CSE 574-D: Introduction to Machine Learning Fall2024

Assignment 1

Instructor: Alina Vereshchaka **Deadline: Oct/10/2024**

Teammate1 Name: Nithin Pulla **Teammate1 Student ID:** 50591651
Teammate2 Name: Bharadwaj Gutha **Teammate2 Student ID:** 50602525

Part I: Data Analysis & Preprocessing

Dataset-Penguin:

a. Overview of the Dataset

The dataset contains information about penguins. Below are the key details:

- **Domain**: It consists Penguins data, including biological measurements and characteristics of different penguin species and locations.
- **Type of Data**: The dataset contains both categorical and numerical data.
- **Number of Samples**: There are 344 samples (rows) in the dataset.
- **Features**:
  - *species* (categorical): The species of the penguin (e.g., Adelie).
  - *island* (categorical): The island where the penguin was observed.
  - *calorie requirement* (numerical): The estimated calorie requirement of the penguin.
  - *average sleep duration* (numerical): The penguin's average sleep duration (in hours).
  - *bill_length_mm* (numerical): Length of the penguin's bill in millimeters.
  - *bill_depth_mm* (numerical): Depth of the penguin's bill in millimeters.
  - *flipper_length_mm* (numerical): Length of the penguin's flipper in millimeters.
  - *body_mass_g* (numerical): The body mass of the penguin in grams.
  - *gender* (categorical): The gender of the penguin.
  - *year* (numerical): The year the data was recorded.

b. Key Statistics

Key statistics for the dataset are as follows:

- **Mean Calorie Requirement**: 5270.00 kcal (with a standard deviation of 1067.96 kcal).
- **Mean Average Sleep Duration**: 10.45 hours (with a standard deviation of 2.27 hours).
- **Bill Length**: Mean = 45.49 mm, Standard Deviation = 10.82 mm.
- **Bill Depth**: Mean = 18.02 mm, Standard Deviation = 9.24 mm.
- **Flipper Length**: Mean = 197.76 mm, Standard Deviation = 27.76 mm.
- **Body Mass**: Mean = 4175.46 g, Standard Deviation = 858.71 g.
- **Missing Values**:
  - *species*: 11 missing values.

– *island*: 10 missing values.
– *bill_length_mm*: 7 missing values.
– *bill_depth_mm*: 11 missing values.
– *flipper_length_mm*: 8 missing values.
– *body_mass_g*: 5 missing values.
– *gender*: 17 missing values.
– *year*: 2 missing values.

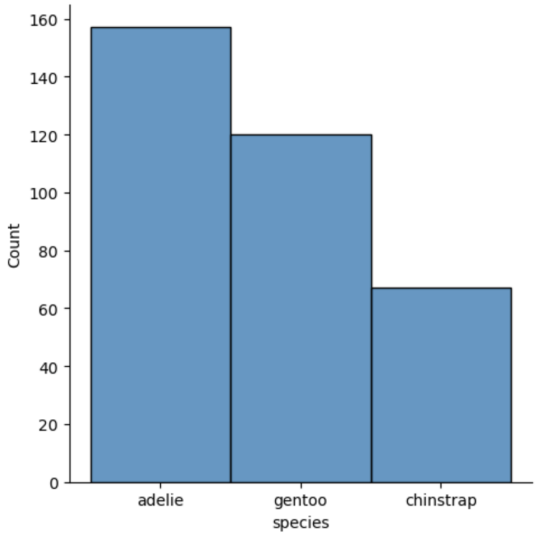


Figure 1: Count of Species

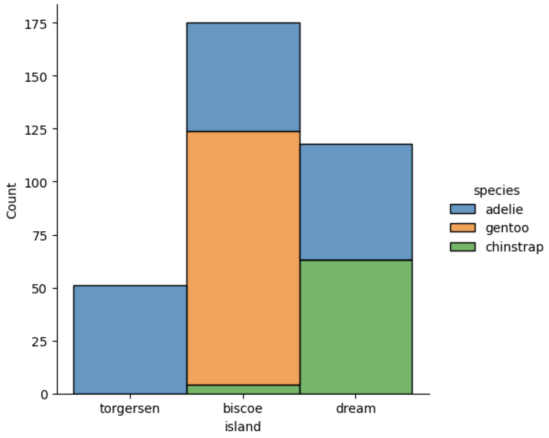


Figure 2: Species in Islands

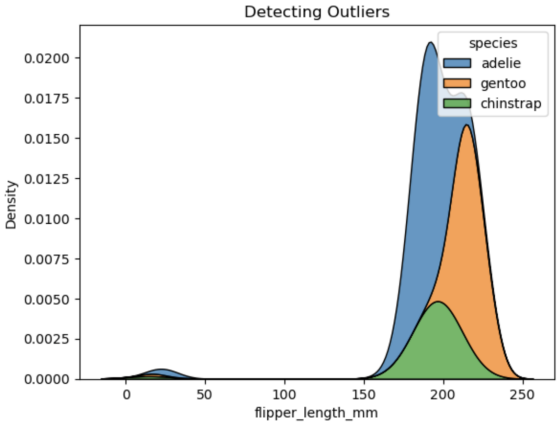


Figure 3: Detection of Outliers

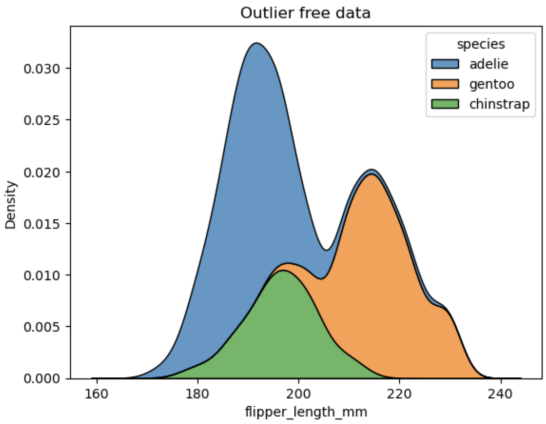


Figure 4: Outlier Free

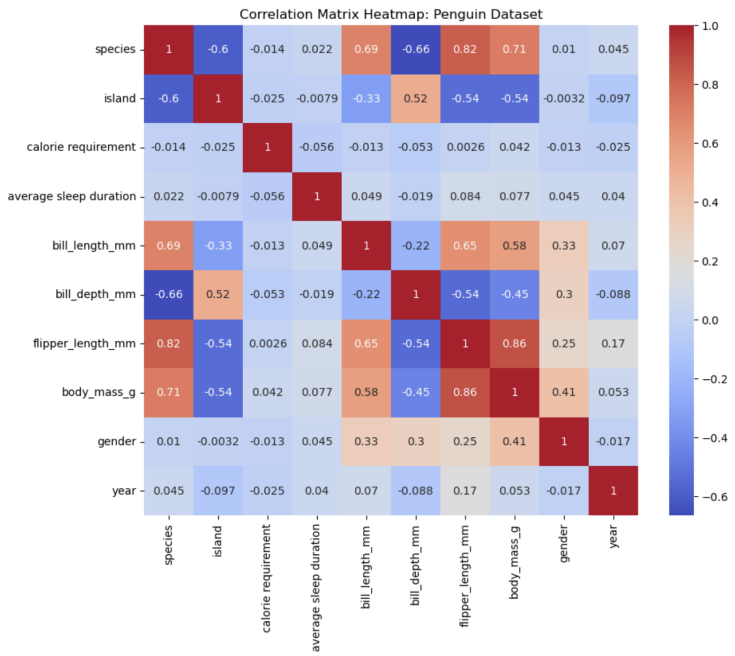


Figure 5: Correlation Matrix

Insights from Visualizations

- **Figure 1: Count of Species** – The Adelie species has the highest count, followed by Gentoo and Chinstrap species.
- **Figure 2: Species in Islands** – Biscoe Island hosts the highest diversity of species, while Torgersen and Dream Islands show lower species counts, with distinct species distributions.
- **Figure 3: Detection of Outliers** – There are significant outliers in the flipper length measurements across species, with some values deviating from the expected range.
- **Figure 4: Outlier-Free Data** – After removing outliers, the flipper length distribution across species is smoother and more consistent, with clear separation between species.
- **Figure 5: Correlation Matrix** - This matrix helps to choose the final features and target. Red indicating high correlation and blue indicating the lowest correlation.

