ML_HW2

Mehdi Jamalkhah, SID: 810100111

April 2024

Contents

1 Parzen Window Variance

Proof.

$$
\begin{aligned}
E[\hat{P}(x)] &= E[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{V_N}\Phi(\frac{x-x_i}{V})] \\
&= \frac{1}{N}\sum_{i=1}^{N}E[\frac{1}{V_N}\Phi(\frac{x-x_i}{V})] \\
&\overset{\text{i.i.d}}{=} E[\frac{1}{V_N}\Phi(\frac{x-x_i}{V})] \\
&= \int \frac{1}{V_N}\Phi(\frac{x-x_i}{V})P(x_i)dx_i
\end{aligned} \tag{1.1}
$$

$$
\begin{aligned}
var(\hat{P}_N(x)) = \sigma_N^2(x) &= var(\frac{1}{N}\sum_{i=1}^{N}\frac{1}{V_N}\Phi(\frac{x-x_i}{V})) \\
&= \frac{1}{N^2}\sum_{i=1}^{N}var(\frac{1}{V_N}\Phi(\frac{x-x_i}{V})) \\
&\overset{\text{i.i.d}}{=} \frac{1}{N}var(\frac{1}{V_N}\Phi(\frac{x-x_i}{V})) \\
&= \frac{1}{N}E[\frac{1}{V_N^2}\Phi^2(\frac{x-x_i}{V})] - \underbrace{\frac{1}{N}E^2[\frac{1}{V_N}\Phi(\frac{x-x_i}{V})]}_{\geq 0} \\
&\leq \frac{1}{N}\int \frac{1}{V_N^2}\Phi^2(\frac{x-x_i}{V})P(x_i)dx_i \\
&\leq \frac{1}{N}\frac{sup(\Phi)}{V_N}\underbrace{\int \frac{1}{V_N}\Phi(\frac{x-x_i}{V})P(x_i)dx_i}_{=\text{Equation (1.1)}} \\
&\leq \frac{sup(\Phi)E[\hat{P}(x)]}{NV_N}
\end{aligned}
$$

□

2 Parzen Window Estimator

2.1

$$\begin{aligned}
\text{Equation (1.1)} \longrightarrow \bar{P}_n(x) &= \int \frac{1}{V_n}\Phi(\frac{x-x_i}{h_n})P(x_i)dx_i \\
x<0 \longrightarrow \bar{P}_n(x) &= \int_{-\infty}^{0} \frac{1}{V_n} e^{\frac{xi-x}{h}}(0)dx_i \\
&= 0
\end{aligned}$$

$$\begin{aligned}
0 \le x \le a \longrightarrow \bar{P}_n(x) &= \int_0^x \frac{1}{V_n} e^{\frac{xi-x}{h_n}} \frac{1}{a} dx_i \\
&= \left[\frac{h}{aV_n} e^{\frac{x_i-x}{h}}\right]_0^x \\
(d=1 \to h_n = v_n) \Rightarrow \qquad &= \frac{1}{a}(1-e^{-\frac{x}{h_n}})
\end{aligned}$$

$$\begin{aligned}
x \ge 0 \longrightarrow \bar{P}_n(x) &= \int_0^a \frac{1}{V_n} e^{\frac{xi-x}{h_n}} \frac{1}{a} dx_i \\
&= \left[\frac{h}{aV_n} e^{\frac{x_i-x}{h_n}}\right]_0^a \\
(d=1 \to h_n = v_n) \Rightarrow \qquad &= \frac{1}{a}(e^{\frac{a}{h_n}}-1)e^{-\frac{x}{h_n}}
\end{aligned}$$

2.2

$$\begin{aligned}
bias_percent(\hat{p}(x)) &= \frac{E[\hat{p}(x)]-p(x)}{p(x)} \\
&= \frac{\frac{1}{a}(1-e^{-\frac{x}{h_n}})-\frac{1}{a}}{\frac{1}{a}} \\
&= -e^{-\frac{x}{h_n}} \\
\frac{-1}{100} < |bias| < \frac{1}{100} &\to \frac{-1}{100} < e^{\frac{x_0}{h_n}} < \frac{1}{100} \\
&\to -\frac{x_0}{h_n} < -2ln(10) \\
&\to h_n < \frac{x_0}{2ln(10)}
\end{aligned}$$

3 Regularization

3.1

Proof.

