Report for Part 1 Data Analysis & Preprocessing

1. Methods used for data preprocessing -

1.1. The first step in our data preprocessing was to handle null values. So we first found out the number of null values for each column in the dataset. And then imputed the missing values with mean, median and mode depending upon the type of the feature and the feature's data. For eg.- In the penguins dataset we have handled missing values in the below format.

```
df['species']=df['species'].fillna(df['species'].mode()[0])
df['island']=df['island'].fillna(df['island'].mode()[0])
df['gender']=df['gender'].fillna(df['gender'].mode()[0])
df['bill_length_mm']=df['bill_length_mm'].fillna(df['bill_length_mm'].mean())
df['bill_depth_mm']=df['bill_depth_mm'].fillna(df['bill_depth_mm'].mean())
df['flipper_length_mm']=df['flipper_length_mm'].fillna(df['flipper_length_mm'].mean())
df['body_mass_g']=df['body_mass_g'].fillna(df['body_mass_g'].median())
df['year']=df['year'].fillna(df['year'].mode()[0])
df.isna().sum()
```

1.2. We then handled the mismatched string format in the dataset. We achieved this by converting the strings into lower or upper case. For eg.- In the penguins dataset we standardized the string formats by capitalizing the first letter of the string while keeping other letters in lowercase.

```
def standardize(value):
    return value.capitalize()
df['species']=df['species'].apply(standardize)
df['island']=df['island'].apply(standardize)
df['gender']=df['gender'].apply(standardize)
```

✓ 0.0s

1.3. We then handled the outlier using Interquartile range and replaced the values that are below the lower bound and above the upper bound with the median of the feature.

```
columns=['bill_length_mm','bill_depth_mm','flipper_length_mm','body_mass_g']
def impute_outlier(df, column):
    impute_column = df[column]
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1

    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    non_outlier_median = impute_column[(df[column] >= lower_bound) & (df[column] <= upper_bound)].median()

    def impute_value(value):
        if value < lower_bound or value > upper_bound:
            return non_outlier_median
        else:
            return value

    df[column] = df[column].apply(impute_value)

for column in columns:
    impute_outlier(df, column)

df
```

1.4. We then normalized the values in the features that are non-categorical and rescaled the values in the range from 0 to 1. For eg.- in the penguins dataset we normalized bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g, calorie requirement.

```
def normalize(df,column):
    column_min = df[column].min()
    column_max = df[column].max()
    print("Minimum value for ",column," is: ",column_min)
    print("Maximum value for ",column," is: ",column_max)
    df[column] = (df[column] - column_min) / (column_max - column_min)

columns=['bill_length_mm', 'bill_depth_mm','flipper_length_mm', 'body_mass_g','calorie requirement']
for column in columns:
    normalize(df,column)
df
```

2.Overview of the dataset -

Penguin Dataset -

2.a.The penguin dataset primarily focuses on the species type and the biological characteristics such as body measurements. The dataset consists of categorical and numerical values. And the dataset contains 344 samples and 10 features.

2.b. Below are the key statistics such as mean and standard deviation and missing values.

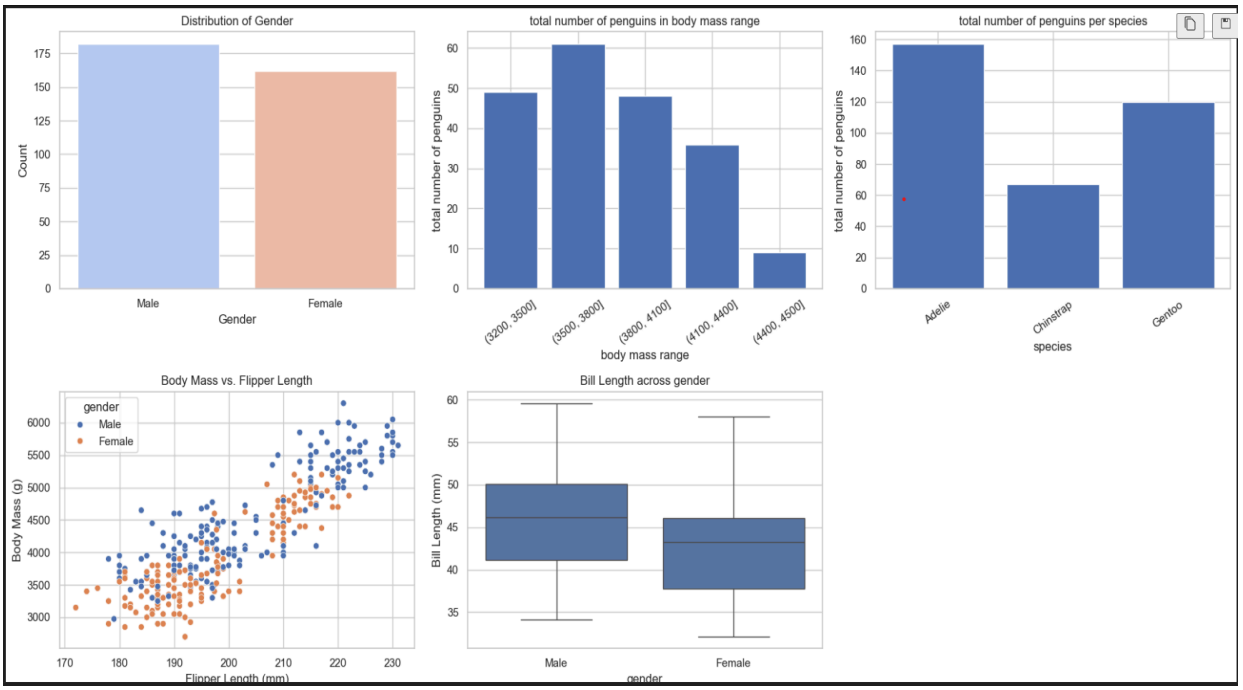
```
df.describe()
```
✓ 0.0s

	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

```
#finds columns with null values
df.isna().sum()
```
✓ 0.0s

```
species                    11
island                     10
calorie requirement         0
average sleep duration      0
bill_length_mm              7
bill_depth_mm              11
flipper_length_mm           8
body_mass_g                 5
gender                     17
year                        2
dtype: int64
```

2.c.



Observation -

1.The first graph shows that the distribution of male and female is fairly similar.

