ASSIGNMENT-1 REPORT

PART -1

TASK 1 : PENGUIN DATASET PREPROCESSING

1.1 Introduction :

This report details the steps taken to preprocess the Penguin dataset, which is important for preparing the data for analysis. ˜Proper data preprocessing helps improve model performance by fixing missing values, converting categorical data into numerical form, and normalizing features, making the dataset ready for effective machine learning use.

1.2 Package and Library Usage :

The packages used are :

Pandas (pd): Used to create Data Frame and handle the manipulations.

NumPy **(**np**)**:Used for percentile calculations, and mathematical computations.

Matplotlib**:** Used for plotting

1.3 DataFrame :

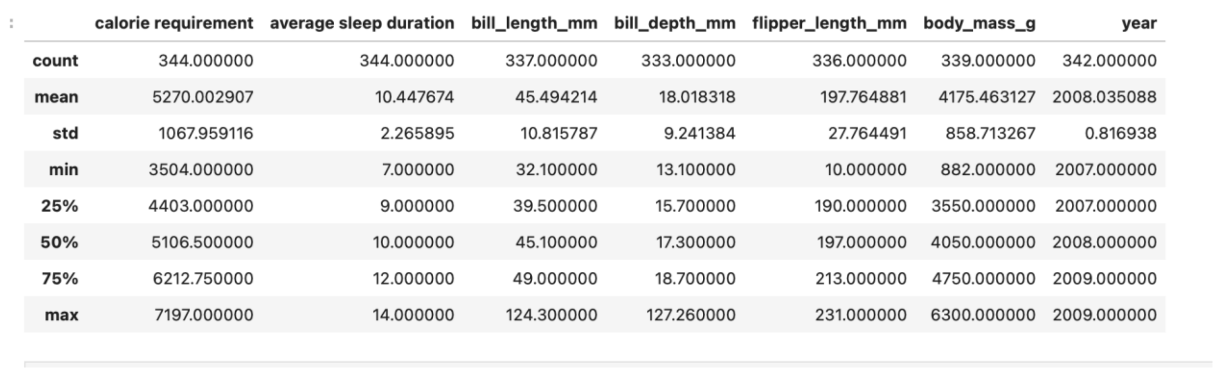
Loading the dataset using pandas is the first step and this process involves importing the dataset into a environment so that it can be manipulated and analyzed.

The dataset consists of following Features :

- species
- island
- calorie requirement
- average sleep duration
- bill_Length_mm
- bill_depth__mm
- flipper_length_mm
- body_mass_g
- gender

1.4 Dataset Statistics :

Dataset statistics provide a summary of the key characteristics of the data, helping to understand its structure using the describe().



	calorie requirement	average sleep duration	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	year
count	344.000000	344.000000	337.000000	333.000000	336.000000	339.000000	342.000000
mean	5270.002907	10.447674	45.494214	18.018318	197.764881	4175.463127	2008.035088
std	1067.959116	2.265895	10.815787	9.241384	27.764491	858.713267	0.816938
min	3504.000000	7.000000	32.100000	13.100000	10.000000	882.000000	2007.000000
25%	4403.000000	9.000000	39.500000	15.700000	190.000000	3550.000000	2007.000000
50%	5106.500000	10.000000	45.100000	17.300000	197.000000	4050.000000	2008.000000
75%	6212.750000	12.000000	49.000000	18.700000	213.000000	4750.000000	2009.000000
max	7197.000000	14.000000	124.300000	127.260000	231.000000	6300.000000	2009.000000

1.5 Handling mismatched string formats :

We have standardized the data so that there are no differences in the capitalization which can lead to inconsistencies and duplicate entries in the dataset. So, we converted all string values to lower case.

1.6 Handling Outliers :

Outliers are data points that deviate significantly from the model's predictions, we have identified the outliers using the Interquartile Range (IQR) method.

1.7 Data Visualization :

We have visualized the data as follows :

Bar Chart : We plotted the penguin dataset by species and created a bar chart [Figure-1] to visualize the counts of each species. And 'adelie' seems to have more count. And also plotted the bar graph based on the flipper range to know the count of penguins. Range of (190 - 200) has more count of penguins.

Heatmap: We have selected the numerical columns from the dataset and computed their correlation matrix, visualizing it with a heatmap using Seaborn. The heatmap displays the strength and direction of relationships between variables, with annotations for clarity.

Kde plot: We have created a kde plot[Figure-2], displaying the distribution of bill lengths for different penguin species in the dataset. The plot is enhanced for easy comparison of bill length distributions among species.

Pie chart: We created a pie chart[Figure-3] to visualize the distribution of these species and providing a clear representation of the proportion of each species within the dataset.

Scatter plot: We created a scatter plot[Figure-4] to visualize the relationship between bill length and bill depth for different penguin species in the dataset. Each point represents a penguin, with colours indicating species.

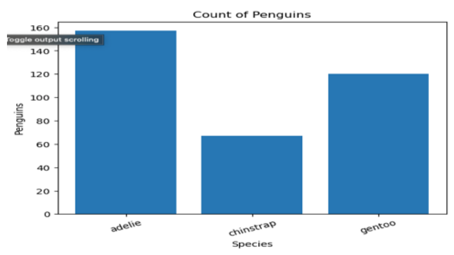


Figure-1 Bar Graph

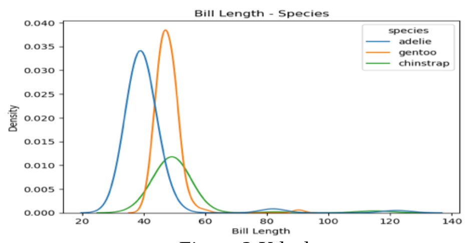


Figure-2 Kdeplot

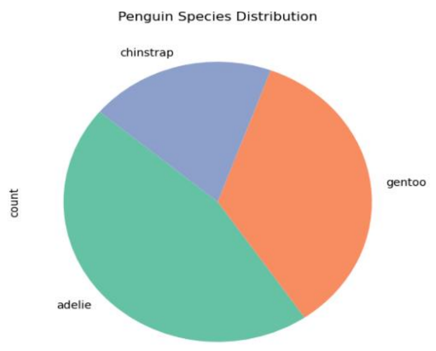


Figure-3 Pie chart

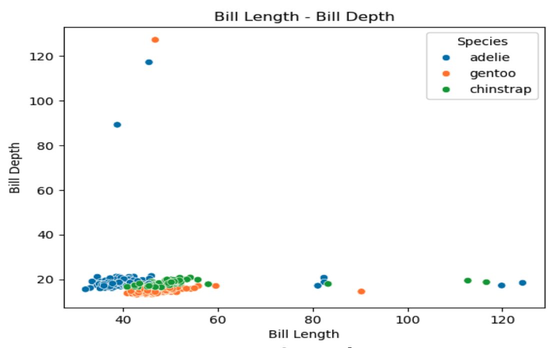


Figure-4 Scatterplot

1.8 Correlation Identification:

We visualized the correlations between various features and the 'gender' variable, finding out significant relationships. By encoding categorical variables, filtering low-correlation features, and presenting the results through a heatmap, and concluded that columns with low correlation are ‘year’ and ‘island’.