Dataset-Wine

a. Overview of the Dataset

- *Domain*: This dataset pertains to wine different physicochemical properties of wines and quality.
- *Type of Data*: It contains both numerical and categorical data. Most of the features are continuous, while the ‘quality‘ feature is a numerical score, and ‘type‘ is a categorical variable.
- *Number of Samples*: 6,497 wine samples.
- *Features*: There are 13 features in total:
  - *fixed acidity* (numerical): The amount of fixed acids in the wine (e.g., tartaric acid) measured in grams per liter.
  - *volatile acidity* (numerical): The amount of acetic acid in the wine, which can lead to an unpleasant vinegar taste.

- *citric acid* (numerical): The amount of citric acid in the wine, which adds freshness and flavor.
- *residual sugar* (numerical): The amount of sugar remaining after fermentation, measured in grams per liter.
- *chlorides* (numerical): The amount of salt in the wine, measured in grams per liter.
- *free sulfur dioxide* (numerical): The amount of free SO2, which prevents microbial growth and oxidation, measured in milligrams per liter.
- *total sulfur dioxide* (numerical): The total amount of SO2, including both free and bound forms, measured in milligrams per liter.
- *density* (numerical): The density of the wine, which is closely related to the alcohol and sugar content.
- *pH* (numerical): The pH level of the wine, indicating its acidity or alkalinity.
- *sulphates* (numerical): A wine additive that acts as an antimicrobial and antioxidant, measured in grams per liter.
- *alcohol* (numerical): The alcohol content of the wine, measured as a percentage.
- *quality* (numerical): The wine's quality score, based on sensory data (range: 0–10).
- *type* (categorical): The type of wine (e.g., red or white).

b. Key Statistics

- **Fixed Acidity**: Mean = 7.22 g/L, Standard Deviation = 1.29 g/L.
- **Volatile Acidity**: Mean = 0.34 g/L, Standard Deviation = 0.16 g/L.
- **Citric Acid**: Mean = 0.32 g/L, Standard Deviation = 0.14 g/L.
- **Residual Sugar**: Mean = 5.44 g/L, Standard Deviation = 4.76 g/L.
- **Chlorides**: Mean = 0.05 g/L, Standard Deviation = 0.02 g/L.
- **Free Sulfur Dioxide**: Mean = 30.53 mg/L, Standard Deviation = 17.67 mg/L.
- **Total Sulfur Dioxide**: Mean = 115.74 mg/L, Standard Deviation = 56.52 mg/L.
- **Density**: Mean = 0.99 g/cm^3, Standard Deviation = 0.00 g/cm^3.
- **pH**: Mean = 3.22, Standard Deviation = 0.15.
- **Sulphates**: Mean = 0.53 g/L, Standard Deviation = 0.15 g/L.
- **Alcohol**: Mean = 10.49%, Standard Deviation = 1.19%.
- **Quality**: Mean = 5.82, Standard Deviation = 0.87.

Missing Values:

- No missing values in the dataset.