2.Penguins with body mass range between 3500-3800 form the largest group and the count of penguins group size decreases as we move to the extreme.

3.Most penguins fall under the Adelie species while there are very few penguins that fall under Chinstrap species.

4.Male penguins have higher body mass and longer flipper length compared to females.

5.Males have longer bill length than females on average.

Diamond Dataset -

2.a.The diamond dataset primarily focuses on the price of the diamond and its characteristics. The dataset contains both numerical and categorical columns. The dataset consists of 53940 samples and 12 features.

2.b.Below are the key statistics with mean and standard deviation, along with missing values in the dataset -

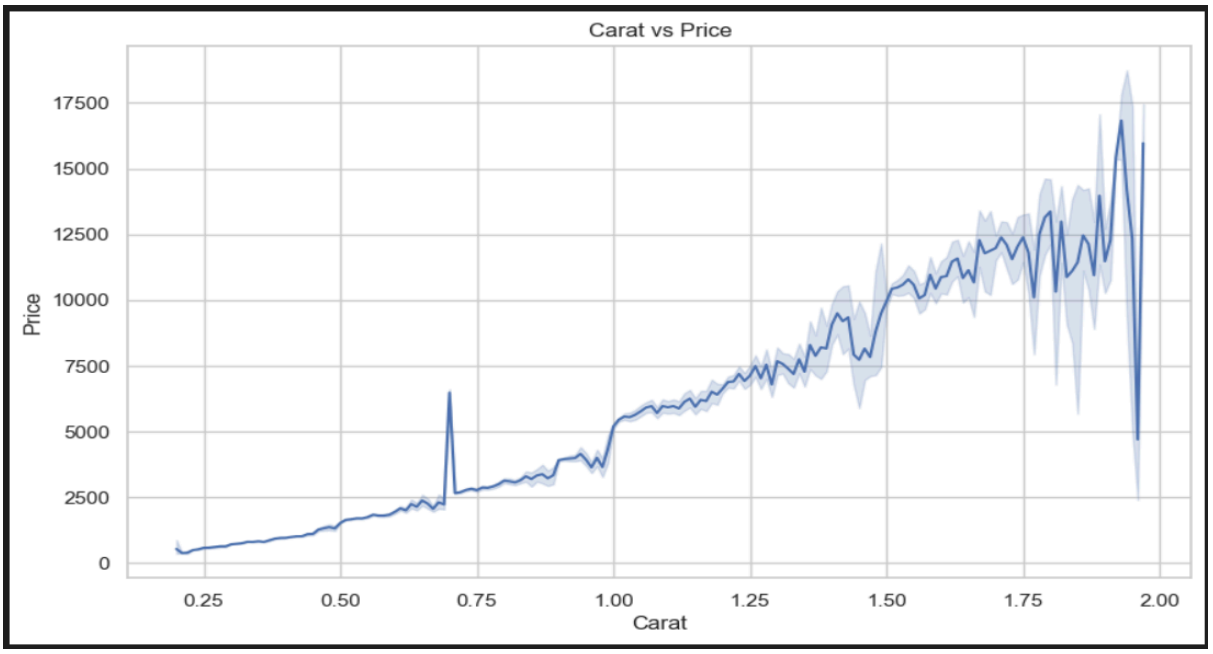
```
diamond_df.describe()
```

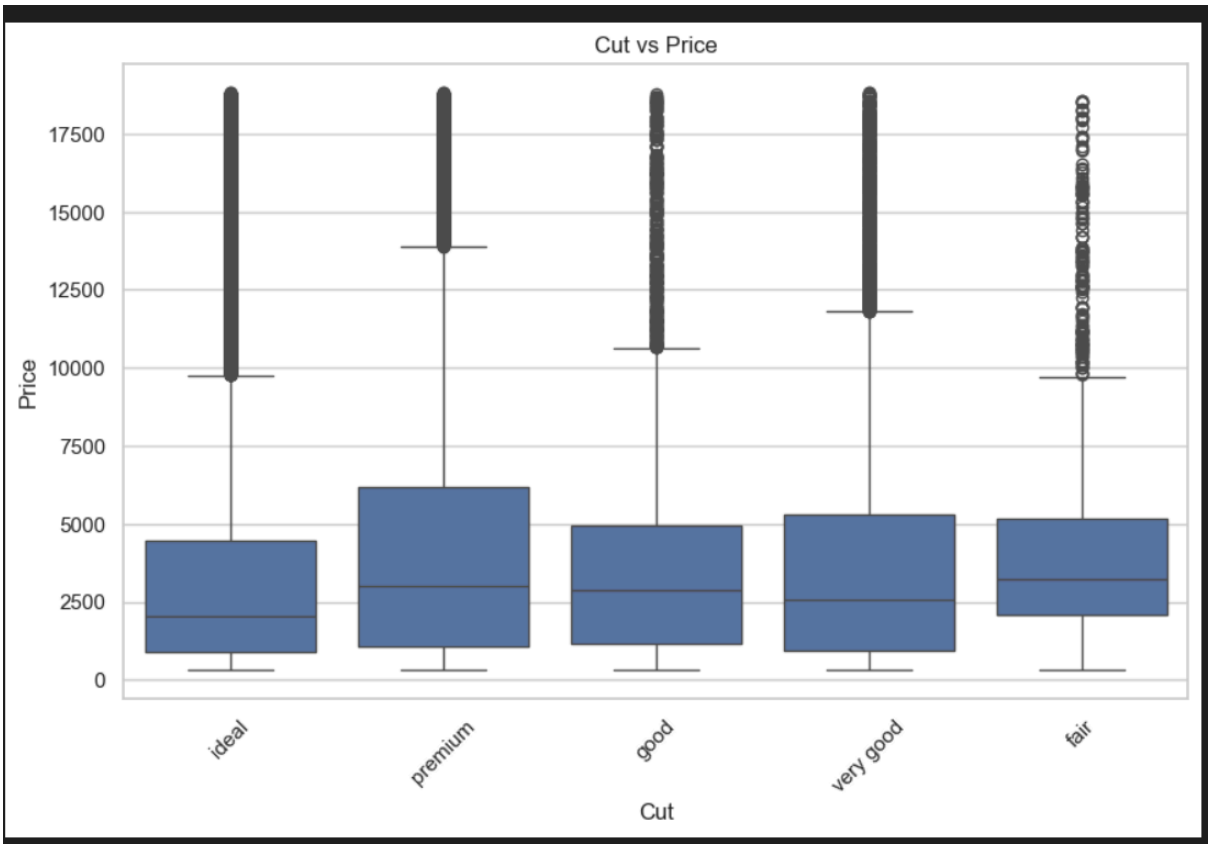
	carat	average us salary	number of diamonds mined (millions)	depth	table	price	x	y	z
count	51073.000000	53940.000000	53940.000000	51866.000000	51030.000000	51844.000000	51027.000000	52208.000000	51532.000000
mean	0.797823	39521.990100	2.902669	61.750175	57.456332	3933.022047	5.731451	5.734517	3.538203
std	0.473747	5486.892971	1.325985	1.433485	2.231611	3989.013631	1.121433	1.142543	0.706057
min	0.200000	30000.000000	0.600000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	34780.000000	1.750000	61.000000	56.000000	951.000000	4.710000	4.720000	2.910000
50%	0.700000	39547.500000	2.910000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	44252.000000	4.050000	62.500000	59.000000	5327.250000	6.540000	6.540000	4.030000
max	5.010000	48999.000000	5.200000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