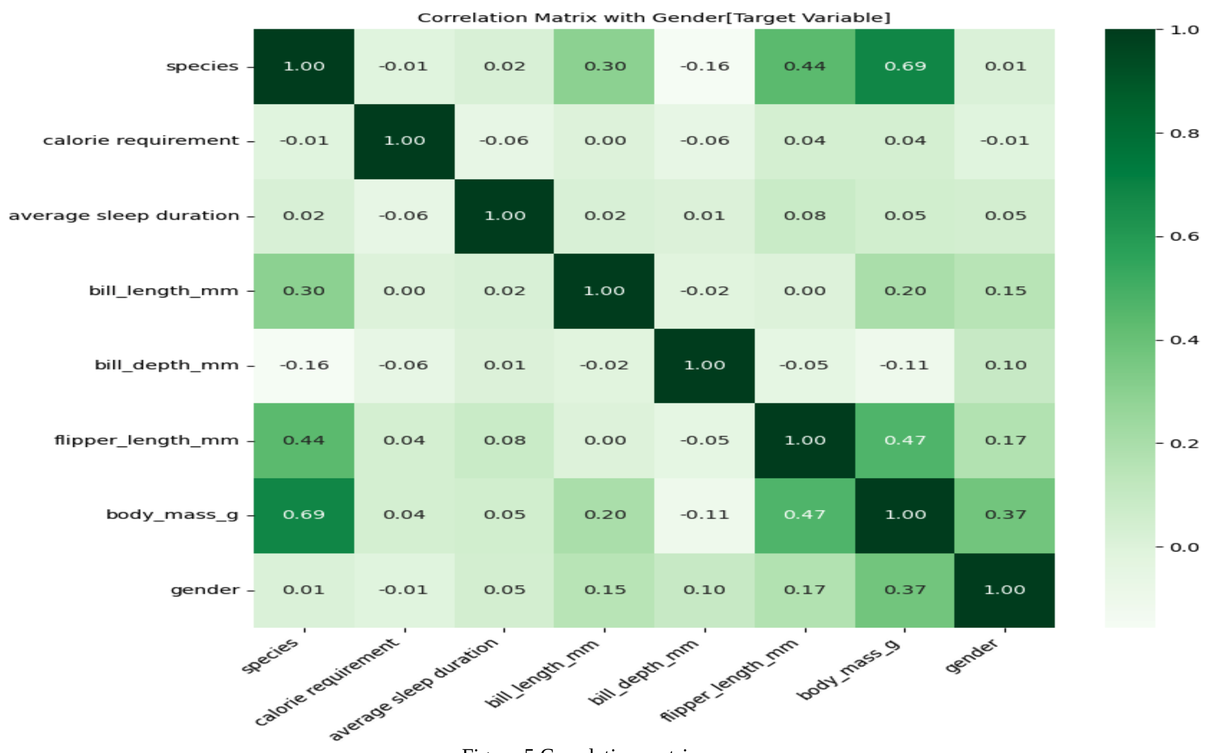


Figure-5 Correlation matrix

1.9 Onehot encoding:

We processed the DataFrame by encoding categorical variables into a numerical format using dummy variables, and converting all data types to integers. These steps prepared the dataset for further analysis and modelling.

1.10 Normalization:

We normalized the numerical columns in the DataFrame using min-max normalization, transforming the data into a consistent scale between 0 and 1.

TASK 2 : DIAMOND DATASET PREPROCESSING

2.1 Introduction :

This report outlines the steps undertaken to preprocess the Diamond dataset, which is crucial for preparing the data for analysis. Effective data preprocessing enhances model performance by addressing missing values, converting categorical variables into numerical representations, and normalizing features, ultimately ensuring the dataset is primed for accurate and efficient machine learning applications.

2.2 Package and Library Usage :

The packages used are :

Pandas (pd): Used to create DataFrame and handle the manipulations.

NumPy **(**np**)**:Used for percentile calculations, and mathematical computations.

Matplotlib and seaborn **:** Used for plotting

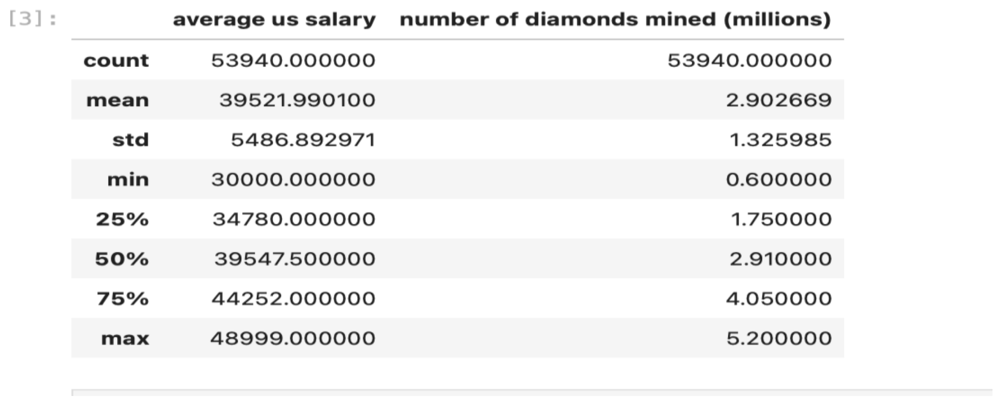
2.3 DataFrame :

Loading the dataset using pandas is the first step and this process involves importing the dataset into a environment so that it can be manipulated and analyzed. The dataset consists of following Features :

- **carat**
- **cut**
- **color**
- **clarity**
- **depth**
- **average us salary**
- **number of diamonds mined (millions)**
- **table**
- **price**
- **x**
- **y**
- **z**

2.4 Dataset Statistics :

Dataset statistics provide a summary of the key characteristics of the data, helping to understand its structure using the describe().



[3]:

	average us salary	number of diamonds mined (millions)
count	53940.000000	53940.000000
mean	39521.990100	2.902669
std	5486.892971	1.325985
min	30000.000000	0.600000
25%	34780.000000	1.750000
50%	39547.500000	2.910000
75%	44252.000000	4.050000
max	48999.000000	5.200000

2.5 Elimination of Invalid entries:

We eliminated invalid entries in a DataFrame and replaced specified string values with missing values (NaN). This preprocessing step is vital for maintaining data quality and improving model performance.

2.6 Handling mismatched string formats :

We have standardized the data so that there are no differences in the capitalization which can lead to inconsistencies and duplicate entries in the dataset. So, we converted all string values to lower case.

2.7 Handling Outliers :

Outliers are data points that deviate significantly from the model’s predictions, we have identified the outliers using the Interquartile Range (IQR) method.

2.8 Data Visualization :

We have visualized the data as follows :

Bar Chart : We created a bar chart to visualize the count of diamonds for each quality of cut in the dataset. The chart displays the cut categories on the x-axis and the corresponding number of diamonds on the y-axis, enhancing understanding of the distribution of diamond cuts.

Heatmap: We have selected the numerical columns from the dataset and computed their correlation matrix, visualizing it with a heatmap using Seaborn. The heatmap displays the strength and direction of relationships between variables, with annotations for clarity.

Box plot: We created a box plot to illustrate the relationship between diamond cut quality and colour in the dataset.

Pie chart: We created a pie chart to visualize the distribution of these cut and providing a clear representation of the proportion of each cut within the dataset.

Kde plot: We visualized the distribution of diamond prices in dataset. It tells us how diamond prices are distributed and highlighting areas of higher frequency within the price range.