$$\begin{aligned}
\hat{\beta} &= argmin_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda||\beta||_2^2 \\
L &= \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda||\beta||_2^2 \\
&= ||Y - A\beta||_2^2 + \lambda||\beta||_2^2 \\
&= (Y - A\beta)^T(Y - A\beta) + \lambda||\beta||_2^2 \\
&= Y^TY - Y^TA\beta - \beta^TA^TY + \beta^TA^TA\beta + \lambda||\beta||_2^2 \\
\frac{dL}{d\beta} &= 2A^TA\beta - A^TY - A^TY + 2\lambda\beta \\
&= 2A^TA\beta - 2A^TY + 2\lambda\beta
\end{aligned}$$

$$\begin{aligned}
\frac{dL}{d\beta} &= 0 \rightarrow A^TA\hat{\beta} - A^TY + \lambda\hat{\beta} = 0 \\
A^TY &= (A^TA + \lambda I)\hat{\beta} \\
\hat{\beta} &= (A^TA + \lambda I)^{-1}A^TY
\end{aligned}$$

□

3.2

L1 regularization, also known as Lasso regularization, prevents overfitting by introducing a penalty term into the model's loss function based on the absolute values of the model's parameters. L1 regularization seeks to reduce some model parameters toward zero in order to lower the number of non-zero parameters in the model (sparse model).

$$J(\theta) = \sum_{i=1}^{n} cost(h_\theta(x^{(i)}, y^{(i)}) + \lambda \sum_{j=1}^{d} |\theta_j|$$

L2 regularization, also known as Ridge regularization, is technique that avoids overfitting by introducing a penalty term into the model's loss function(same as L1 regularization) based on the squares of the model's parameters. The goal of L2 regularization is to keep the model's parameter sizes short and prevent oversizing.

$$J(\theta) = \sum_{i=1}^{n} cost(h_\theta(x^{(i)}, y^{(i)}) + \lambda||\theta_j||_2^2$$

Difference between L1 L2 regularization:

1. L1 Reg. Selects a subset of the most important features, but in L2 Reg. all features are used by the model.
2. L1 Optimization is non-convex, but L2 optimization is convex.

3. L1 Reg. is sensitive to outliers, but L2 Reg. is robust to outliers.

4. L1 Reg. produces sparse solutions (some parameters are shrunk towards zero), but L2 Reg. produces non-sparse solutions (all parameters are used by the model).

4 Nearest-Neighbor Rule

In this question, we will attempt to plot the decision boundary for the nearest neighbor (NN) classifier.