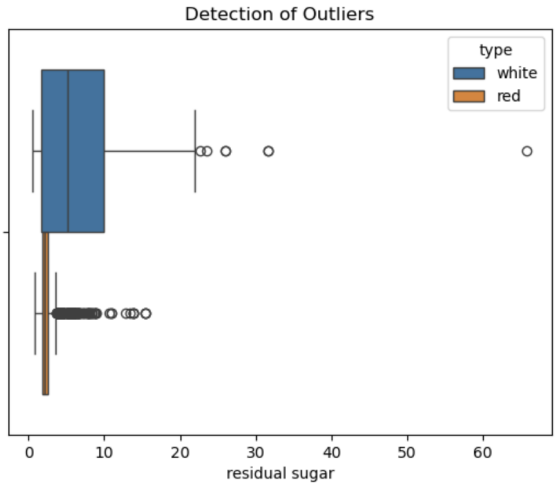


Figure 6: Detection of Outliers

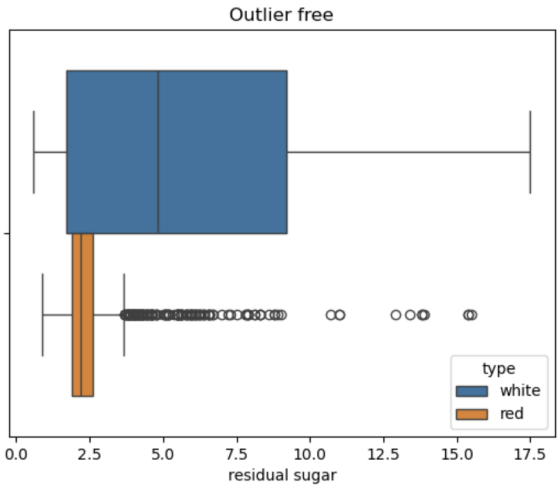


Figure 7: Outlier Free

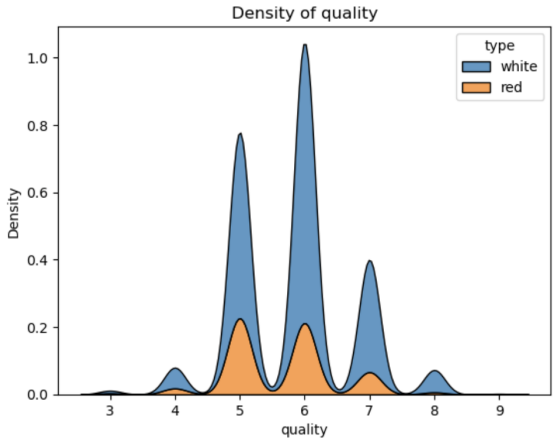


Figure 8: Centered values of Quality

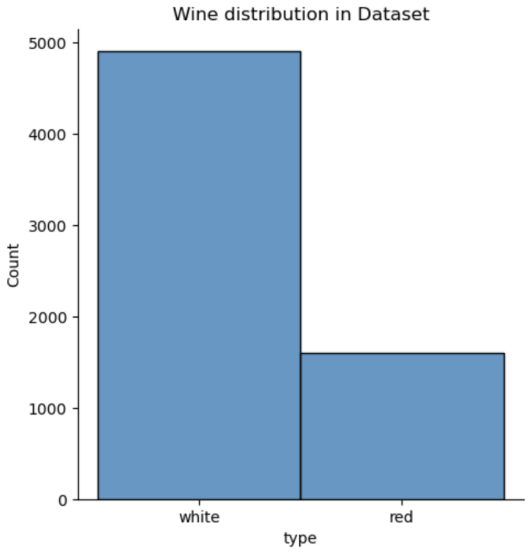


Figure 9: Red-White Wine distribution

Insights from Visualizations

- **Figure 6: Detection of Outliers** – There are noticeable outliers in the residual sugar content, particularly for white wines, with values extending beyond the typical range.
- **Figure 7: Outlier Free** – After removing the most extreme outliers, the residual sugar values for both red and white wines are more consistent, though some minor outliers remain.
- **Figure 8: Centered Values of Quality** – The distribution of quality scores shows that white wines have a wider and more varied spread, with clear peaks around scores of 5, 6, and 7, while red wines have a narrower range.
- **Figure 9: Red-White Wine Distribution** – The dataset is dominated by white wine samples, with nearly double the count of red wine samples.
- **Figure 10: Correlation Matrix** - This matrix helps to choose the final features and target. Red indicating high correlation and blue indicating the lowest correlation.

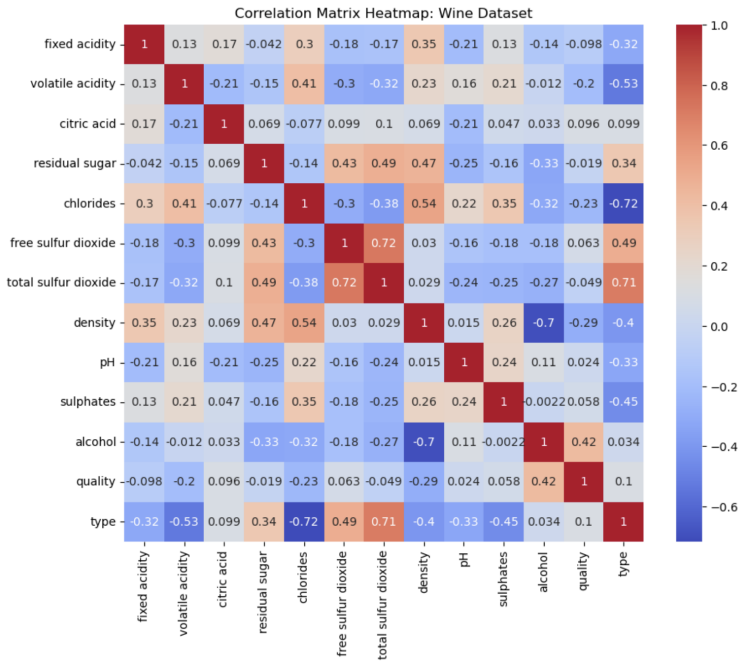


Figure 10: Correlation Matrix

Dataset-Diamond

a. Overview of the Dataset

- **Domain**: The dataset contains information about diamonds, focusing on their characteristics, quality, and pricing.
- **Data Type**: The dataset includes both categorical and numerical data.
- **Number of Samples**: 53,940 samples (rows).
- **Number of Features**: 13 features (columns).
- **Features**:
  - *carat* (numerical): The weight of the diamond in carats.
  - *cut* (categorical): The quality of the diamond's cut (e.g., Fair, Good, Very Good, Premium, Ideal).
  - *color* (categorical): The color grade of the diamond, where D is the best and J is the worst.
  - *clarity* (categorical): The clarity of the diamond (e.g., IF, VVS1, VVS2, VS1, VS2, SI1, SI2).
  - *average us salary* (numerical): The average salary in the U.S. (context-specific feature).
  - *number of diamonds mined (millions)* (numerical): The number of diamonds mined in millions.
  - *depth* (numerical): The total depth percentage of the diamond.
  - *table* (numerical): The width of the diamond's table as a percentage of its diameter.
  - *price* (numerical): The price of the diamond in dollars.
  - x (numerical): The length of the diamond in millimeters.
  - y (numerical): The width of the diamond in millimeters.
  - z (numerical): The depth of the diamond in millimeters.

b. Key Statistics

Average US Salary:

- Count: 53,940
- Mean: $39,521.99
- Standard Deviation: $5,486.89
- Minimum: $30,000
- 25th Percentile: $34,780
- Median (50th Percentile): $39,547.50
- 75th Percentile: $44,252
- Maximum: $48,999
- Missing Values: 0

Number of Diamonds Mined (in Millions):

- Count: 53,940
- Mean: 2.90 million
- Standard Deviation: 1.33 million
- Minimum: 0.6 million
- 25th Percentile: 1.75 million
- Median (50th Percentile): 2.91 million
- 75th Percentile: 4.05 million
- Maximum: 5.2 million
- Missing Values: 0