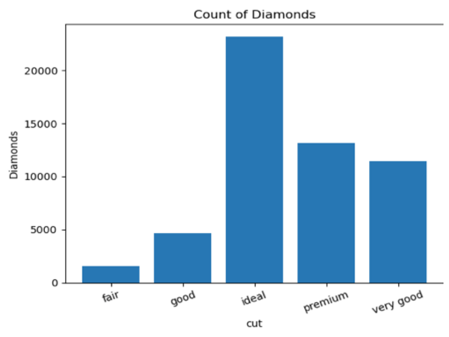


Figure-6 Bar Graph – Diamonds

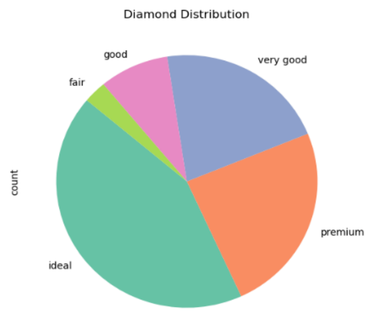


Figure-7 Pie chart - Diamonds

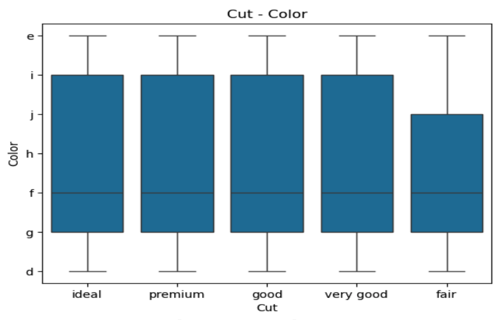
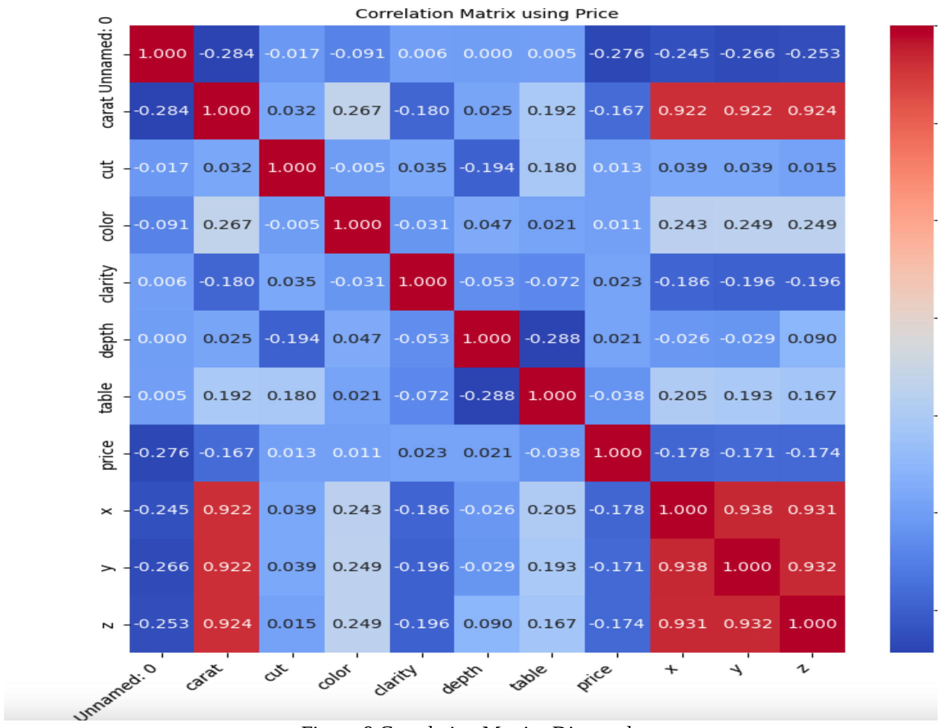


Figure-8 Box plot - Diamonds

2.9 Correlation Identification:

We visualized the correlations between various features and the 'price' variable, finding out significant relationships. By encoding categorical variables, filtering low-correlation features, and presenting the results through a heatmap, and concluded that columns with low correlation are ‘number of diamonds mined (millions)’ and ‘average us salary’.



	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
Unnamed: 0	1.000	-0.284	-0.017	-0.091	0.006	0.000	0.005	-0.276	-0.245	-0.266	-0.253
carat	-0.284	1.000	0.032	0.267	-0.180	0.025	0.192	-0.167	0.922	0.922	0.924
cut	-0.017	0.032	1.000	-0.005	0.035	-0.194	0.180	0.013	0.039	0.039	0.015
color	-0.091	0.267	-0.005	1.000	-0.031	0.047	0.021	0.011	0.243	0.249	0.249
clarity	0.006	-0.180	0.035	-0.031	1.000	-0.053	-0.072	0.023	-0.186	-0.196	-0.196
depth	0.000	0.025	-0.194	0.047	-0.053	1.000	-0.288	0.021	-0.026	-0.029	0.090
table	0.005	0.192	0.180	0.021	-0.072	-0.288	1.000	-0.038	0.205	0.193	0.167
price	-0.276	-0.167	0.013	0.011	0.023	0.021	-0.038	1.000	-0.178	-0.171	-0.174
x	-0.245	0.922	0.039	0.243	-0.186	-0.026	0.205	-0.178	1.000	0.938	0.931
y	-0.266	0.922	0.039	0.249	-0.196	-0.029	0.193	-0.171	0.938	1.000	0.932
z	-0.253	0.924	0.015	0.249	-0.196	0.090	0.167	-0.174	0.931	0.932	1.000

Figure-9 Correlation Matrix- Diamonds

2.10 Onehot encoding:

We processed the DataFrame by encoding categorical variables into a numerical format using categorical, and converting all data types to integers. These steps prepared the dataset for further analysis and modelling.

2.11 Normalization:

We normalized the numerical columns in the DataFrame using min-max normalization, transforming the data into a consistent scale between 0 and 1.

Task 3. Wine Dataset preprocessing

3.1 Introduction :

This report discusses the procedure involved in preprocessing the Wine dataset. The wine dataset consists of two separate dataframes which are White wine and red wine. In this task we concatenate both of these dataframes into one dataset and apply preprocessing steps. Data preprocessing helps in understanding the distinct features of data and improves model performance.

3.2 Package and Library Usage :

The packages used are :

Pandas (pd): Used to create DataFrame and handle the manipulations.

NumPy **(np)**:Used for percentile calculations, and mathematical computations.

Matplotlib and seaborn **:** Used for plotting

3.3 DataFrame :

Loading the dataset using pandas is the first step and this process involves importing the dataset into environment so that it can be manipulated and analysed.

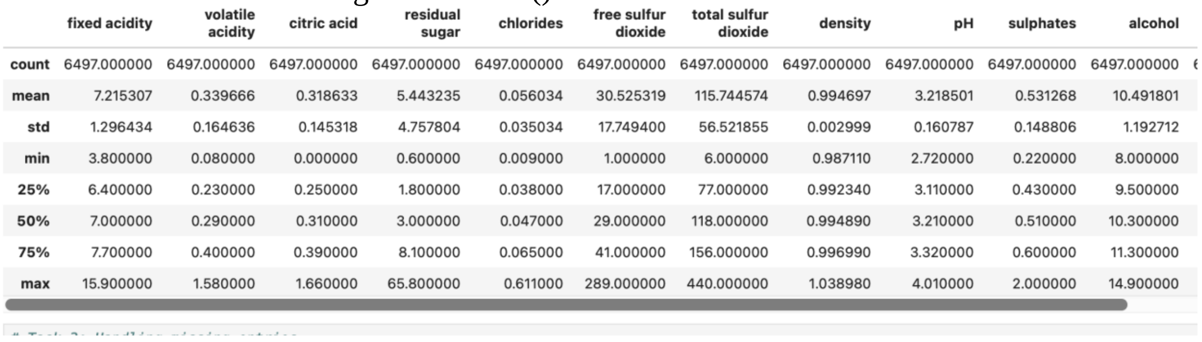
As mentioned earlier, we concatenate both the red wine and white wine data into a single data frame

The merged dataset consists of following Features :

- **fixed acidity**
- **volatile acidity**
- **citric acid**
- **residual sugar**
- **chlorides**
- **free sulfur dioxide**
- **total sulfur dioxide**
- **density**
- **pH**
- **sulphates**
- **alcohol**
- **quality**

3.4 Dataset Statistics :

Dataset statistics provide a summary of the key characteristics of the data, helping to understand its structure using the describe().



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268	10.491801
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000

3.5 Elimination of Invalid entries:

We eliminated invalid entries in a DataFrame and replaced specified string values with missing values (NaN). This preprocessing step is vital for maintaining data quality and improving model performance.

3.6 Handling mismatched string formats :

We have standardized the data so that there are no differences in the capitalization which can lead to inconsistencies and duplicate entries in the dataset. So, we converted all string values to lower case.

3.7 Handling Outliers :

Outliers are data points that deviate significantly from the model's predictions, we have identified the outliers using the Interquartile Range (IQR) method.

3.8 Data Visualization :

We have visualized the data as follows :

Bar Chart : We created a bar chart to visualize the count of diamonds for each quality of cut in the dataset. The chart displays the wine quality categories on the x-axis and the corresponding count frequency of wine on the y-axis, enhancing understanding of the wine quality.

Heatmap: We have selected the numerical columns from the dataset and computed their correlation matrix, visualizing it with a heatmap using Seaborn. The heatmap displays the strength and direction of relationships between variables, with annotations for clarity.

Box plot: We created a box plot to illustrate the quantity of alcohol in wine from the the dataset.