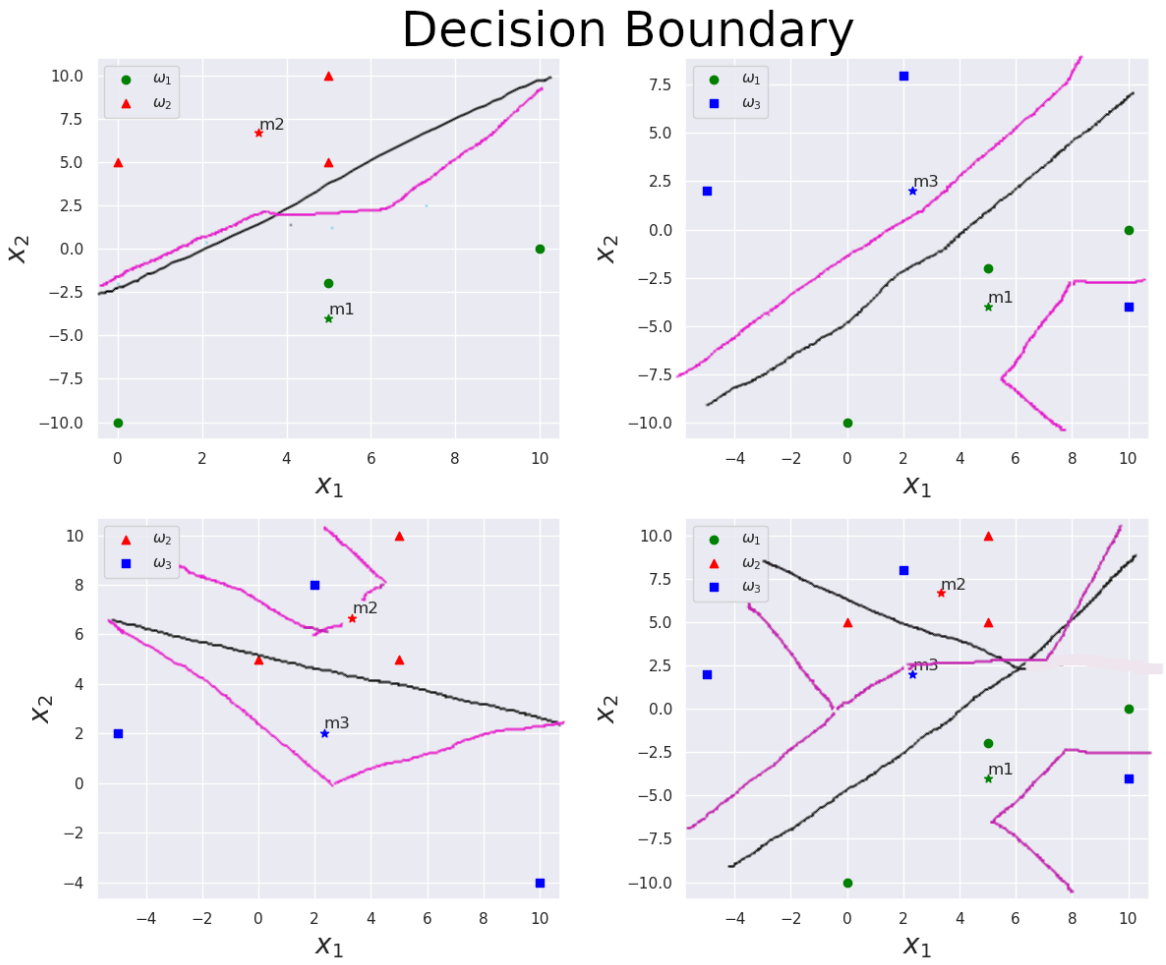

Figure 4.1: One nearest neighbor vs nearest mean decision boundary

The red line represents the decision boundary based on the nearest neighbor algorithm, while the black line represents the decision boundary based on the nearest mean algorithm. It is evident that the nearest neighbor algorithm creates a more complex decision boundary, which enables it to construct a stronger classifier. However, this increased complexity also introduces a higher risk of overfitting the data.

5 Bayes vs Nearest-Neighbor Classifier

5.1

suppose: $P(w_1) = P(w_2)$

$$\begin{aligned}\hat{w} = 1 &\rightarrow P(w_1|x) > P(w_2|x) \\ &\rightarrow P(x|w_1)P(w_1) > P(x|w_2)P(w_2) \\ &\rightarrow P(x|w_1) > P(x|w_2)\end{aligned}$$

$$\rightarrow \begin{cases} \hat{w} = 1 & \text{if } 0 \leq x \leq \frac{1}{3} \\ \hat{w} = 2 & \text{if } \frac{1}{3} < x \leq 1 \end{cases}$$

$$\begin{aligned}P(erroe) &= P(w_1) - \int_{R_1}[P(x|w_1)P(w_1) - P(x|w_2)P(w_2)]dx \\ &= \frac{1}{2} - \int_0^{\frac{1}{3}}(\frac{3}{2} * \frac{1}{2})dx \\ &= \frac{1}{2} - \frac{1}{4} \\ &= \frac{1}{4}\end{aligned}$$

5.2

$$\begin{aligned}P_1(e) &= \int P(x)P(e|x)dx \\ &= \int P(x)(P(w_1|x)P(w_2|x) + P(w_2|x)P(w_1|x))dx \\ &= 2 * \int P(x)P(w_1|x)P(w_2|x)dx \\ &= 2 * [\int_0^{\frac{1}{3}} P(x)P(w_1|x)P(w_2|x)dx + \\ &\int_{\frac{1}{3}}^{\frac{2}{3}} P(x)P(w_1|x)P(w_2|x)dx + \int_{\frac{2}{3}}^{1} P(x)P(w_1|x)P(w_2|x)dx] \\ &= 2 * \int_{\frac{1}{3}}^{\frac{2}{3}} P(x)P(w_1|x)P(w_2|x)dx \\ &= 2 * \frac{1}{2} * {}^{\circ}\frac{1}{2}\int_{\frac{1}{3}}^{\frac{2}{3}} P(x)dx \\ &= \frac{1}{2}\int_{\frac{1}{3}}^{\frac{2}{3}} \int P(x|w)P(w)dwdx \\ &= \frac{1}{2}\int_{\frac{1}{3}}^{\frac{2}{3}} \frac{3}{2}dx \\ &= \frac{1}{2} * \frac{1}{3} * \frac{3}{2}dx \\ &= \frac{1}{4}\end{aligned}$$

6 Parzen Window

In this section, we estimate a distribution using the Parzen window and compare the results for different values of V_N and N. You can observe the true and estimated distribution in Figure. 6.1.