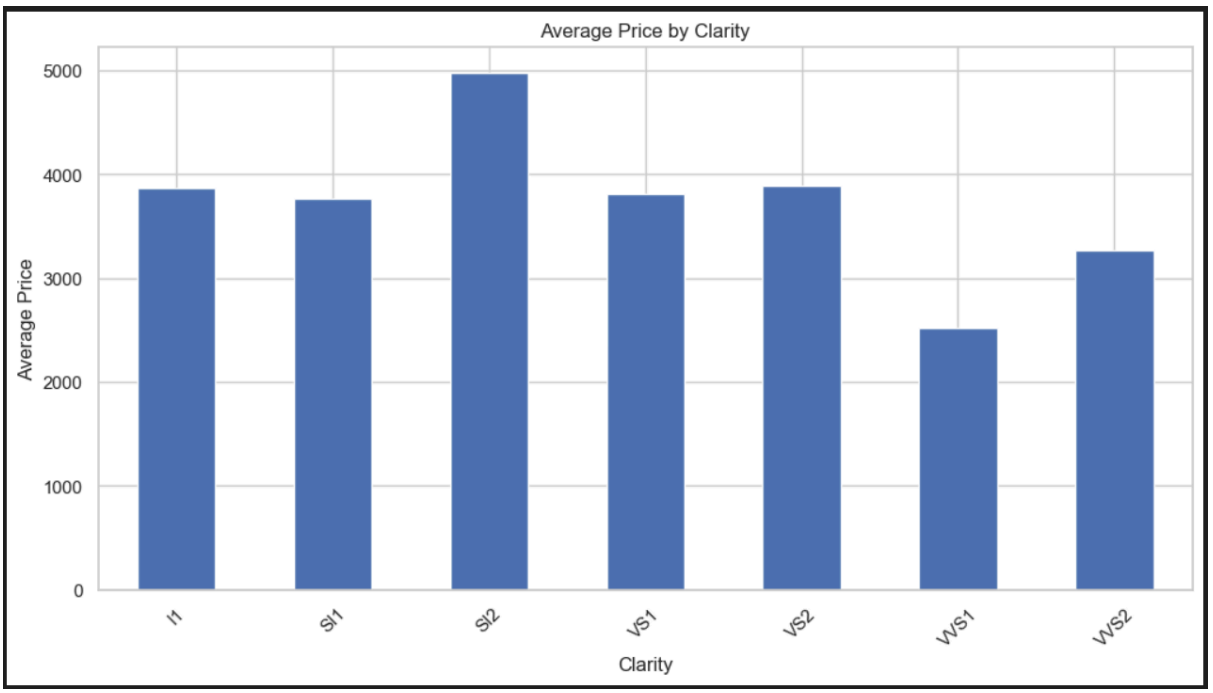
```
diamond_df.isna().sum()
```

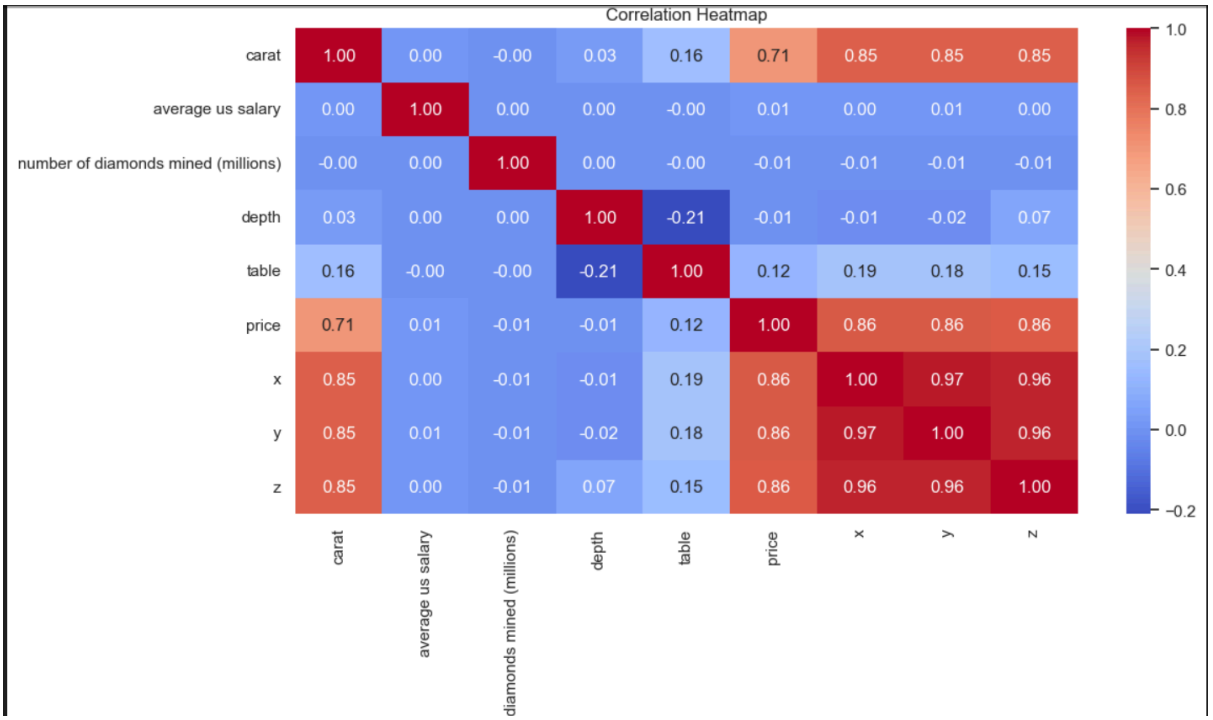
```
carat                                2867
cut                                  2682
color                                2890
clarity                              3472
average us salary                       0
number of diamonds mined (millions)     0
depth                                2074
table                                2910
price                                2096
x                                    2913
y                                    1732
z                                    2408
dtype: int64
```