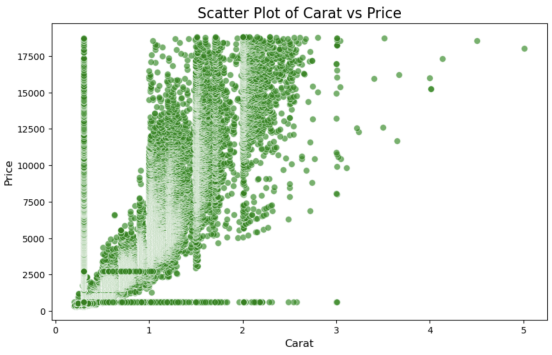


Figure 11: Price distribution of diamonds

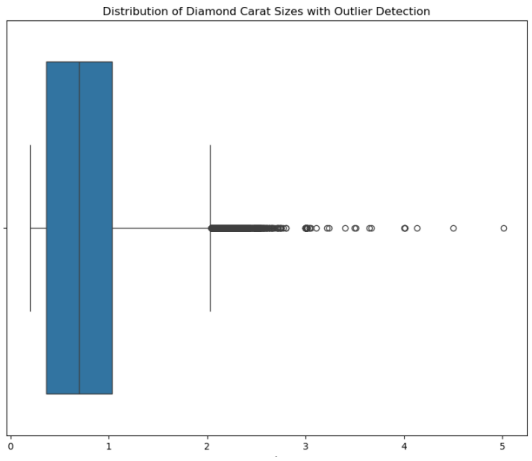


Figure 12: Carat distribution

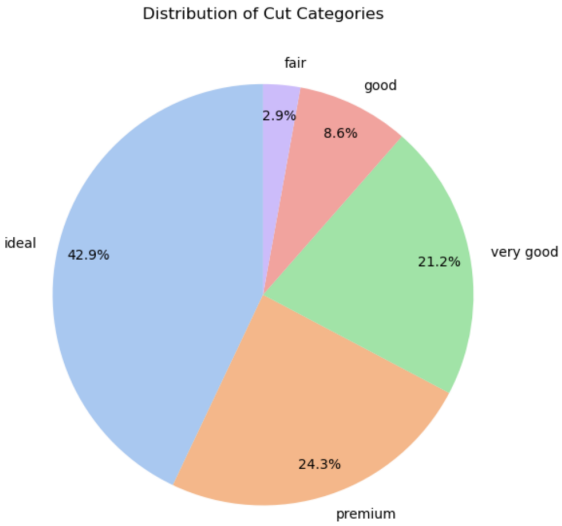


Figure 13: cut categorie distribution of diamonds

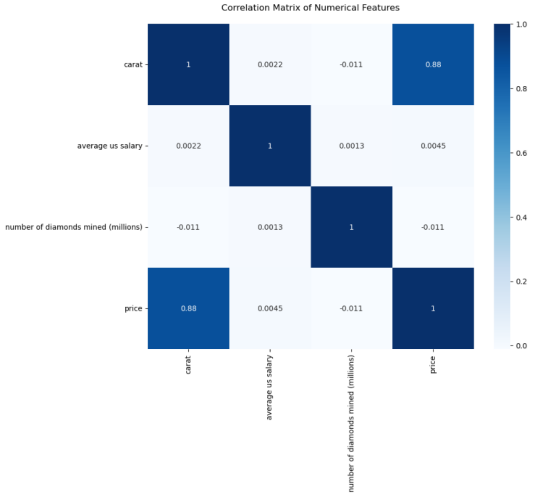


Figure 14: Coleration between numerical features

Insights from Visualizations

- **Figure 11: Price distribution of diamonds** – The scatter plot shows relation between the carat size and the price of diamonds. As the carat size increases, so does the price
- **Figure 12: Carat distribution** – The box plot shows the distribution of carat sizes for diamonds, As we can observe some outliers. Most diamonds are concentrated in the lower carat range (around 1-2 carats), with fewer diamonds in the higher carat range.
- **Figure 13: Cut category distribution of diamonds** – The pie chart displays the proportion of different cut categories among diamonds.where nearly half of the diamonds reside in ideal category .
- **Figure 14: Correlation between numerical features** – The correlation matrix reveals the relationships between various numerical features of the diamonds dataset. Darker squares indicate a higher correlation between features. This can help identify which features are most strongly relations.

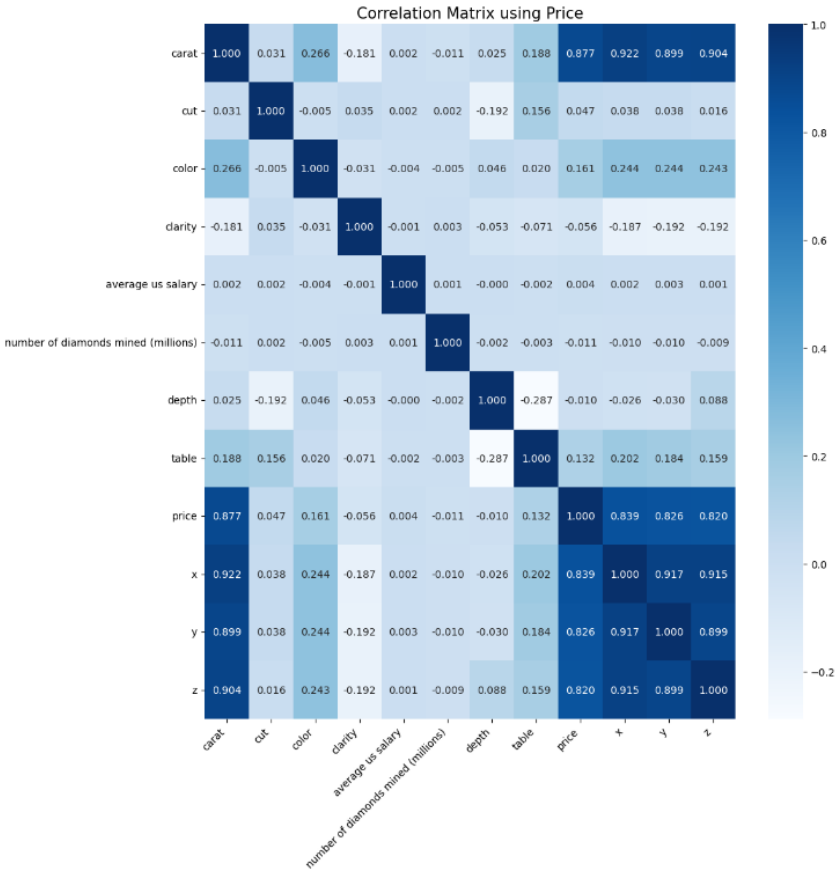


Figure 15: Correlation matrix on diamond dataset

Part II: Logistic Regression using Gradient Descent

Dataset used: Penguin

Analysis of Results

The graph and the provided results offer insight into how hyperparameters, such as the learning rate and number of iterations, influence the model's accuracy and training loss.

Learning Rate

The learning rate controls how quickly the model adapts to the problem. A lower learning rate tends to converge more slowly but may lead to a more stable final model, while a higher learning rate might speed up convergence but could risk overshooting the optimal solution.

- **Hyperparameter Set 1** {`'learning_rate': 0.01, 'iterations': 10000`}: The higher learning rate of 0.01 caused the model to reach a final accuracy of 85.51% with a final training loss of 0.5230, but it seems to plateau early.
- **Hyperparameter Set 2** {`'learning_rate': 0.001, 'iterations': 100000`}: A lower learning rate of 0.001 and more iterations allowed the model to achieve a slightly higher accuracy of 86.96% with a minimal reduction in final training loss to 0.5222. The slower learning rate provided better long-term improvements.

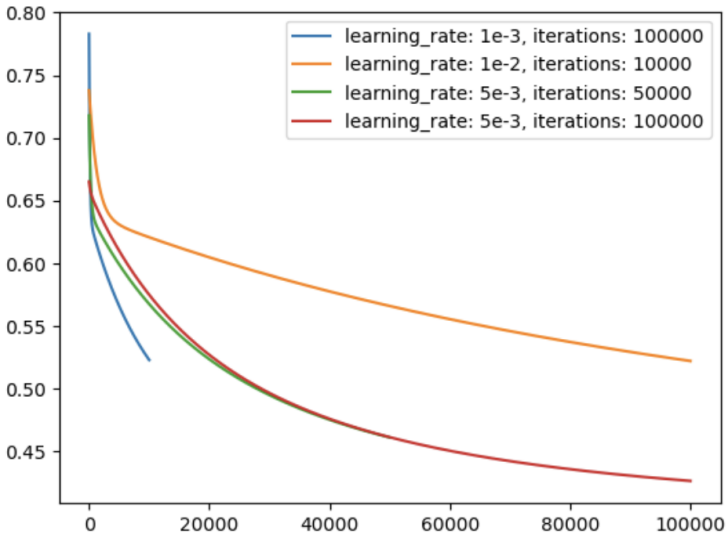


Figure 16: Loss Graph

- **Hyperparameter Sets 3 & 4** {`'learning_rate': 0.005`}: Both of these settings, with different iterations (50,000 vs. 100,000), showed the highest accuracy of 89.86%. However, the model with more iterations (Set 4) reached a lower final training loss of 0.4265 compared to 0.4612 in Set 3, indicating that additional iterations helped further refine the model's performance without changing accuracy.

Iterations

Increasing the number of iterations allows the model to learn more from the data, generally resulting in improved accuracy and reduced training loss.

- Set 2 (100,000 iterations, learning rate 0.001) shows that a lower learning rate but more iterations can outperform a higher learning rate in accuracy.
- Set 4 (100,000 iterations, learning rate 0.005) demonstrates the effectiveness of both a medium learning rate and sufficient iterations, leading to the lowest loss (0.4265) and maintaining high accuracy.