Kde plot: We visualized the distribution the alcohol content in wine and its corresponding density from the given dataset, which helps us in understanding the distribution of alcohol density

Histogram: We plotted a histogram to visualize the distribution of alcohol and their respective count from the combined dataset

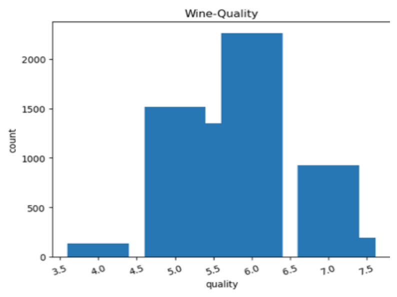


Figure-10 Bar Graph – Wine

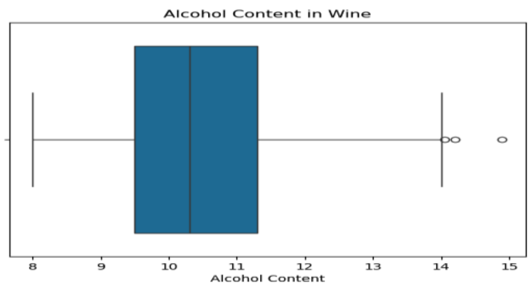


Figure-11 Pie chart - Wine

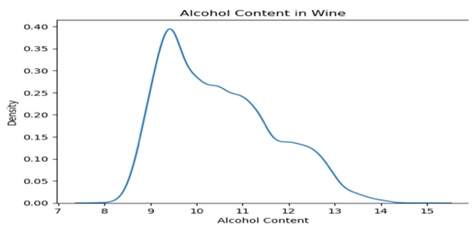


Figure-12 Kdeplot– Wine

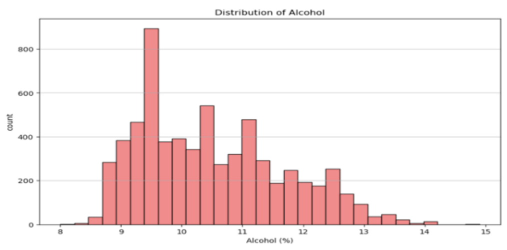


Figure-7 Histogram - Wine

3.9 Correlation Identification:

We visualized the correlations between various features and the ‘quality’ variable, finding out significant relationships. By encoding categorical variables, filtering low-correlation features, and presenting the results through a heatmap, and concluded that columns with low correlation are 'sulphates', 'free sulfur dioxide' and 'pH' and eliminated them to enhance model performance.

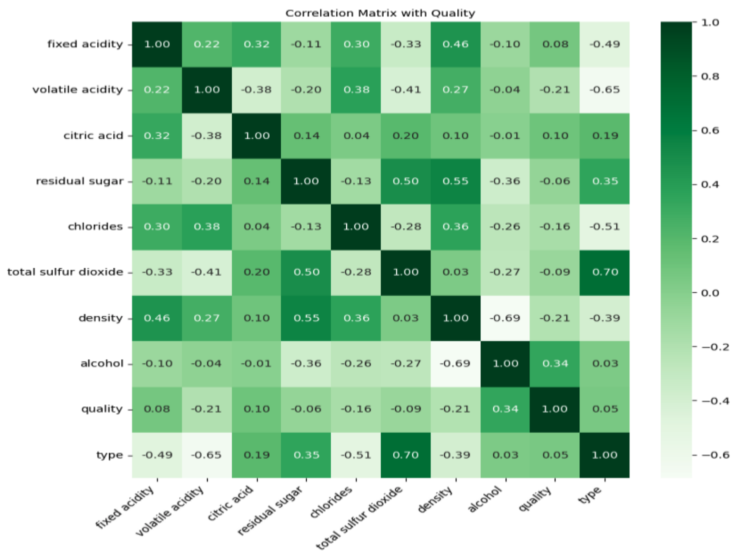


Figure-13 Correlation Matrix– Wine

3.10 Onehot encoding:

This step is skipped as we have no categorical data stored in the wine dataset.

3.11 Normalization:

We normalized the numerical columns in the DataFrame using min-max normalization, transforming the data into a consistent scale between 0 and 1.

Part-2

Introduction:

This task focuses on building a linear regression model using gradient descent on the pre-processed penguins dataset and generate probabilities that maps a real number to a value between 0 and 1.

1.1 Package and Library Usage :

The packages used are :

Pandas (pd): Used to create DataFrame and handle the manipulations.

NumPy **(**np**):**Used for percentile calculations, and mathematical computations.

Matplotlib and seaborn **:** Used for plotting

1.2 Target value:

Once we load the libraries and import the dataset, we'll have to choose the target variable as we initialize model building. We have chosen gender as our target variable. The process involves:

Dropping the target variable['Gender'] from the dataframe and assign the newly formed dataset with a variable 'X'. The dropped column is stored in a variable 'y' which is our target variable.

1.3 Divide the dataset

The dataset is further divided into train and test data with a ratio of 80 percent train set and the remaining is assigned to test the model trained. We stored the data in their respective variable names of X_train, X_test, y_train, y_test. This helps us training and testing the efficiency of our model.

1.4 Shape of data

We print the shape of each variable in test and train set. This helps in understanding the distribution of data in each set.

1.5 Logistic Regression

This step is crucial as define our logistic regression model along with its corresponding parameters and apply the gradient descent function. Below are the steps performed:

We initialized the model with parameters such as learning rate (the rate at which model changes weights), iterations (number of iterations), weights (feature weight values), bias term and loss values after each iteration.

1.6 Sigmoid function

The function is defined where it maps the output to a value between 0 and 1

1.7 Cost function

It performs binary cross-entropy loss which evaluates the model's prediction to the actual values. It is significant because the decrease in its value implies model fitting well in the data.

1.8 Gradient descent

We initialize X as a input feature matrix, y contains actual labels, d as number of features and N as number of samples

This function runs by computing the predicted values with the weights and bias.

The next step involves estimating the loss with current weights

As we update the weights and bias, we simultaneously calculate the cost and update to the loss as model continues to learn.

1.9 Predictions

We fit the data by calling gradient descent function and predict the output by applying a sigmoid function. The output is categorized as follows:

If probability >=0.5 the predicted class is categorized as 1, 0 otherwise.

1.10 Hyper parameter tuning

We test various hyperparameters and choose the best set of combination based on accuracy achieved.

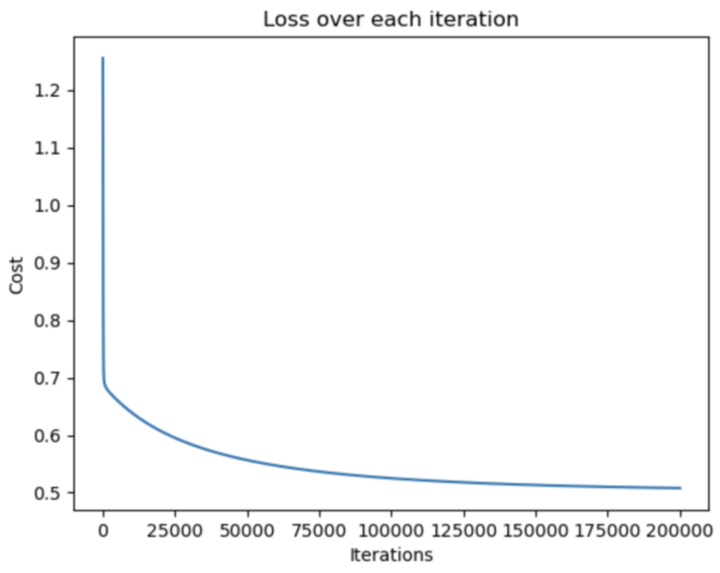
Learning rates we trained with: 1e-3, 5e-3 and 1e-2
Iterations: 50000, 100000, 200000

1.12 Accuracy and saving the best model

As we iterate and model progress, we update the best model combinations and save the final model with best accuracy in our pickle file. We achieved an accuracy of 86.96 percent as saved this model.

1.13 Loss graph:

We visualized a graph that helps us understand loss values over each iteration. The X-axis contains the number of iterations while y-axis denotes the cost function here.



PART – 3

TASK -1: WINE DATASET PREPROCESSING

3.1 Introduction :

This report discusses the procedure involved in preprocessing the Wine dataset. The wine dataset consists of two separate dataframes which are White wine and red wine. In this task we concatenate both of these dataframes into one dataset and apply preprocessing steps. Data preprocessing helps in understanding the distinct features of data and improves model performance.

3.2 Package and Library Usage :

The packages used are :

Pandas (pd): Used to create DataFrame and handle the manipulations.

NumPy **(**np**)**:Used for percentile calculations, and mathematical computations.

Matplotlib and seaborn **:** Used for plotting

3.3 DataFrame :

Loading the dataset using pandas is the first step and this process involves importing the dataset into environment so that it can be manipulated and analysed.