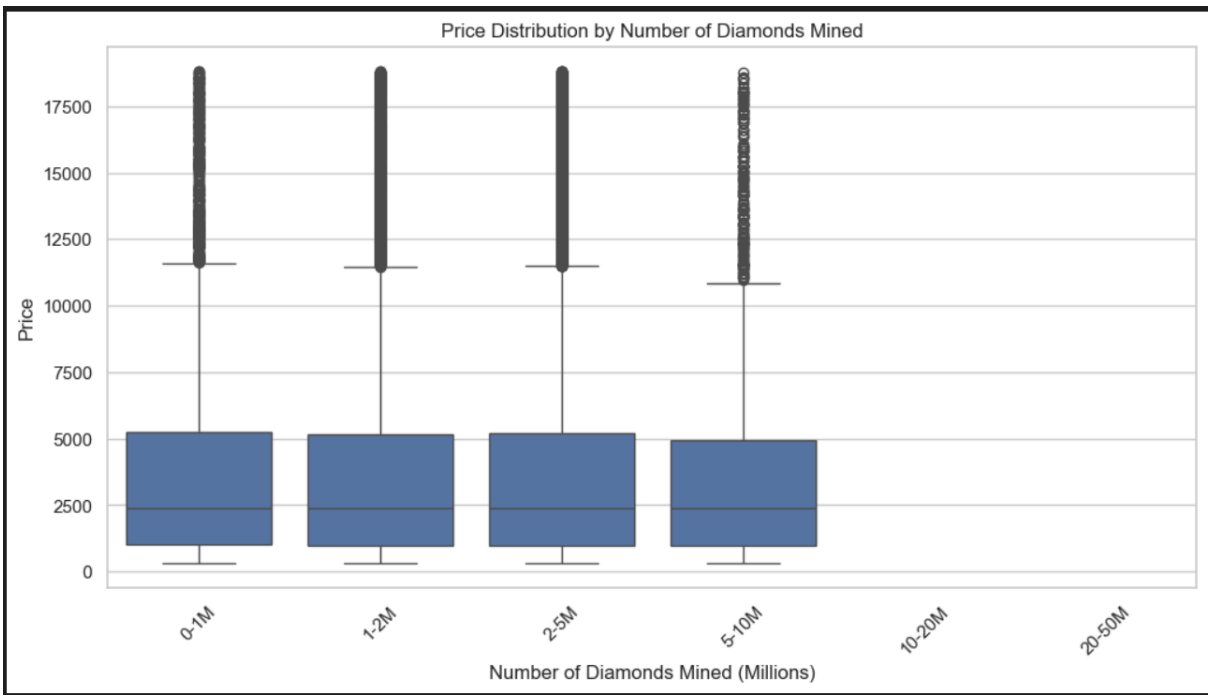
2.c.


Carat vs Price
Price
17500
15000
12500
10000
7500
5000
2500
0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Carat


Cut vs Price
Price
17500
15000
12500
10000
7500
5000
2500
0
ideal
premium
good
very good
fair
Cut


Average Price by Clarity
Average Price
5000
4000
3000
2000
1000
0
I1
SI1
SI2
VS1
VS2
VVS1
VVS2
Clarity


Correlation Heatmap
carat 1.00 0.00 -0.00 0.03 0.16 0.71 0.85 0.85 0.85
average us salary 0.00 1.00 0.00 0.00 -0.00 0.01 0.00 0.01 0.00
number of diamonds mined (millions) -0.00 0.00 1.00 0.00 -0.00 -0.01 -0.01 -0.01 -0.01
depth 0.03 0.00 0.00 1.00 -0.21 -0.01 -0.01 -0.02 0.07
table 0.16 -0.00 -0.00 -0.21 1.00 0.12 0.19 0.18 0.15
price 0.71 0.01 -0.01 -0.01 0.12 1.00 0.86 0.86 0.86
x 0.85 0.00 -0.01 -0.01 0.19 0.86 1.00 0.97 0.96
y 0.85 0.01 -0.01 -0.02 0.18 0.86 0.97 1.00 0.96
z 0.85 0.00 -0.01 0.07 0.15 0.86 0.96 0.96 1.00
carat
average us salary
diamonds mined (millions)
depth
table
price
x
y
z
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



Observation -

1.We can see a positive correlation between carat and price, and prices keep increasing as carat size increases.

2.We can see the price distribution of diamonds across different cut qualities, with premium cuts having the highest price.

3.We can see the average price by clarity, with SI2 clarity having the highest average price.

4.Correlation heatmap shows strong positive correlation between carat,price and x/y/z dimensions of diamonds.

5.We can see the price distribution by the number of diamonds mined, with a fairly consistent distribution across the number of diamonds mined.

Wine dataset -

2.a. The dataset consists of quality of wines, 2 different wine types, characteristics and the chemical properties. The dataset contains both numerical and categorical columns. The dataset consists of 6497 samples and 13 features.

2.b. Below are the key statistics with mean and standard deviation. There are no missing values in the dataset.