Best Accuracy

- A **learning rate of 0.005** combined with a high number of iterations (100,000) yields the best overall results in terms of both accuracy (89.86%) and the lowest final training loss (0.4265).
- The learning rate needs to be balanced carefully to avoid overshooting or slow convergence, and the number of iterations should be large enough to allow the model to fully converge for the best performance.

Benefits and Drawbacks of Using a Logistic Regression Model

Benefits

- **Simplicity and Interpretability**: Logistic Regression is a simple model that is easy to implement and interpret. The coefficients provide an indication of the influence of each feature on the prediction, which can be useful for understanding the relationship between input variables and the output.
- **Fast and Efficient**: Logistic Regression is computationally efficient and works well with smaller datasets. In this case, it quickly converged to good performance, with accuracy ranging from 85.51% to 89.86%, depending on the hyperparameters.

- **No Need for Scaling Features**: Logistic Regression does not require complex feature scaling (although it can benefit from normalization in some cases). This makes it easier to apply, especially in datasets with varying feature scales, like the diamond dataset.
- **Probabilistic Output**: Logistic Regression provides probabilities as output, which is useful for decision-making processes. It also allows you to adjust the decision threshold based on the specific needs of the task.
- **Regularization**: Logistic Regression supports regularization techniques (L1, L2) that can help prevent overfitting, especially when the number of features is large relative to the sample size.

Drawbacks

- **Linearity in the Features**: Logistic Regression assumes a linear relationship between the input features and the log-odds of the target variable. This assumption might be too simplistic for complex data, leading to underfitting. If the relationship between the variables is nonlinear, Logistic Regression may not perform well without feature engineering or transformation.
- **Performance on Large and Complex Datasets**: Logistic Regression may struggle to capture patterns in large or complex datasets with high-dimensional features or non-linear relationships. In these cases, more complex models like Decision Trees, Random Forests, or Neural Networks may be required.
- **Sensitive to Outliers**: Logistic Regression can be sensitive to outliers, as they can significantly impact the estimation of the model's coefficients, particularly when using gradient-based optimization methods. Proper handling of outliers, like removal or transformation, may be necessary before applying the model.
- **Limited Flexibility**: While Logistic Regression works well for binary classification tasks, it may not be the best option for more complex classification problems, such as multi-class classification, without modifications (e.g., using one-vs-rest or one-vs-one strategies).
- **Dependence on Hyperparameters**: As shown in the given data, the performance of Logistic Regression can depend heavily on hyperparameters like the learning rate and number of iterations. Selecting the appropriate values for these parameters can be challenging and may require extensive tuning.

Conclusion

Logistic Regression is a great starting point for classification tasks due to its simplicity, efficiency, and interpretability. However, it may not be suitable for more complex datasets with nonlinear relationships or high-dimensional data without additional feature engineering or hyperparameter tuning.

Part III: Linear & Ridge Regressions using OLS

Dataset-Diamond

a. Overview of the Dataset

- **Domain**: The dataset contains information about diamonds, focusing on their characteristics, quality, and pricing.
- **Data Type**: The dataset includes both categorical and numerical data.
- **Number of Samples**: 53,940 samples (rows).
- **Number of Features**: 13 features (columns).
- **Features**:

- *carat* (numerical): The weight of the diamond in carats.
- *cut* (categorical): The quality of the diamond's cut (e.g., Fair, Good, Very Good, Premium, Ideal).
- *color* (categorical): The color grade of the diamond, where D is the best and J is the worst.
- *clarity* (categorical): The clarity of the diamond (e.g., IF, VVS1, VVS2, VS1, VS2, SI1, SI2).
- *average us salary* (numerical): The average salary in the U.S. (context-specific feature).
- *number of diamonds mined (millions)* (numerical): The number of diamonds mined in millions.
- *depth* (numerical): The total depth percentage of the diamond.
- *table* (numerical): The width of the diamond's table as a percentage of its diameter.
- *price* (numerical): The price of the diamond in dollars.
- x (numerical): The length of the diamond in millimeters.
- y (numerical): The width of the diamond in millimeters.
- z (numerical): The depth of the diamond in millimeters.

b. Key Statistics

Average US Salary:

- Count: 53,940
- Mean: $39,521.99
- Standard Deviation: $5,486.89
- Minimum: $30,000
- 25th Percentile: $34,780
- Median (50th Percentile): $39,547.50
- 75th Percentile: $44,252
- Maximum: $48,999
- Missing Values: 0

Number of Diamonds Mined (in Millions):

- Count: 53,940
- Mean: 2.90 million
- Standard Deviation: 1.33 million
- Minimum: 0.6 million
- 25th Percentile: 1.75 million
- Median (50th Percentile): 2.91 million
- 75th Percentile: 4.05 million
- Maximum: 5.2 million
- Missing Values: 0

Model Evaluation

Loss Value

- **Linear Regression**:
  - MSE on training set: 0.025357178020076636
  - MSE on test set: 0.005310063673178923
- **Ridge Regression**:
  - MSE on training set: 0.02535717815694855
  - MSE on test set: 0.00531058257753823

Observation: The MSE values for both models are nearly the same , with minor differences in performance between the training and testing sets, and using ridge regression provided no significant increments over the linear model for this dataset.

Predictions vs Actual plots

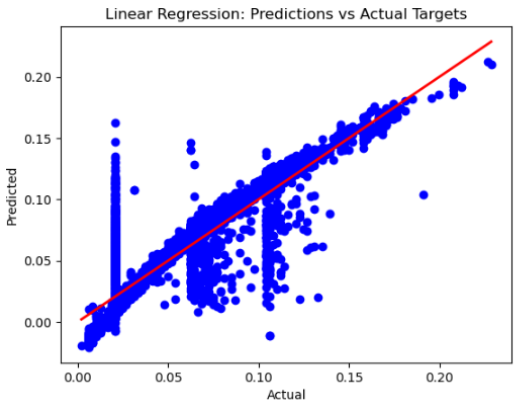


Figure 17: Linear regression

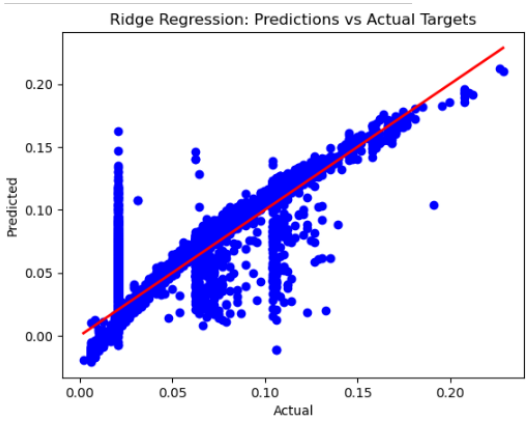


Figure 18: Ridge regression

Observation: The MSE values for both models are nearly the same , with minor differences in performance between the training and testing sets, The residuals, represented around the line, show a similar stricture in both models

Discussion

Benefits and drawbacks of using OLS for weight computation.

Benefits:

- OLS performs well with small datasets. It can provide accurate weight estimates in such cases.
- OLS is a computationally efficient method. It provides a closed-form solution for the weights, which can build linear relationships between features and the target variable.
- OLS does not require hyperparameter tuning . The solution is purely determined by the data.