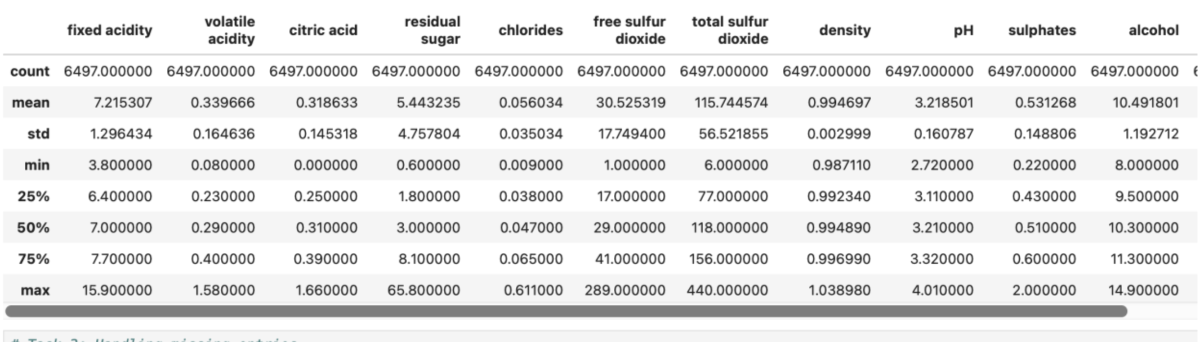
As mentioned earlier, we concatenate both the red wine and white wine data into a single data frame

The merged dataset consists of following Features :

- **fixed acidity**
- **volatile acidity**
- **citric acid**
- **residual sugar**
- **chlorides**
- **free sulfur dioxide**
- **total sulfur dioxide**
- **density**
- **pH**
- **sulphates**
- **alcohol**
- **quality**
-

3.4 Dataset Statistics :

Dataset statistics provide a summary of the key characteristics of the data, helping to understand its structure using the describe().



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268	10.491801
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000

3.5 Elimination of Invalid entries:

We eliminated invalid entries in a DataFrame and replaced specified string values with missing values (NaN). This preprocessing step is vital for maintaining data quality and improving model performance.

3.6 Handling mismatched string formats :

We have standardized the data so that there are no differences in the capitalization which can lead to inconsistencies and duplicate entries in the dataset. So, we converted all string values to lower case.

3.7 Handling Outliers :

Outliers are data points that deviate significantly from the model's predictions, we have identified the outliers using the Interquartile Range (IQR) method.

3.8 Data Visualization :

We have visualized the data as follows :

Bar Chart : We created a bar chart to visualize the count of diamonds for each quality of cut in the dataset. The chart displays the wine quality categories on the x-axis and the corresponding count frequency of wine on the y-axis, enhancing understanding of the wine quality.

Heatmap: We have selected the numerical columns from the dataset and computed their correlation matrix, visualizing it with a heatmap using Seaborn. The heatmap displays the strength and direction of relationships between variables, with annotations for clarity.

Box plot: We created a box plot to illustrate the quantity of alcohol in wine from the the dataset.

Kde plot: We visualized the distribution the alcohol content in wine and its corresponding density from the given dataset, which helps us in understanding the distribution of alcohol density

Histogram: We plotted a histogram to visualize the distribution of alcohol and their respective count from the combined dataset

3.9 Correlation Identification:

We visualized the correlations between various features and the 'quality' variable, finding out significant relationships. By encoding categorical variables, filtering low-correlation features, and presenting the results through a heatmap, and concluded that columns with low correlation are 'sulphates', 'free sulfur dioxide' and 'pH' and eliminated them to enhance model performance.

3.10 Onehot encoding:

This step is skipped as we have no categorical data stored in the wine dataset.

TASK-2 - LINEAR REGRESSION USING OLS FOR WINE DATASET

3.11 Splitting the data:

The dataset df_diamond1 is split into two parts:

- **Training set**: 80% of the data will be used to train the model.
- **Test set**: 20% of the data will be used to evaluate the model's performance on unseen data.
- And defined the target variable as 'quality' and also printed the shapes of X_train, X_test, y_train, y_test.

3.12 Normalization:

When you normalize your data after splitting it into training and test sets, it ensures that the normalization is based only on the training data statistics i.e., mean and standard deviation.

So, If we normalize after splitting, we calculate the mean and standard deviation of the training set and use those values to normalize both the training and test sets. This is important to avoid data leakage.

3.13 Calculation of weights :

We implemented a linear regression model using the OLS, which calculates the weights that minimize the mean squared error between predicted and actual values.

We added a bias term to the training and test data which is done by attaching a column of ones to the feature matrices X_train and X_test using np.c_.

The OLS equation for the weight calculation is given as :

$$\mathbf{w = (X^T \cdot X)^{-1} \cdot X^T \cdot y}$$

- X^T is the transpose of the feature matrix.
- $(X^T \cdot X)^{-1}$ calculates the inverse of the matrix product, which helps find the optimal weights that minimize the loss function.
- The final multiplication with y yields the weight vector w that best fits the training data.

3.14 Mean Squared Error :

The Mean Squared Error (MSE) is calculated for both the training and test sets:

$$\mathbf{MSE = 1/n \sum(y - y')^2}$$

Here, y represents the actual target values, and y’ represents the predicted values. The MSE quantifies how well the model performs, with lower values indicating better accuracy.

The calculated MSE values for both the training and test sets are printed. This provides information about the model's performance on the training data.

3.15 Plot Predictions vs. Actual Values:

Finally, we visualized a scatter plot to depict the relationship between the actual target values and the predicted values for the test set.

This visual representation helps assess the model's performance and reveals any patterns in the predictions.

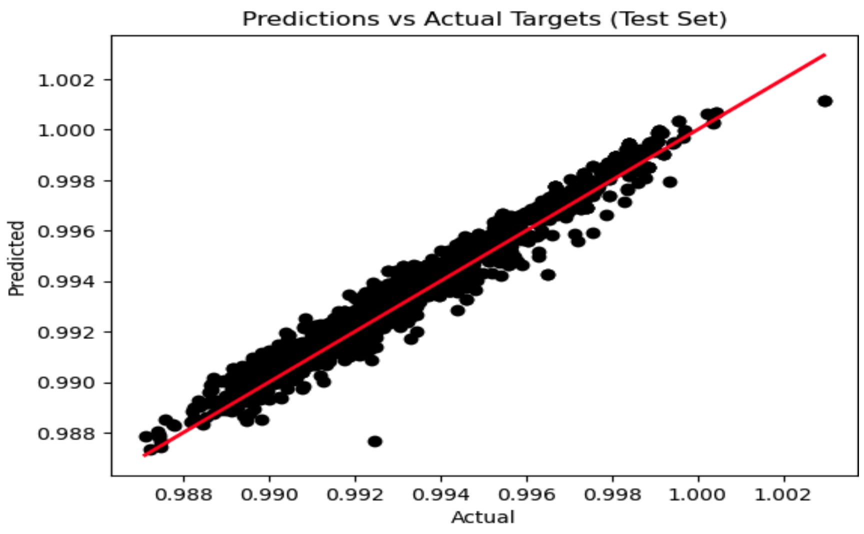


Figure-1 Linear regression with OLS

TASK-3 - RIDGE REGRESSION USING OLS FOR WINE DATASET

3.16 Minimizing the regularized squared loss:

We minimized the regularized squared loss using the Ridge regression implementation.

$$J(w) = 1/2 \sum_{i=1}^{n} (y_i - y_i')^2 + 1/2\ \lambda.y'$$

J(w): This is the total loss function we aim to minimize.

n: The number of observations (samples) in the dataset.

y_i: The actual target value for the i-th observation.

y_I': The predicted value for the i-th observation, calculated as: $X_i.w$

3.17 Calculation of weights :

We computed the weights for the Ridge Regression model using matrix operations. The computation involves taking the inverse of the sum of the product of X^T and X along with the regularization term, and then multiplying by X^T and the target variable y.

The Ridge Regression weights are computed using the formula:

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

X^T is the transpose of the feature matrix.

λI is the regularization term, where I is the identity matrix. This term helps to penalize large weights.

The first entry of the identity matrix is set to zero, ensuring that the bias term is not regularized.

3.18 Mean Squared Error :

The Mean Squared Error (MSE) is calculated for both the training and test sets:

$$\mathbf{MSE = 1/n \sum(y - y')^2}$$

Here, y represents the actual target values, and y' represents the predicted values. The MSE quantifies how well the model performs, with lower values indicating better accuracy.