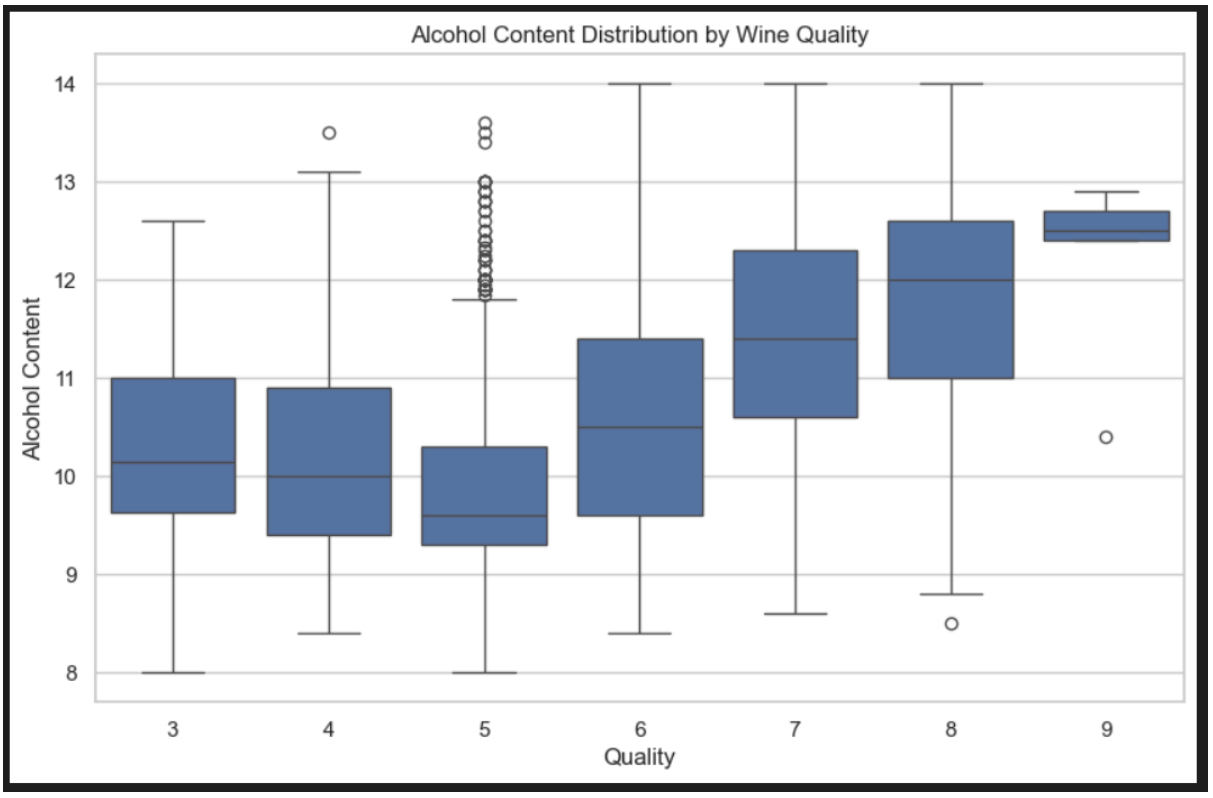
```
wine_df.describe()
```
✓ 0.0s

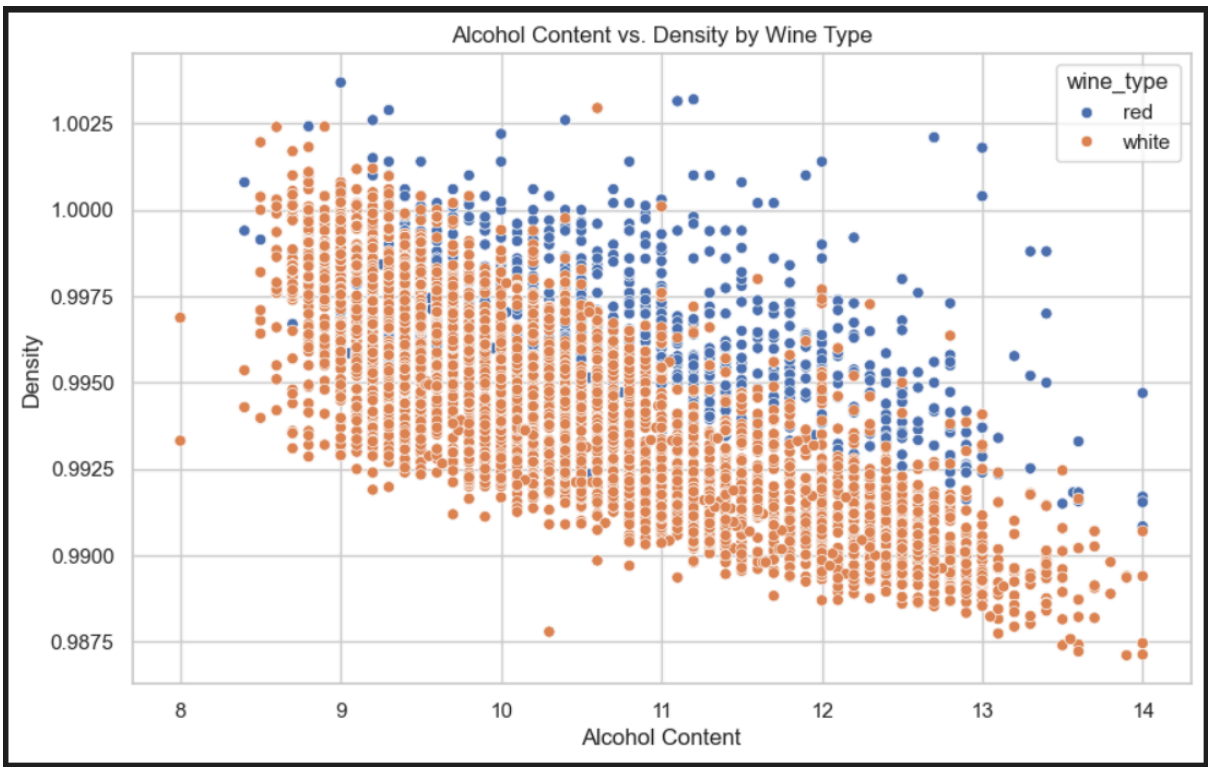
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268	10.491801	5.818378
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712	0.873255
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000	5.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000	6.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000	6.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000