Drawbacks:

- OLS is highly sensitive to outliers because it minimizes the squared residuals. Large errors (outliers) have a high influence in affecting the result
- If the features are highly correlated (multicollinearity), OLS can produce large and unstable weight estimates, which may result in overfitting.
- OLS takes the linearity of the connection between the independent and dependent variables as given. The OLS model could yield erroneous results if this presumption is not true.

Strengths of Linear Regression:

- Linear regression is easy to implement. The relationship between the input features and the target variable is modeled is represented as a straight line
- As we use OLS the computation is quick , which makes it easy for implementation in large data sets
- When We have linear relation between the features and target variable , we can get accurate predictions

Drawbacks of Linear Regression:

- The model assumes that relation between the features and the target is always linear if the relation is not linear then it leads to poor predictions.
- The model is very sensitive to outliers , if outliers are present in the data then it affects the model which would lead to large variation in results.
- If the data has high multicollinearity that is features are interdependent of each other then it leads to variance in the predicted value
- Linear regression does not perform well when there are more features than data points

Motivation for Using L2 Regularization (Ridge Regression):

In linear regression, we aim to minimize the sum of squared errors between the predicted and actual values but when we have high multicollinearity or when the model over-fits regression might result in huge variance

Ridge Regression helps solve this problem by adding a term to the cost function . The purpose of this adjustment, called L2 regularization, is to keep the model simpler, improve its performance on new data, and avoid overfitting. The updated cost function in Ridge Regression looks like this:

$$\text{Ridge Cost Function} : J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Where:

- The first part, $\frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$, represents the sum of squared residuals (errors).
- The second part, $\frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$, is the regularization term, which restricts large coefficients $\mathbf{w}$.

How Ridge Regression Improves Upon Linear Regression:

- In linear regression, the model might overfit the , especially when the dataset is small or there is high-dimensional data.This overfitting data is handled by ridge regression
- In general linear regression, multicollinearity (high correlation between features) can result in the large coefficient , which causes high variance. Ridge regression adds penalty and reduces the multicollinearity results in stable coefficients
- By reducing the high varience, ridge regression provides more stable estimates even if multicollinearity is present.
- Ridge regression reduces the model's overfitting problem
- The regularization variable λ can help to control bias and variance, allowing ridge regression to fir the model properly

Limitations of Ridge Regression Compared to Linear Regression:

- If the regularization variable λ is too large, ridge regression can underfit the model and would not capture the dependent variable in this situaltion
- Ridge regression does not perform feature selection , it retains all the features even the features are irrelevant.
- If the features are not normalized properly , in this case the model will be sensitive to large data which will result in irregular results.

Part IV: Elastic Net Regression using Gradient Descent

Dataset-Wine

a. Overview of the Dataset

- *Domain*: This dataset pertains to wine different physicochemical properties of wines and quality.
- *Type of Data*: It contains both numerical and categorical data. Most of the features are continuous, while the 'quality' feature is a numerical score, and 'type' is a categorical variable.
- *Number of Samples*: 6,497 wine samples.
- *Features*: There are 13 features in total:
  - *fixed acidity* (numerical): The amount of fixed acids in the wine (e.g., tartaric acid) measured in grams per liter.
  - *volatile acidity* (numerical): The amount of acetic acid in the wine, which can lead to an unpleasant vinegar taste.
  - *citric acid* (numerical): The amount of citric acid in the wine, which adds freshness and flavor.
  - *residual sugar* (numerical): The amount of sugar remaining after fermentation, measured in grams per liter.
  - *chlorides* (numerical): The amount of salt in the wine, measured in grams per liter.
  - *free sulfur dioxide* (numerical): The amount of free SO2, which prevents microbial growth and oxidation, measured in milligrams per liter.
  - *total sulfur dioxide* (numerical): The total amount of SO2, including both free and bound forms, measured in milligrams per liter.

- *density* (numerical): The density of the wine, which is closely related to the alcohol and sugar content.
- *pH* (numerical): The pH level of the wine, indicating its acidity or alkalinity.
- *sulphates* (numerical): A wine additive that acts as an antimicrobial and antioxidant, measured in grams per liter.
- *alcohol* (numerical): The alcohol content of the wine, measured as a percentage.
- *quality* (numerical): The wine's quality score, based on sensory data (range: 0–10).
- *type* (categorical): The type of wine (e.g., red or white).

b. Key Statistics

- **Fixed Acidity**: Mean = 7.22 g/L, Standard Deviation = 1.29 g/L.
- **Volatile Acidity**: Mean = 0.34 g/L, Standard Deviation = 0.16 g/L.
- **Citric Acid**: Mean = 0.32 g/L, Standard Deviation = 0.14 g/L.
- **Residual Sugar**: Mean = 5.44 g/L, Standard Deviation = 4.76 g/L.
- **Chlorides**: Mean = 0.05 g/L, Standard Deviation = 0.02 g/L.
- **Free Sulfur Dioxide**: Mean = 30.53 mg/L, Standard Deviation = 17.67 mg/L.
- **Total Sulfur Dioxide**: Mean = 115.74 mg/L, Standard Deviation = 56.52 mg/L.
- **Density**: Mean = 0.99 g/cm^3, Standard Deviation = 0.00 g/cm^3.
- **pH**: Mean = 3.22, Standard Deviation = 0.15.
- **Sulphates**: Mean = 0.53 g/L, Standard Deviation = 0.15 g/L.
- **Alcohol**: Mean = 10.49%, Standard Deviation = 1.19%.
- **Quality**: Mean = 5.82, Standard Deviation = 0.87.

Missing Values:

- No missing values in the dataset.

Model Evaluation

The model is developed to predict pH levels of the wine, for given values of features other than *quality, density.*

Loss Value

The loss for each model combination is tracked and reported. The best loss value is found by comparing across different combinations of weight initialization methods, epochs, and stopping criteria (threshold). For each combination, the *Mean Squared Error (MSE)* and *Best Loss* are reported.

Loss and MSE for Each Hyperparameter Set

Note: The best performed combination is highlighted with bold text.

Threshold	Epochs	Weight Initialization	MSE	Regularized Loss
0.001	1000	Xavier	0.02637	0.01435
0.001	1000	Random	0.02822	0.01433
0.001	1000	Zero	0.01693	0.00886
0.001	5000	Xavier	0.01641	0.00862
0.001	5000	Random	0.01641	0.00861
0.001	5000	Zero	0.01638	0.00860
None	1000	Xavier	0.02928	0.01490
None	1000	Random	0.04536	0.02322
None	1000	Zero	0.01693	0.00886
None	**5000**	**Xavier**	**0.01637**	**0.00859**
None	5000	Random	0.01655	0.00870
None	**5000**	**Zero**	**0.01637**	**0.00859**

Table 1: MSE and Best Loss for Different Hyperparameter Combinations

Effects of Different Weight Initialization Methods

- **Xavier Initialization**: Xavier initialization led to relatively strong performance in both cases with a best loss of **0.00859**. This method helps the model converge faster, providing smoother loss curves across different epochs. For example, with 5000 epochs and no threshold, the model achieved a mean squared error (MSE) of **0.01637**.
- **Random Initialization**: Random initialization exhibited more variability in the results, requiring more epochs to achieve comparable performance. While the best loss with 5000 epochs and a threshold of 0.001 was **0.00861**, its MSE was slightly higher at **0.01641**, showing more instability compared to Xavier initialization.
- **Zero Initialization**: Surprisingly, zero initialization performed comparably well in this specific experiment, achieving the best loss of **0.00859** with 5000 epochs and no threshold. The results were very much similar to the Xavier Weight initialization. The model using zero initialization with early stopping at 2268 epochs, where the gradient dropped below the threshold, also performed strongly with an MSE of **0.01637**. This suggests that the combination of epochs and stopping criteria can influence the effectiveness of zero initialization.
- **Overall** Zero Weight initialization and Xavier Weight initialization gave similar results in the convergence plots. Zero Weight initialization converged faster and was aggressive when compared to Xavier Weight initialization. Xavier had smooth convergence and required more iterations, the difference is not significant.