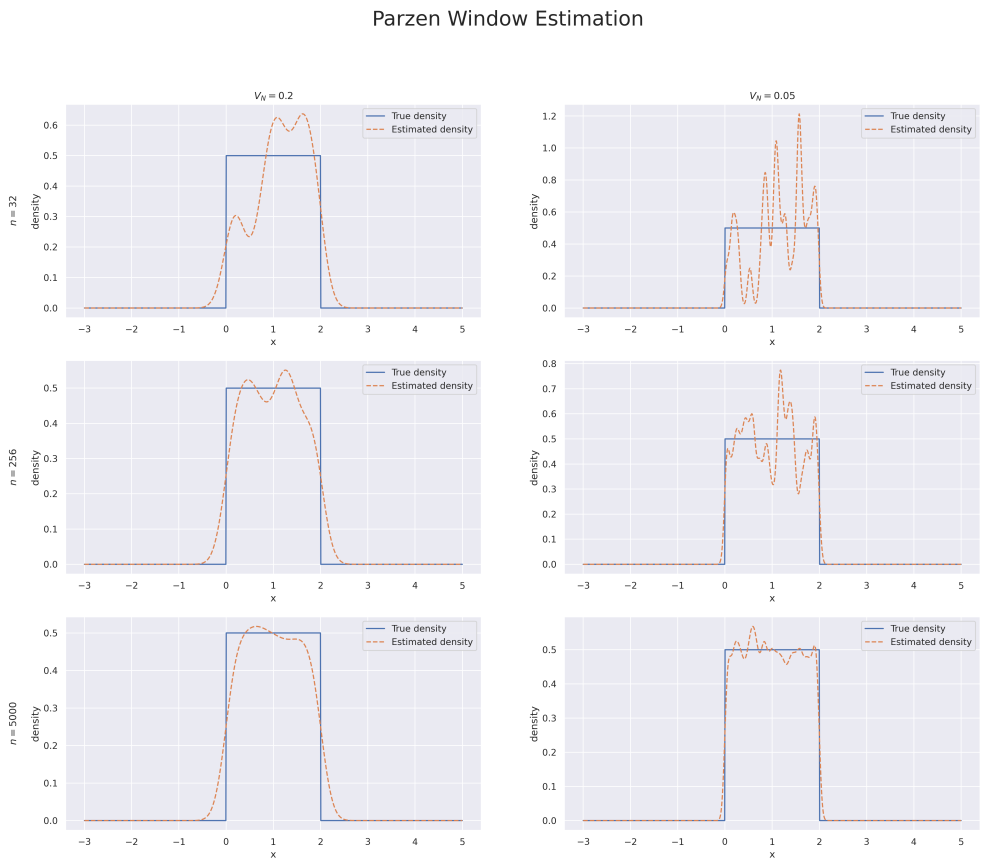

Figure 6.1: Parzen window estimation and true distribution

As V_N decreases, the estimation becomes more accurate. However, the estimated distribution may become too peaked since each sample contributes significantly to the region near itself. On the other hand, when V_N increases, we obtain a smoother curve but with more error.

Regarding N, as N increases, the estimated distribution becomes more accurate because we have more knowledge about the true distribution. In the limit as N approaches infinity, we obtain the exact distribution.

