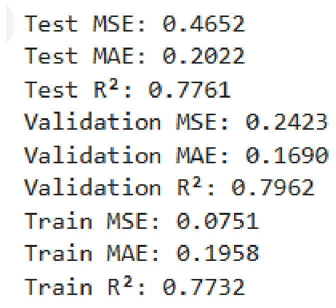
- **RBF (SVR):** with C=10 and gamma='scale' showed the best performance, achieving the lowest MSE of 0.0463 on the training data and 0.2952 on the validation data.

Step 5: Model Evaluation on Test Set

We implemented a multiple regression model using features extracted from polynomial transformations (polynomial features) with gradient descent to optimize the weights. After dividing the data into training, validation, and testing sets, the data is converted into fifth-degree polynomial features. The values are then normalized using standard scaling. Using error metrics like MSE, MAE, and R2 on the datasets, the model assesses its performance and optimizes the weights over several epochs using Gradient Descent.

Following training and assessment of the model's performance on the validation set, the model with the lowest MSE and highest R2 values was chosen as the top performer. The test dataset was then used to evaluate the chosen model's prediction accuracy and generalization capacity.

Result:



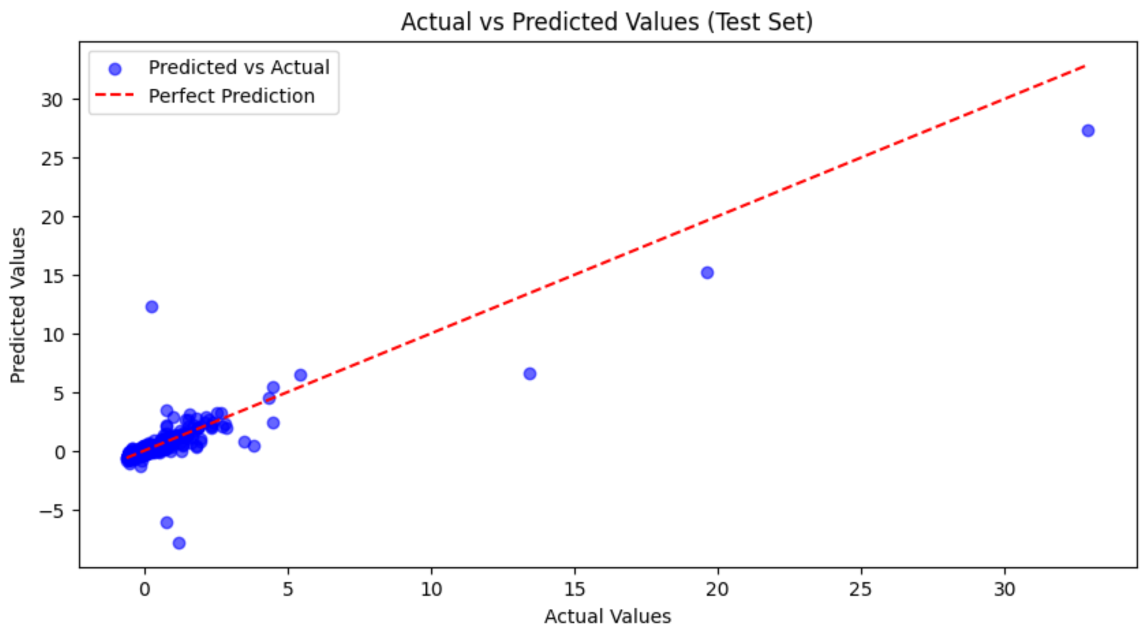
```
Test MSE: 0.4652
Test MAE: 0.2022
Test R²: 0.7761
Validation MSE: 0.2423
Validation MAE: 0.1690
Validation R²: 0.7962
Train MSE: 0.0751
Train MAE: 0.1958
Train R²: 0.7732
```

Figure 34: Results of Model Evaluation on Test Set

- **Performance on the test set**: MSE = 0.4652, MAE = 0.2022, and $R^2 = 0.7761$. This indicates that the model performs well in predicting the true values.

- **Performance on the validation set**: MSE = 0.2423, MAE = 0.1690, and $R^2 = 0.7962$, meaning the model fits the data effectively.

- **Performance on the training set**: MSE = 0.0751, MAE = 0.1958, and $R^2 = 0.7732$, showing that the model is well-trained with no clear signs of overfitting.