Insights on Weight Initialization:

- **Xavier Initialization**: Provided consistently good convergence and faster loss reduction across different settings, achieving the best MSE and loss performance overall. The best performance was achieved with 5000 epochs and no threshold, producing a best loss of **0.00859**.
- **Random Initialization**: Showed higher variability and required longer training to approach the same performance as Xavier. While competitive, its convergence characteristics were more unstable.
- **Zero Initialization**: While typically expected to underperform, in this experiment, it managed to match Xavier initialization in certain cases. The best performance was achieved with 5000 epochs and no threshold, producing a best loss of **0.00859**.

Effects of Stopping Criteria

- **Threshold-Based Early Stopping**: The models that used a threshold for early stopping performed well, particularly when the threshold was set at **0.001**. This prevented overfitting and reduced unnecessary training time. For example, zero initialization with 5000 epochs and early stopping at 2268 epochs had a final MSE of **0.01638** and a best loss of **0.00860**, illustrating that the model converged early without sacrificing accuracy.
- **No Stopping Criterion**: The models that did not use early stopping ran for the full 5000 epochs. While this approach allowed the models to converge fully, it didn't necessarily provide significant gains. For instance, zero initialization with no threshold still produced the best overall performance, with an MSE of **0.01637** and a best loss of **0.00859**.

Discussion

Elastic Net Regularization vs. L1 and L2 Regularization

- **L1 Regularization (Lasso)**: Encourages sparsity in the model by forcing many coefficients to be zero, effectively performing feature selection. However, it can struggle when highly correlated variables are present, as it tends to randomly assign weight to one feature and drop the others.
- **L2 Regularization (Ridge)**: Encourages smaller weight values without forcing them to zero. This works well for multicollinear data but doesn't perform feature selection.
- **Elastic Net**: Combines the strengths of both L1 and L2 regularization. It benefits from L1's sparsity (feature selection) and L2's smoothness (handling correlated features). In your dataset (wine data), Elastic Net regularization can help prevent overfitting and handle any multicollinearity among features, which is common in datasets with many chemical properties like wine data.

Advantages of Elastic Net:

- It provides a balance between feature selection (via L1) and regularization (via L2), making it useful in scenarios with correlated features.
- It avoids some of the pitfalls of using L1 or L2 regularization alone, which may lead to either too few features or ineffective regularization.

Benefits and Drawbacks of Elastic Net and Gradient Descent

Benefits:

- **Elastic Net**: Helps in reducing overfitting while selecting the most relevant features, making it a good choice for models with many input features and potential multicollinearity.
- **Gradient Descent**: Is a powerful optimization algorithm that helps the model converge to a solution, especially when dealing with large datasets. It efficiently minimizes the loss function.

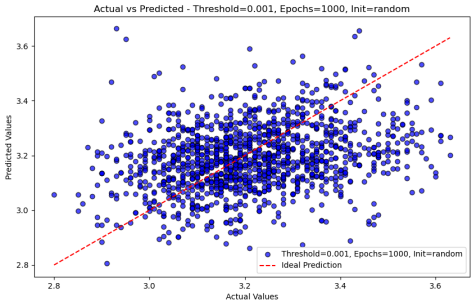
Drawbacks:

- **Elastic Net**: Requires tuning two hyperparameters: one for L1 and one for L2 regularization. This can increase the complexity of model tuning.
- **Gradient Descent**: Can be sensitive to learning rates and may require careful tuning to ensure convergence. Too small a learning rate leads to slow convergence, while too large a learning rate can cause divergence or overshooting the minimum.

Conclusion

Elastic Net Regularization is highly suitable for the wine dataset, where multicollinearity may exist, and feature selection is important. **Xavier Initialization** is the most reliable initialization method for ensuring smooth convergence. **Early stopping** via a threshold is beneficial in preventing overfitting and speeding up the training process when the model has learned most of the relevant patterns.

Appendix A: Prediction plots



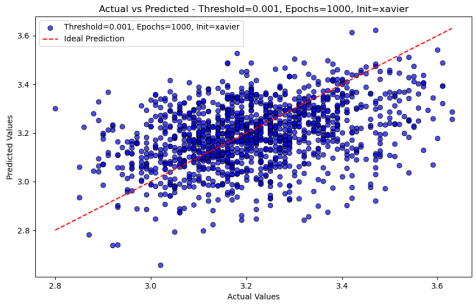
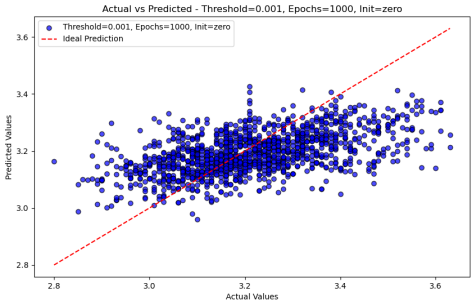


Figure 19: Row 1: Actual vs Predicted Plots for different Hyperparameter sets



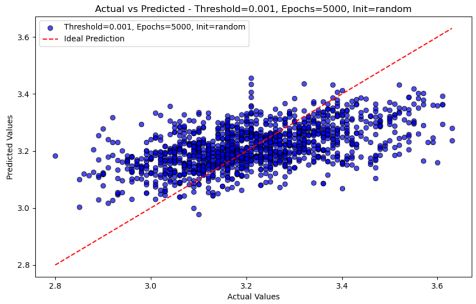
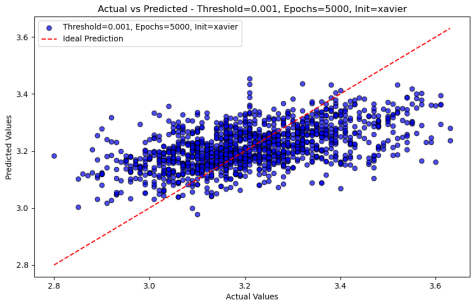


Figure 20: Row 2: Actual vs Predicted Plots for different Hyperparameter sets



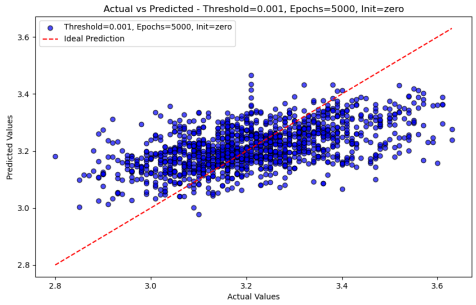
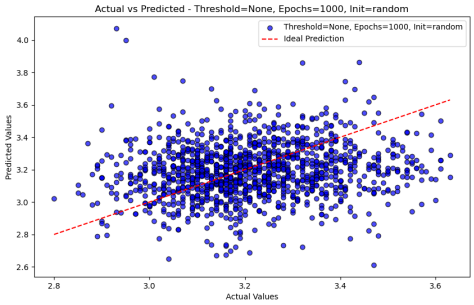