7 Parzen-window Estimates and Classifiers

In this section, we will estimate three normal distributions using the Parzen estimator. We will consider two different sample sizes for each class: 50 and 500. The samples are displayed in the figures below:

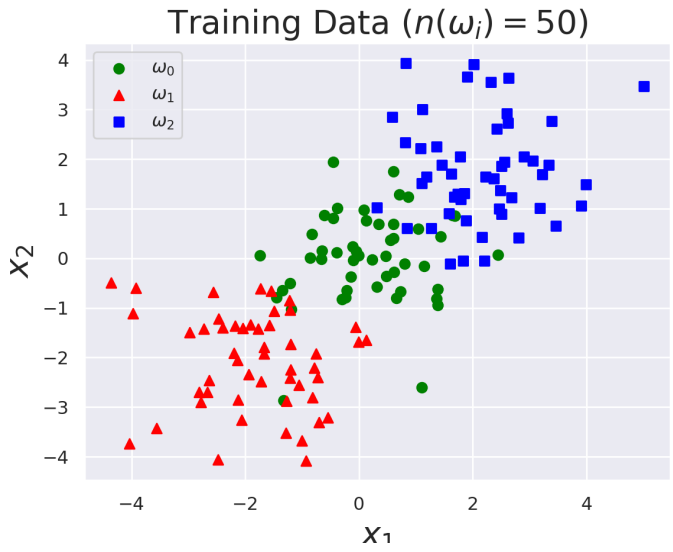

Figure 7.1: Training data($n = 50$)

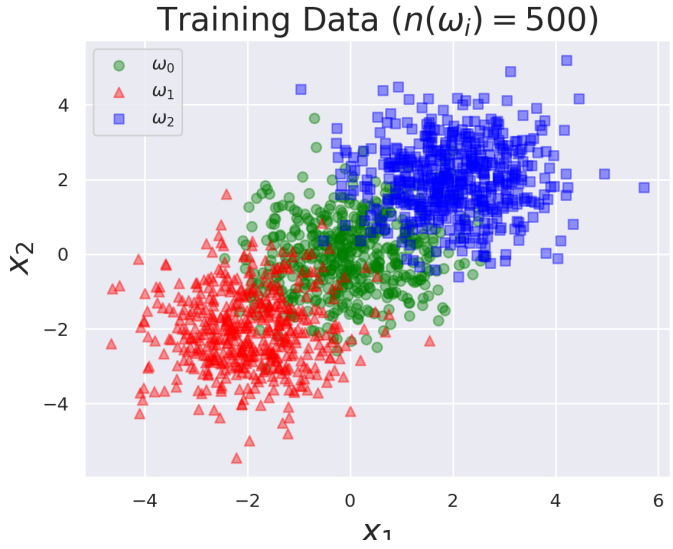

Figure 7.2: Training data($n = 500$)

We considered four test data points and used the estimated distribution along with the Bayes classifier to predict their classes. Here are the results:

	$N = 50$	$N = 500$
$V_N = 1$	0.50	0.50
$V_N = 0.1$	0.50	0.75

Table 7.1: Accuracy of Bayes classifier with Parzen window

To gain a clearer understanding of the estimated distribution, let's create a plot of the decision

boundary.