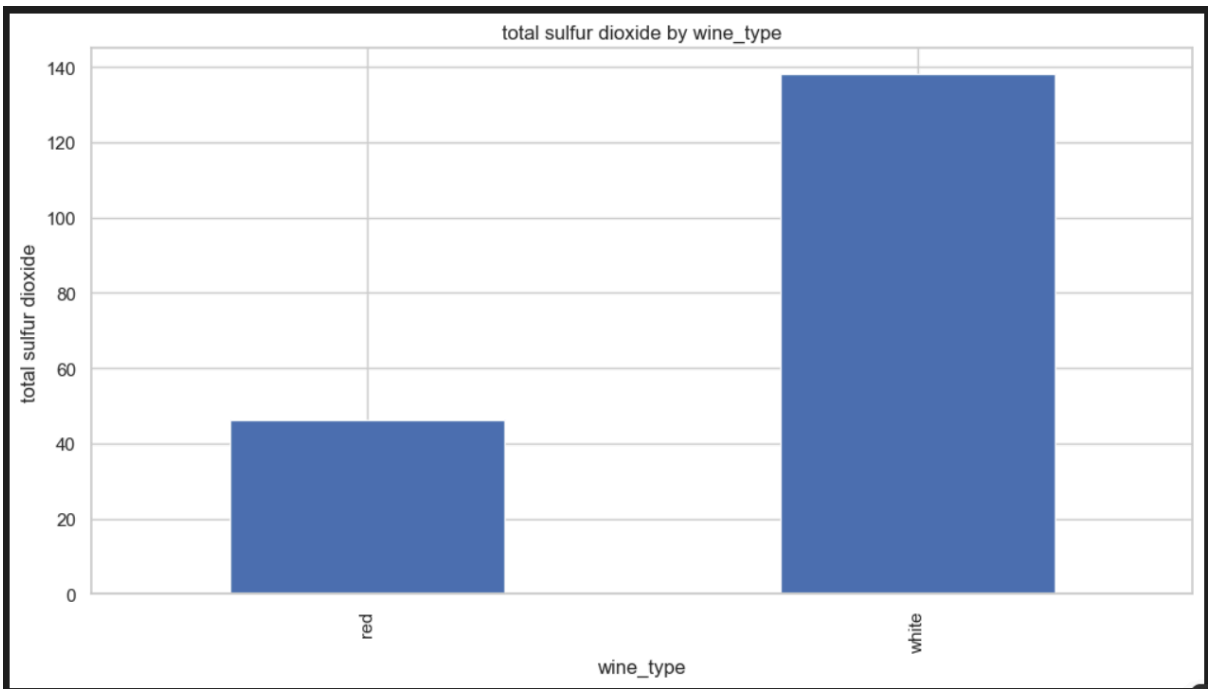
```
wine_df.isna().sum()
```
✓ 0.0s

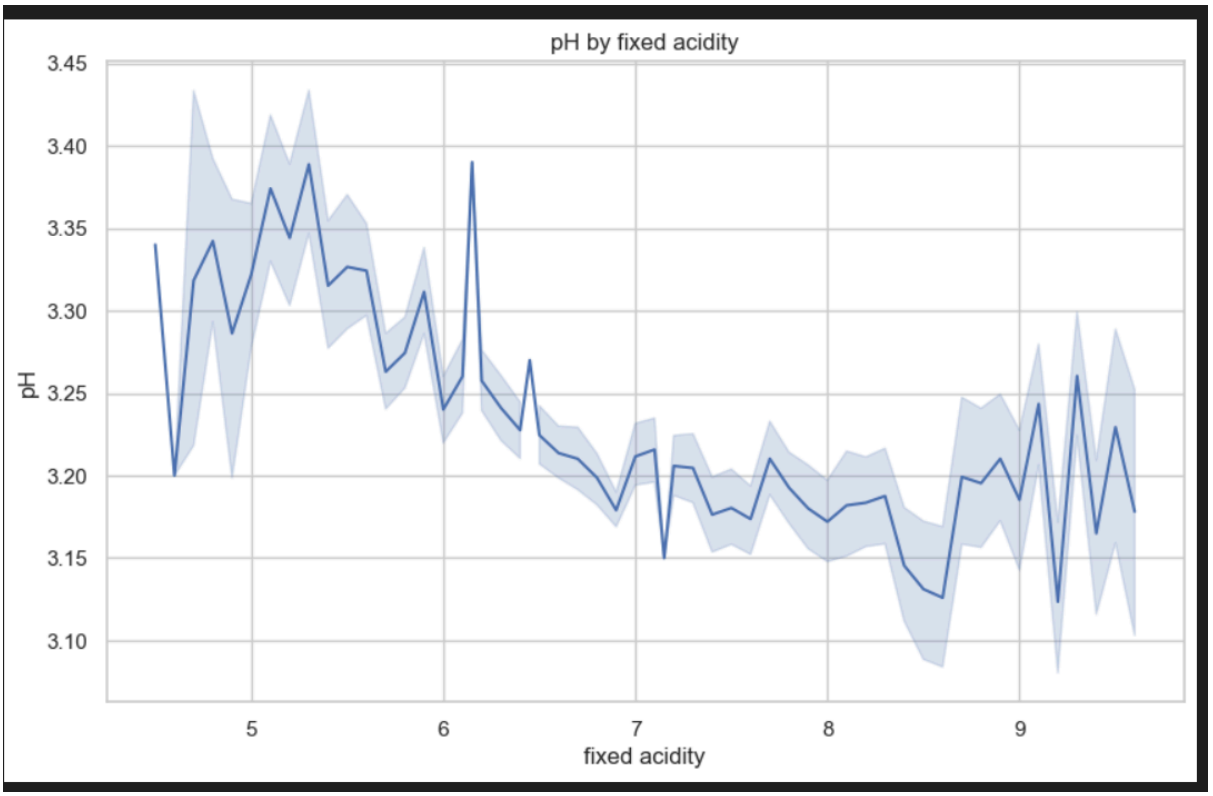
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
wine_type               0
dtype: int64
```

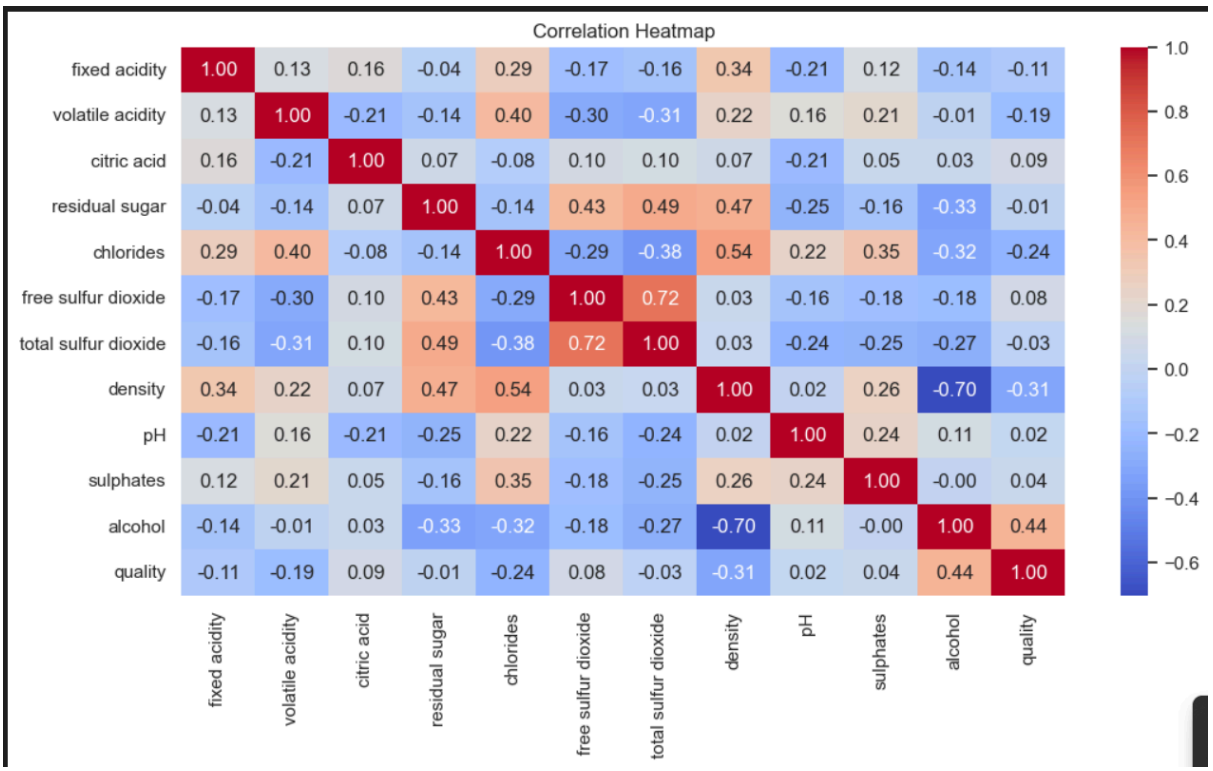
2.c.


Alcohol Content Distribution by Wine Quality
Alcohol Content
14
13
12
11
10
9
8
3
4
5
6
7
8
9
Quality


Alcohol Content vs. Density by Wine Type
wine_type
red
white
Density
1.0025
1.0000
0.9975
0.9950
0.9925
0.9900
0.9875
8
9
10
11
12
13
14
Alcohol Content


total sulfur dioxide by wine_type
total sulfur dioxide
140
120
100
80
60
40
20
0
red
white
wine_type


pH by fixed acidity
pH
3.45
3.40
3.35
3.30
3.25
3.20
3.15
3.10
5
6
7
8
9
fixed acidity



Observation -

1.We can see that the alcohol content of wine generally increases as the quality rating increases.

2.We can see that the red wines have higher alcohol content and lower density compared to white wines.

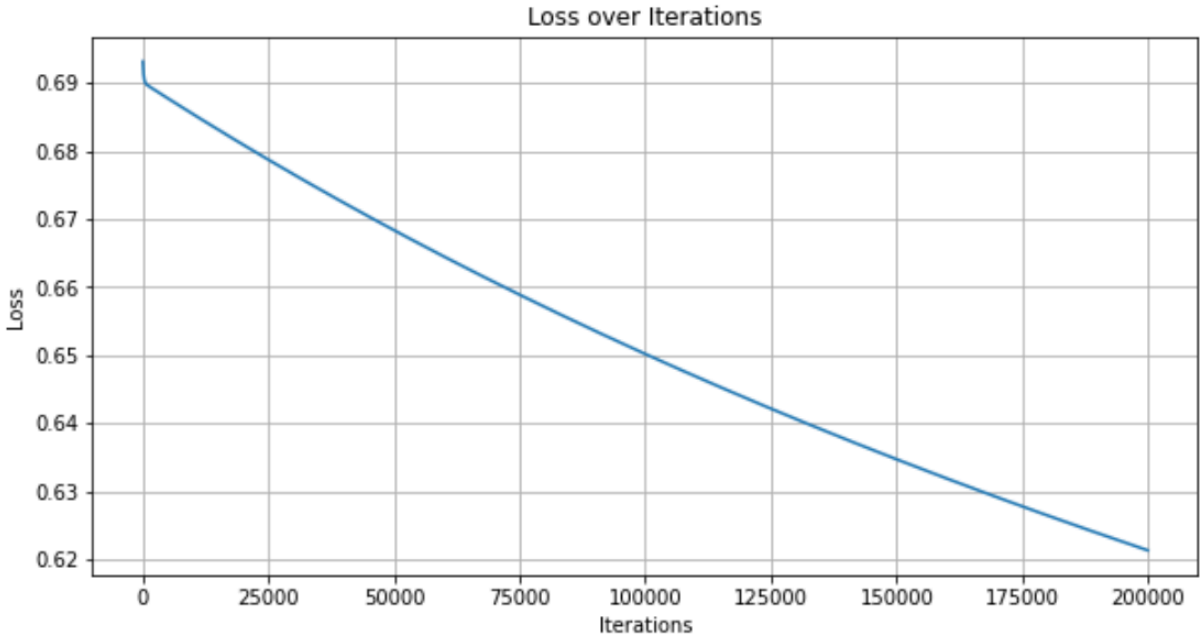
3.We can see that the white wine has higher total sulfur dioxide than the red wine.

4.The graph shows a general trend of decreasing pH as the fixed acidity increases.

5.The correlation matrix shows a strong correlation between the alcohol content and the quality of wine. And also shows a strong correlation between free sulfur dioxide and total sulfur dioxide in wines.

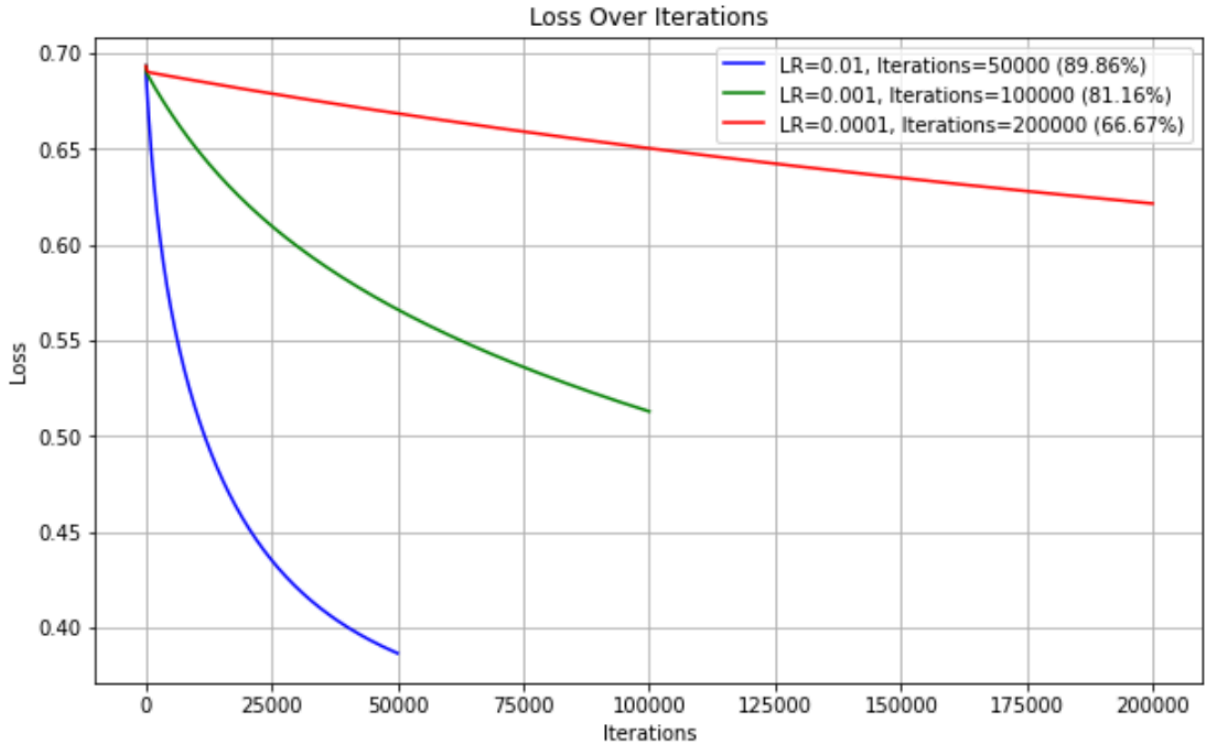
Report for Part 2 - Logistic Regression using Gradient Descent

1. Best accuracy obtained - 89.86% (for learning rate = 0.01 and number of iterations = 50000)
2.



- The graph shows a clear downward trend, indicating that the loss is decreasing as the number of iterations increases. This means the model is learning and improving its performance over time.
- The learning curve is smooth and decreases monotonically, which suggests stable learning without significant fluctuations or oscillations.