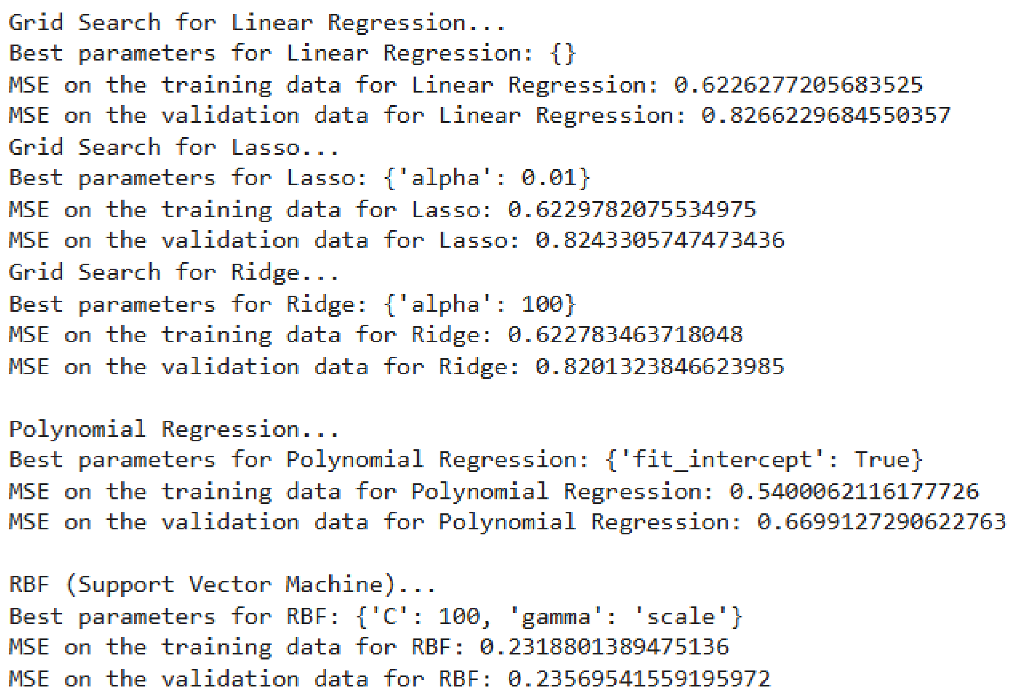


The contrast of the test set's actual and anticipated values is shown in the scatter plot. The link between these variables is shown by the blue points, and perfect predictions are shown by the red dashed line. A few outliers show small prediction errors, but the majority of points show strong model accuracy and are closely aligned with the red line. All things considered, the mode ؤl performs admirably in forecasting the true values.

Step 6: Regression Models for Predicting Car Top Speed

We chose an additional relevant target variable and built multiple regression models to predict the 'top_speed' in the vehicle dataset. Then, a series of models are evaluated using GridSearchCV to find the best hyperparameters for each model. The models tested include Linear Regression, Lasso Regression, Ridge Regression, Polynomial Regression, and RBF (Support Vector Machine). For each model, the Mean Squared Error (MSE) on the training and validation sets is calculated to assess the model's performance.

Result:



```
Grid Search for Linear Regression...
Best parameters for Linear Regression: {}
MSE on the training data for Linear Regression: 0.6226277205683525
MSE on the validation data for Linear Regression: 0.8266229684550357
Grid Search for Lasso...
Best parameters for Lasso: {'alpha': 0.01}
MSE on the training data for Lasso: 0.6229782075534975
MSE on the validation data for Lasso: 0.8243305747473436
Grid Search for Ridge...
Best parameters for Ridge: {'alpha': 100}
MSE on the training data for Ridge: 0.622783463718048
MSE on the validation data for Ridge: 0.8201323846623985

Polynomial Regression...
Best parameters for Polynomial Regression: {'fit_intercept': True}
MSE on the training data for Polynomial Regression: 0.5400062116177726
MSE on the validation data for Polynomial Regression: 0.6699127290622763

RBF (Support Vector Machine)...
Best parameters for RBF: {'C': 100, 'gamma': 'scale'}
MSE on the training data for RBF: 0.2318801389475136
MSE on the validation data for RBF: 0.23569541559195972
```

Figure 35: Results for Predicting Car Top Speed

The figure [38] shows that the RBF (SVR) model achieved the best performance