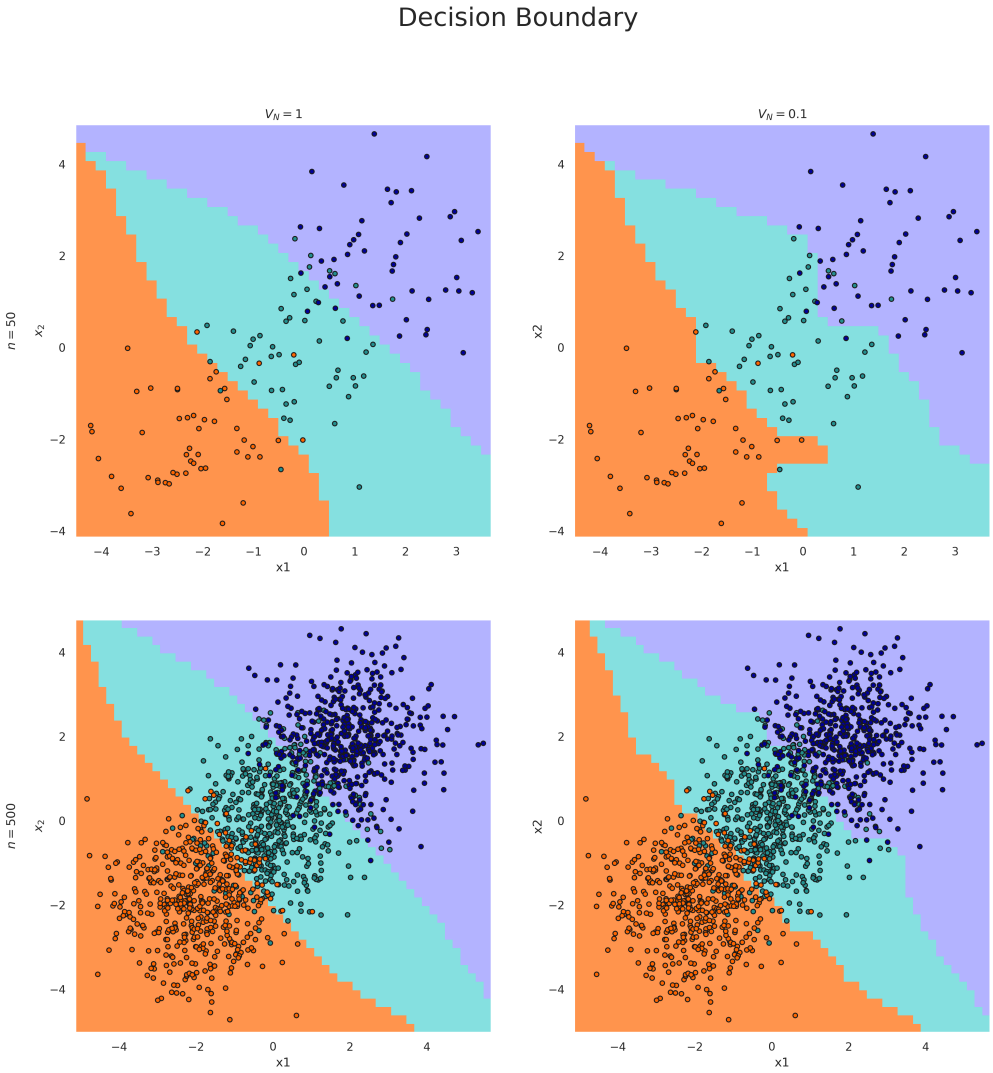

Figure 7.3: Decision boundary of bayes classifier with parzen window

As the number of samples increases, the estimated distribution becomes closer to the true distribution. Additionally, as the value of V_N decreases, the decision boundary becomes more complex and sharp.

In this scenario, reducing the value of V_N assists in estimating the distribution more accurately. As a result, the accuracy of the last estimator is higher compared to the others.

8 Logistic Regression and KNN Classifiers

In this section, we will perform classification on the Wheat Seeds dataset using two algorithms: logistic regression and k-nearest neighbors (KNN).

8.1 Exploratory Data Analysis (EDA)

You can view the scatter plot and histogram of all the features of the dataset in Figure. 8.1.