- The loss starts at around 0.69 at iteration 0 and by 200,000th iterations, the loss decreases to approximately 0.62. The rate of decrease is higher at the beginning as the curve is steeper and gradually slows down as iterations progress curve becomes flatter.
- While the loss is still decreasing at 200,000 iterations, the rate of decrease has slowed significantly, this suggests the model is approaching convergence.

3.



Case 1: Learning Rate: 0.01, Iterations: 50,000 (Blue Curve)

- This configuration achieves the fastest convergence with respect to reducing the loss. It starts high and quickly drops to a relatively lower value in a shorter number of iterations.
- The flattening of the curve towards the end signifies that the model is reaching a point where more iterations do not cause a significant impact in reducing the loss.
- This combination also achieves the highest accuracy of 89.86%.

Case 2: Learning Rate: 0.001, Iterations: 100,000 (Green Curve)

- As compared to the previous case this achieves a slower rate of convergence, even though the curve slowly shows decline over a large number of iterations but it doesn't converge as quickly as compared to the model with a higher learning rate.
- This configuration also performs decently, reaching an accuracy of 81.16%.

Case 3: Learning Rate: 0.0001, Iterations: 200,000 (Red Curve)

- This configuration has the lowest learning rate as it shows the slowest convergence. The curve is relatively flat indicating that the loss decreases slowly over a large number of iterations.

- Despite having the highest number of iterations, this model has a higher loss and a lower accuracy (66.67%) as compared to the other two configurations.
- Having a really small learning rate hindered the model even after having the greatest number of iterations.