Figure 21: Row 3: Actual vs Predicted Plots for different Hyperparameter sets



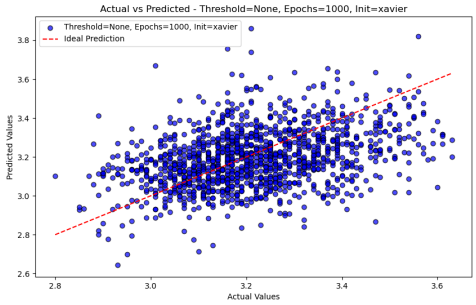
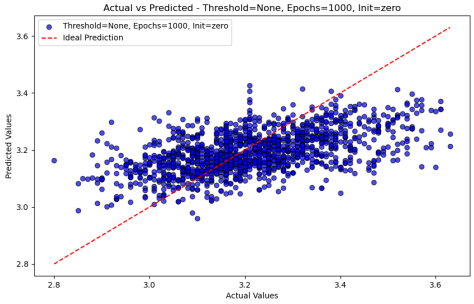


Figure 22: Row 4: Actual vs Predicted Plots for different Hyperparameter sets



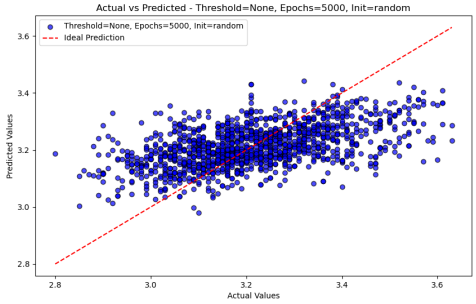
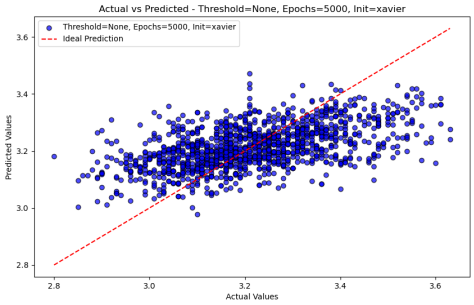
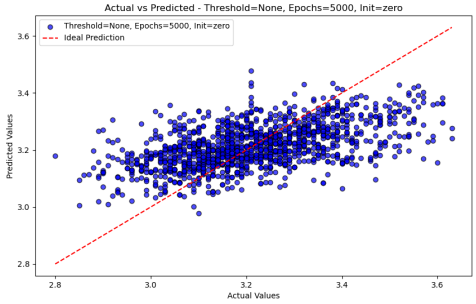


Figure 23: Row 5: Actual vs Predicted Plots for different Hyperparameter sets





Appendix B: Loss Plots

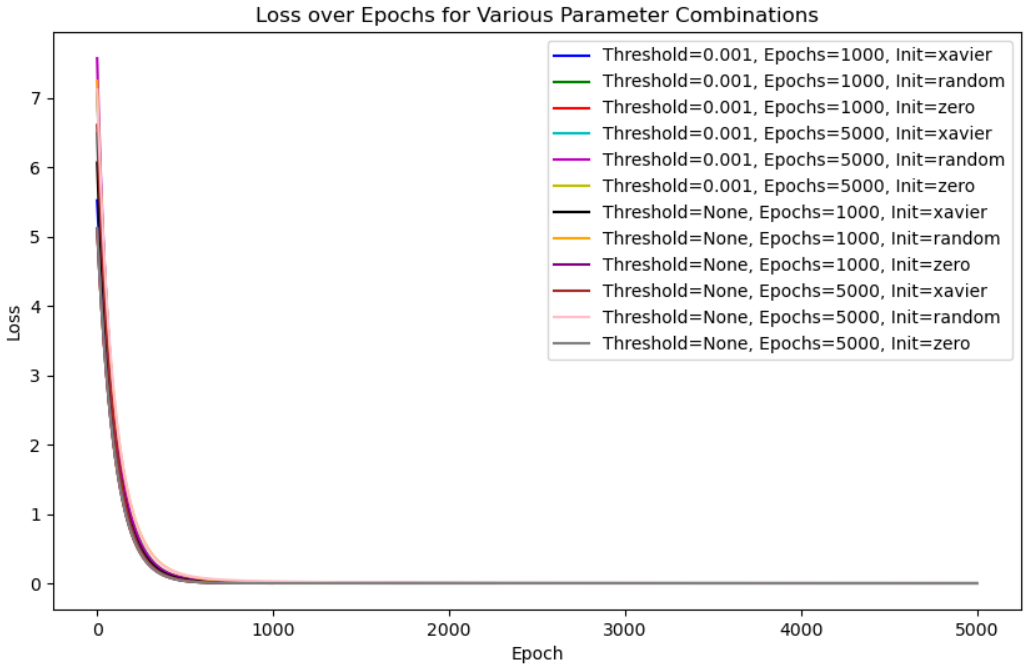
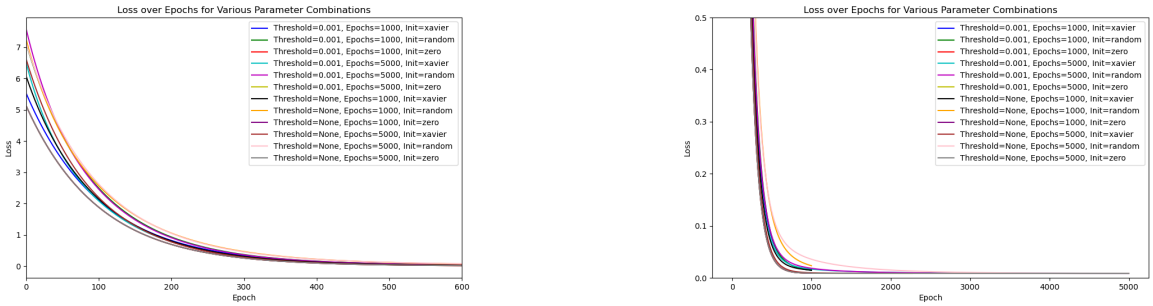


Figure 25: Loss Plots of various hyperparameters set



(a) Limited X to maximize visualization

(b) Limited y to maximize visualization

Figure 26: Maximized Visualization

References

[1] Numpy - Web Resources

[2] CSE574: npulla2_bgutha_assignment1_checkpoint - Course Resources

[3] CSE574: Lecture Slides - Course Resources

[4] Outlier Handling - Web Resources

[5] One hot encoding - Web Resources

[6] Data preprocessing - Web Resources

[7] Batch Normalization - Web Resources

[8] Elasticnet regression - Web Resources

[9] Ridge regression - Web Resources

[10] Linear regression - Web Resources

[11] OLS - Web Resources

[12] Stopping Criteria - Web Resources

[13] Stopping Criteria - Web Resources