The calculated MSE values for both the training and test sets are printed. This provides information about the model's performance on the training data.

3.19 Plot Predictions vs. Actual Values:

Finally, we visualized a scatter plot to depict the relationship between the actual target values and the predicted values for the test set.

This visual representation helps assess the model's performance and reveals any patterns in the predictions.

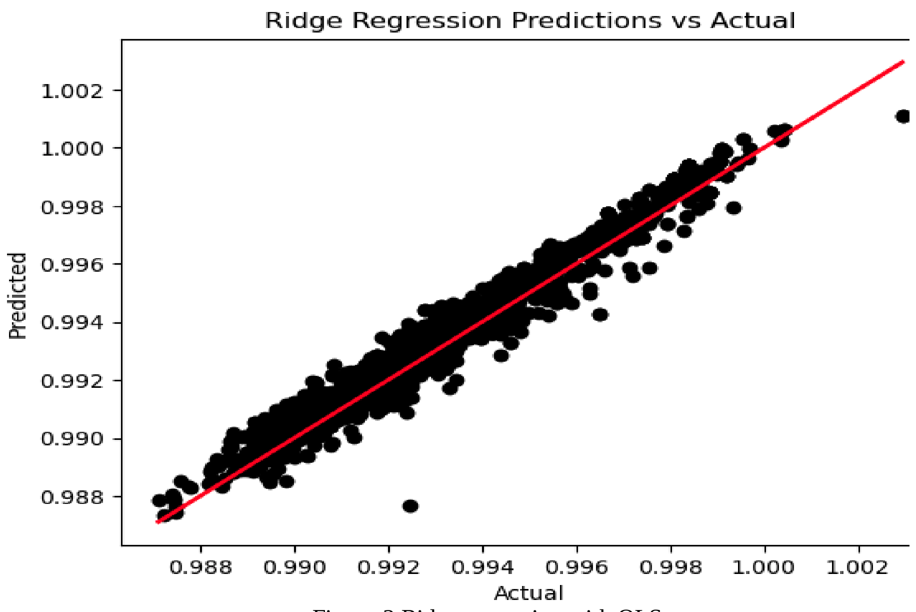


Figure-2 Ridge regression with OLS

3.1. Discuss the benefits and drawbacks of using OLS for weight computation.

Benefits of OLS:

OLS can be used to compute weights by minimizing the sum of squared residuals.

OLS is efficient when applied on small datasets, because of lesser datapoints. we can easily interpret the data.
OLS is unbiased and easy to implement in finding out the best-fit line

Drawbacks of OLS:

OLS is very sensitive to outliers. It is because we minimize the squared errors and in case of large values of outliers, it might not have the right impact on the model.

OLS doesn’t include feature selection mechanism because it includes all the features irrespective of their relevance.

OLS is prone to overfitting, because it learns the noise in the data which makes the predictions on the training set accurate, while performing poorly on unseen data.

Evaluate the strengths and weaknesses of linear regression in general.

Strengths:

Linear regression establishes relationship between the features and the target variable.

It works well for small datasets because it computes the values of coefficients in a single step allowing quick training and evaluation.

When the data is linear, this method is powerful and provides the best estimates of coefficients.

Weakness:

Linear regression depends on assumptions. If any of those assumptions are violated, we get biased results which might be inaccurate.

With high dimensionality, we might require regularization to prevent overfitting

The model cannot find the underlying patterns for large data, and we require complex models to capture it.

Explain the motivation for using L2 regularization and how ridge regression Improves upon linear regression. Discuss its benefits and limitations compared to linear regression.

Using L2 regularization over linear regression helps us in preventing the overfitting. Because we add a penalty term to the loss function, we reduce the model from including large variables.

L2 regularisation priorititze3s features that are important and generalizes new data.

It reduces the variance of the model which increases the performance on unseen data.

Ridge regression can handle multicollinearity better than linear regression and makes them less prone to noise in the data.

Limitations:

The coefficients are reduced till zero which might affect the understanding of relationship between each feature.

Features are close to zero, so they are still present in the data, if feature selection is a priority, this might not be the right choice.

Selecting the right value of lambda heavily influences the model computation. Cross-validating and choosing the right value is important.

PART 4

TASK 1 : DIAMOND DATASET PREPROCESSING

4.1 Introduction :

This report outlines the steps undertaken to preprocess the Diamond dataset, which is crucial for preparing the data for analysis. Effective data preprocessing enhances model performance by addressing missing values, converting categorical variables into numerical representations, and normalizing features, ultimately ensuring the dataset is primed for accurate and efficient machine learning applications.

4.2 Package and Library Usage :

The packages used are :

Pandas (pd): Used to create Data Frame and handle the manipulations.

NumPy (`np`**)**:Used for percentile calculations, and mathematical computations.

Matplotlib and seaborn : Used for plotting

4.3 Data Frame :

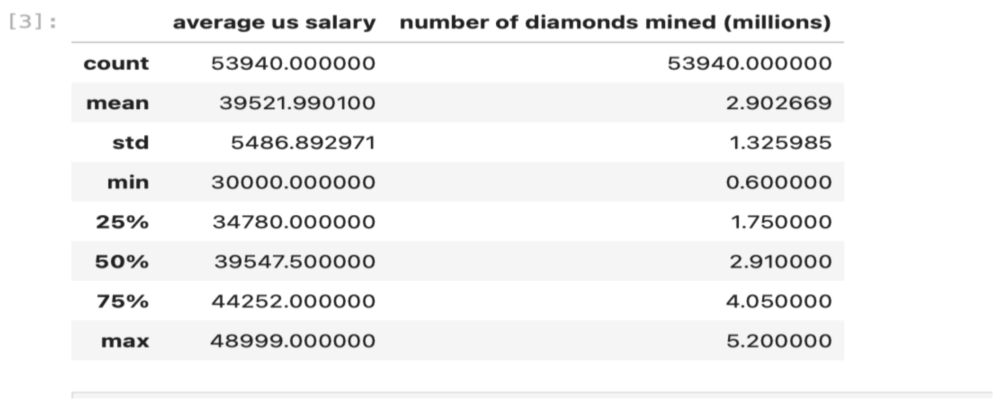
Loading the dataset using pandas is the first step and this process involves importing the dataset into a environment so that it can be manipulated and analyzed.

The dataset consists of following Features :

- **carat**
- **cut**
- **color**
- **clarity**
- **depth**
- **average us salary**
- **number of diamonds mined (millions)**
- **table**
- **price**
- **x**
- **y**
- **z**

4.4 Dataset Statistics :

Dataset statistics provide a summary of the key characteristics of the data, helping to understand its structure using the describe().



```
[3]:          average us salary   number of diamonds mined (millions)
       count       53940.000000                          53940.000000
        mean       39521.990100                              2.902669
         std        5486.892971                              1.325985
         min       30000.000000                              0.600000
         25%       34780.000000                              1.750000
         50%       39547.500000                              2.910000
         75%       44252.000000                              4.050000
         max       48999.000000                              5.200000
```

4.5 Elimination of Invalid entries:

We eliminated invalid entries in a DataFrame and replaced specified string values with missing values (NaN). This preprocessing step is vital for maintaining data quality and improving model performance.

4.6 Handling mismatched string formats :

We have standardized the data so that there are no differences in the capitalization which can lead to inconsistencies and duplicate entries in the dataset. So, we converted all string values to lower case.

4.7 Handling Outliers :

Outliers are data points that deviate significantly from the model's predictions, we have identified the outliers using the Interquartile Range (IQR) method.

4.8 Data Visualization :

We have visualized the data as follows :

Bar Chart : We created a bar chart to visualize the count of diamonds for each quality of cut in the dataset. The chart displays the cut categories on the x-axis and the corresponding number of diamonds on the y-axis, enhancing understanding of the distribution of diamond cuts.

Heatmap: We have selected the numerical columns from the dataset and computed their correlation matrix, visualizing it with a heatmap using Seaborn. The heatmap displays the strength and direction of relationships between variables, with annotations for clarity.

Box plot: We created a box plot to illustrate the relationship between diamond cut quality and color in the dataset.

Pie chart: We created a pie chart to visualize the distribution of these cut and providing a clear representation of the proportion of each cut within the dataset.

Kde plot: We visualized the distribution of diamond prices in dataset. It tells us how diamond prices are distributed and highlighting areas of higher frequency within the price range.