The model with the learning rate = 0.01 and number of iterations = 50,000 achieves both quick convergence and higher accuracy making it the most efficient combination amongst the ones tested. It can also be seen that having a higher learning rate helps the model converge faster but might lead to overfitting, but in this case it worked well.
Having a very low learning rate on the other hand, might not provide optimal results even with a higher number of iterations.
Increasing the number of iterations helps the models with lower learning rates to improve the performance but still having a small learning rate would lead to slow or insufficient convergence.

4. Some benefits of using logistic regression model are:
- Being a linear model, it is easy to implement, understand and efficient to train. It is computationally efficient even with relatively large datasets. The model can converge quickly, especially with well-tuned hyperparameters
- Since logistic regression is a linear model, it is less prone to overfitting as compared to more complex algorithms, especially when regularization techniques like L1 (lasso) or L2 (ridge) are applied.
- It is very efficient when the dataset has features that are linearly separable.

Some drawbacks of using logistic regression model are:
- Non linear problems can't be solved using logistic regression since it has a linear decision surface, that is, it assumes linearity between the dependent variable and the independent variables.
- Logistic Regression is sensitive to outliers, which can disproportionately influence the decision boundary. This can affect model performance if data is not preprocessed properly.

- When we have high dimensional data and the number of features are very large, logistic regression may not perform well unless regularization techniques are applied. It tends to overfit on high dimensional data.