4.9 Correlation Identification:

We visualized the correlations between various features and the 'price' variable, finding out significant relationships. By encoding categorical variables, filtering low-correlation features, and presenting the results through a heatmap, and concluded that columns with low correlation are ‘nuber of diamonds mined (millions)’ and ‘average us salary’.

4.10 Onehot encoding:

We processed the Data Frame by encoding categorical variables into a numerical format using categorical, and converting all data types to integers. These steps prepared the dataset for further analysis and modelling.

TASK-2:

4.11 Splitting the data:

The dataset df_diamond1 is split into two parts:

- **Training set**: 80% of the data will be used to train the model.
- **Test set**: 20% of the data will be used to evaluate the model’s performance on unseen data.
- And defined the target variable as ‘price’ and also printed the shapes of X_train, X_test, y_train, y_test.

4.12 Normalization:

When you normalize your data after splitting it into training and test sets, it ensures that the normalization is based only on the training data statistics i.e., mean and standard deviation.

So, If we normalize after splitting, we calculate the mean and standard deviation of the training set and use those values to normalize both the training and test sets. This is important to avoid data leakage.

4.13 Implementing Elastic Net Regularization :

We have implemented the Elastic Net regression, which combines Lasso (L1) and Ridge (L2) regularization to balance feature selection and stability. And initially assigned the weights to zeros

The model minimizes a loss function that includes both the squared error and penalties for the size of the weights.

The Loss function is given as :

- w are the weights.
- λ1 is the regularization parameter for L2 (Ridge penalty).
- λ2 is the regularization parameter for L1 (Lasso penalty).
- X is the feature matrix, and y is the target.

4.14 Updating of weights :

We have updated the weights using the given formula:

The update rule for each weight w_j in the following two steps:

L2 (Ridge) update**:** The weight is adjusted using gradient descent to reduce the prediction error. At the same time, a penalty is applied to the weight's size, encouraging smaller values to prevent overfitting.

L1 (Lasso) update: After the weight is updated, a process called soft-thresholding is applied. This reduces the weight towards zero if it's small. If the weight is below a certain threshold, it’s set to zero, helping the model eliminate irrelevant features and focus on the most important ones.

We have used the soft-thresholding step because it is essential for applying the Lasso penalty. If the weight w_j is larger than λ2, it is reduced by the amount λ2. If it's smaller than −λ2, it is increased by λ2. If it's within the range [−λ2,λ2], it is set to zero. This step allows the model to effectively eliminates less important features

4.15 Weight Initializations

After we update the wights, we experiment with different methods of weight initialization.

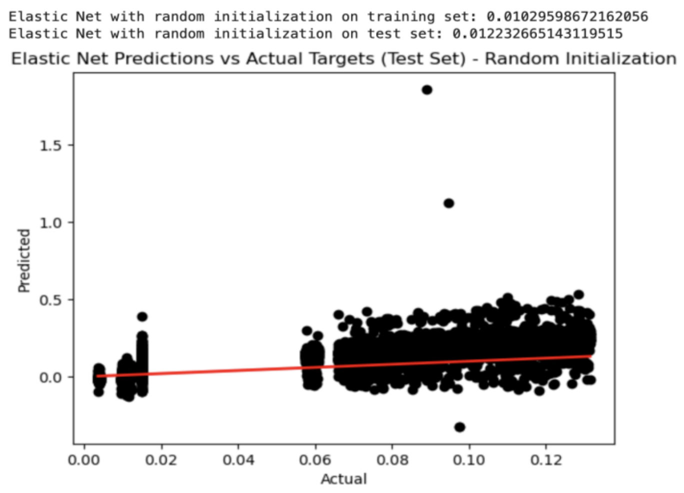
Random: We randomly assign values to weights which is used to infer the variability in resulting over different sets of weights.

Zeroes: We initialized all the weights to zero.

Xavier: This method is used to maintain stability by ensuring gradients doesn’t vanish. This is done by limiting the weight based on the dimensions of data.

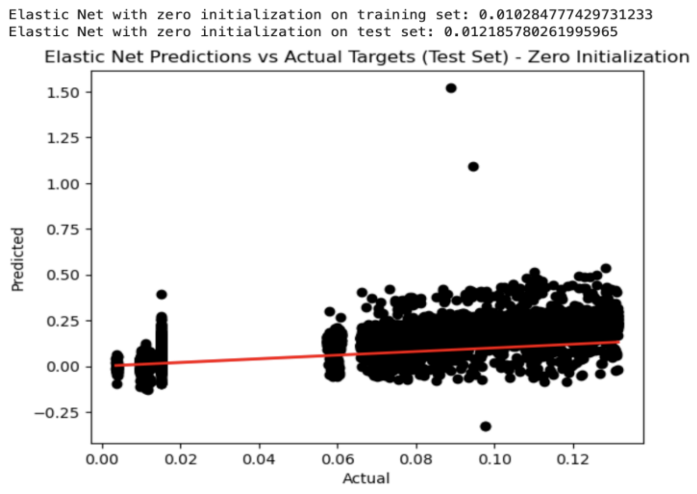
4.16 Model evaluation

Elastic Net with random initialization on training set: 0.01029598672162056
Elastic Net with random initialization on test set: 0.012232665143119515



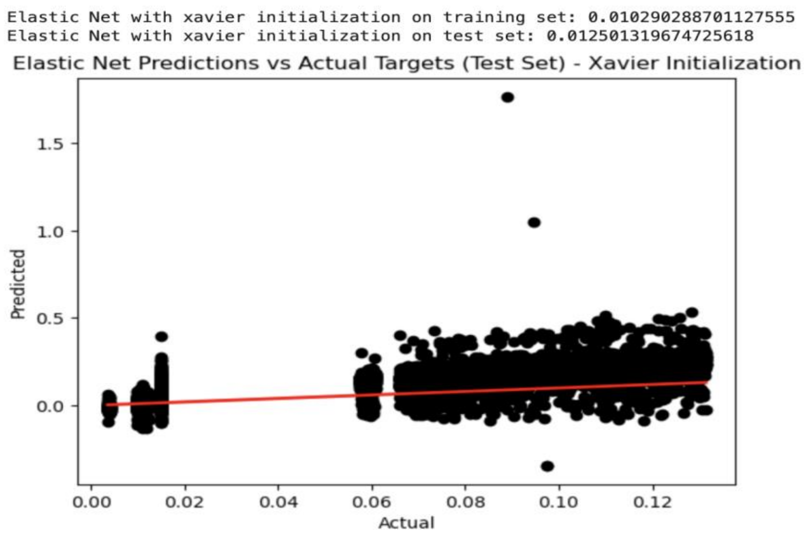
```
Elastic Net with random initialization on training set: 0.010295986721620 56
Elastic Net with random initialization on test set: 0.012232665143119515
```

Elastic Net with zero initialization on training set: 0.010284777429731233
Elastic Net with zero initialization on test set: 0.012185780261995965



```
Elastic Net with zero initialization on training set: 0.010284777429731233
Elastic Net with zero initialization on test set: 0.012185780261995965
```

Elastic Net with xavier initialization on training set: 0.010290288701127555
Elastic Net with xavier initialization on test set: 0.012501319674725618



```
Elastic Net with xavier initialization on training set: 0.010290288701127555
Elastic Net with xavier initialization on test set: 0.012501319674725618
```

Threshold	**Epochs**	**Weights**	**MSE**
0.001	1000	Xavier	0.0302
0.001	1000	Random	0.0451
0.001	1000	Zero	0.0205
0.001	5000	Xavier	0.0158
0.001	5000	Random	0.0283
None	5000	**Zero**	**0.01029**
None	5000	**Xavier**	**0.01028**
None	5000	**Random**	**0.01049**

Effects of different weight initialization methods:

- **Xavier Initialization**: Xavier initialization yielded strong results. It facilitated faster convergence with smoother loss curves over multiple epochs. For instance, after 5000 epochs without a stopping threshold, the model reached a mean squared error (MSE) of 0.01028.

- **Zero Initialization**: Interestingly, zero initialization performed similarly to Xavier. The model, when using zero initialization delivered an MSE of 0.01029. This result suggests that the effectiveness of zero initialization can be enhanced by tuning the epochs.

- **Random Initialization**: Random initialization demonstrated more variability, needing more epochs to match the performance of other methods the MSE was slightly higher at 0.01049, indicating some instability compared to Xavier initialization.

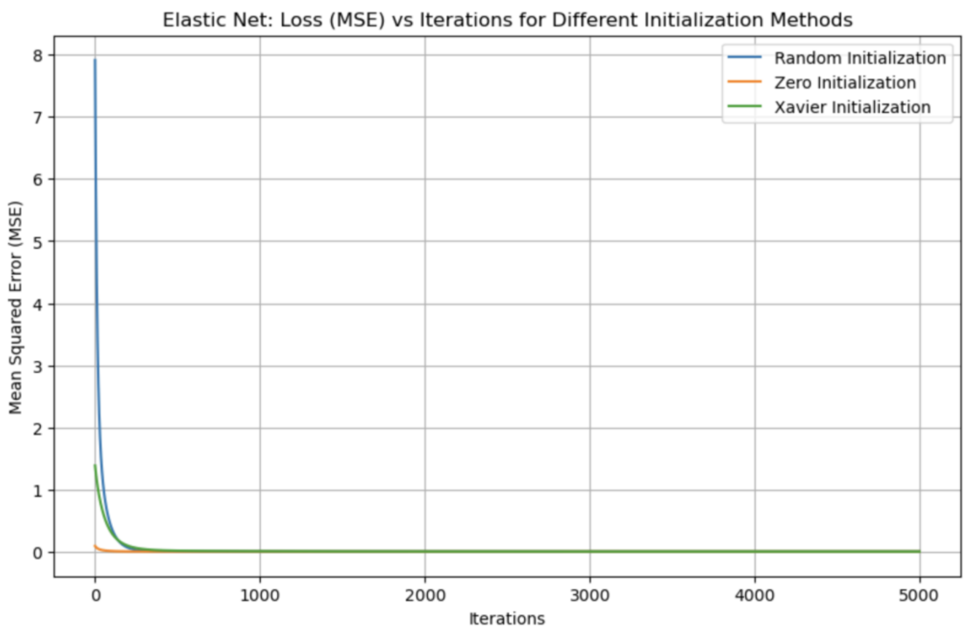
- **Summary**: Both zero and Xavier initialization produced nearly identical results in terms of convergence. Zero initialization was more aggressive, converging faster, while Xavier initialization had a smoother convergence but required more iterations. The performance difference between the two methods was minimal.

Insights of Weight Initialization:

- **Xavier Initialization**: Xavier initialization consistently demonstrated strong convergence and across different configurations. It delivered the best overall performance in terms of MSE, with the most optimal results achieved after 5000 epochs.

- **Random Initialization**: Random initialization exhibited greater variability and required longer training times to reach performance levels comparable to Xavier. Though competitive, its convergence was less stable, leading to fluctuating outcomes.

- **Zero Initialization**: Despite being generally expected to perform poorly, zero initialization surprisingly matched the results of Xavier in this particular experiment. The most favorable performance was observed after 5000 epochs.



4.17 When using Early Stopping:

In early stopping we designed to prevent the model from continuing training once it has effectively learned the underlying patterns in the data.

At each iteration of the training loop, the gradient of the loss function is calculated for each weight (w[j]). The gradient tells the model how much the weights need to be adjusted to minimize the error in predictions.

Convergence Behavior:

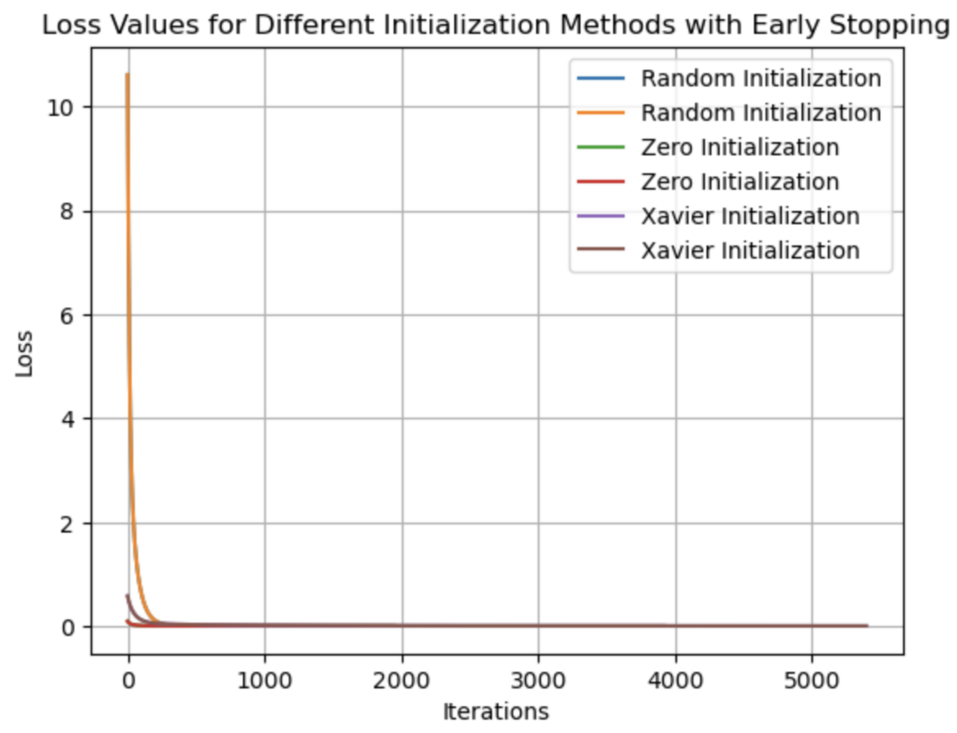
All three initialization methods show a sharp decrease in loss during the first few iterations, indicating that the model quickly learns and reduces error early in the training process.
After around 1000 iterations, the loss stabilizes and becomes almost flat, meaning further training brings very minimal improvement.

Early Stopping:

we can see from the provided data that the training process stopped early when the gradients became small enough, meaning that additional iterations would not significantly reduce the loss.
For Random Initialization, early stopping occurred at iteration 4050, for Zero Initialization it occurred at 1947, and for Xavier Initialization, it stopped at 5047. This indicates that each initialization method reached its point of diminishing returns at different times.

Loss Values at Convergence:

Xavier Initialization performed the best, with the smallest loss value towards the end of the iterations. This shows that the Xavier method helps the model converge more effectively, resulting in better performance on both training and test sets.
Random and Zero Initialization performed similarly, though Zero Initialization reached early stopping earlier, which could indicate a faster convergence, but with slightly higher loss compared to Xavier.



From the Graph:

Xavier Initialization: This method consistently performed the best, reaching the lowest loss value. This suggests that initializing the weights with Xavier allows for faster and more effective learning compared to Random and Zero methods. The test set performance (MSE of 0.0102) also supports this observation.

Random and Zero Initialization: Both methods converged but showed slightly higher final loss values, which indicates that these methods might not be as effective in reducing the error or achieving optimal weight adjustments compared to Xavier.

4.18 Discussion:

L1 Regularization (Lasso): L1 promotes sparsity by driving many coefficients to zero, which helps with feature selection. However, it struggles with correlated features, often keeping one while dropping others.

L2 Regularization (Ridge): L2 penalizes large coefficients, shrinking them but not forcing them to zero. It handles multicollinearity well but doesn't perform feature selection.

Elastic Net: Combines L1's feature selection with L2's ability to manage correlated features. In the wine dataset, Elastic Net is effective in preventing overfitting and handling multicollinearity, balancing sparsity and regularization.

Advantages: Elastic Net strikes a balance, performing feature selection while also addressing correlated features, making it more versatile than using L1 or L2 alone.

Benefits:

Elastic Net: Reduces overfitting while selecting key features, making it ideal for models with many inputs and multicollinearity.

Gradient Descent: Efficiently minimizes the loss function and is well-suited for large datasets, helping models converge.

Drawbacks:

Elastic Net: Requires tuning two hyperparameters (L1 and L2), adding complexity to model optimization.

Gradient Descent: Sensitive to learning rates; improper tuning can result in slow convergence or divergence.

References

- https://www.geeksforgeeks.org/implementation-of-elastic-net-regression-from-scratch/
- https://www.geeksforgeeks.org/xavier-initialization/
- https://alok05.medium.com/ridge-lasso-and-elastic-net-regression-48a6684b7ead