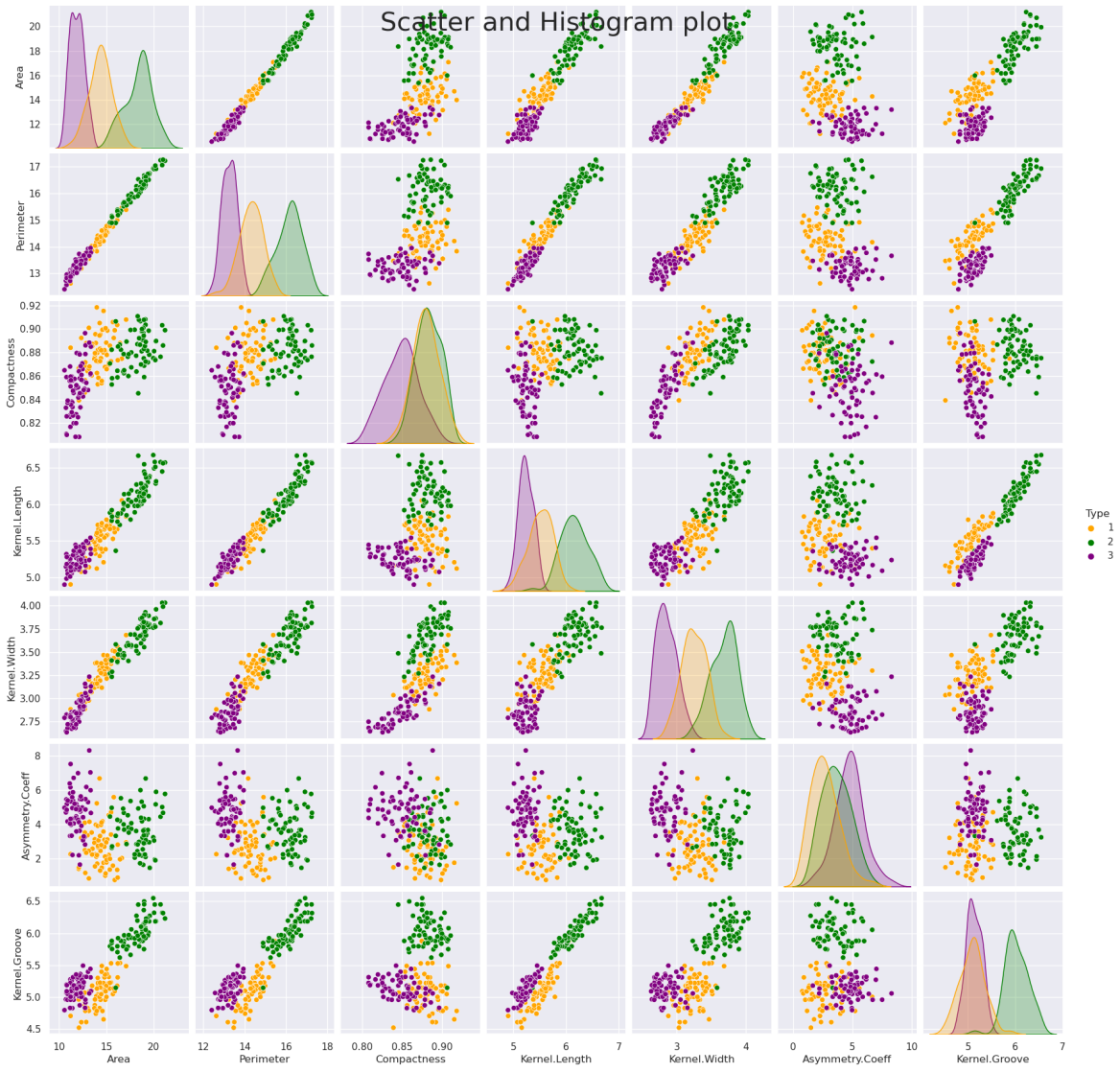

Figure 8.1: Scatter and Histogram Plot

Based on the above plot, it appears that certain pairs of features, such as Area and Perimeter, Area and Kernel Groove, and Kernel Groove and Kernel Length, can better separate the samples compared to other feature pairs.

When considering individual features and examining the distribution of different classes based on

those features, features like Kernel Width, Perimeter, and Area seem to provide better separation between the classes, with less overlap in their distributions.

It is also noticeable that all the histograms indicate that the distributions follow a normal distribution pattern.

8.2 Preprocessing and Normalization

To ensure that no particular feature overshadows another and that our model treats all features equally, we perform feature normalization. This involves adjusting all features to have a mean of 0 and a variance of 1. By doing so, we bring all features to the same scale, allowing for fair comparison.

8.3 Classification

We applied the one-vs-all technique using logistic regression to classify the data, and the results are as follows:

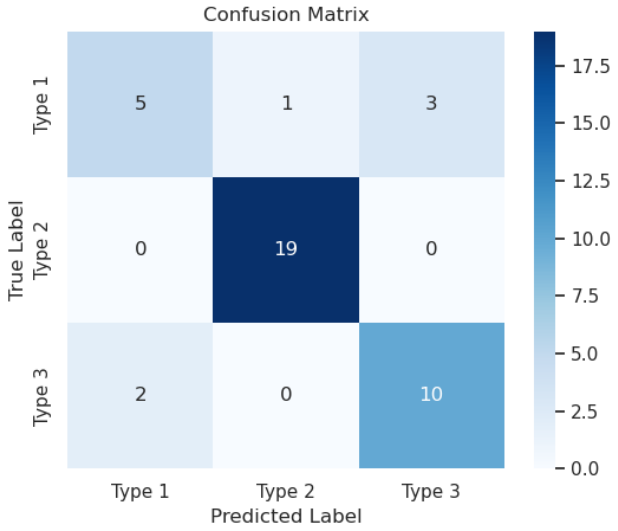

Figure 8.2: Confusion matrix for logistic regression

	precision	recall	f1-score
Type 1	0.71	0.56	0.63
Type 2	0.95	1.00	0.97
Type 3	0.77	0.83	0.80
accuracy			0.85
macro avg	0.81	0.80	0.80
weighted avg	0.84	0.85	0.84

Table 8.1: Classification report for logistic regression

8.4 KNN Classifier

Now, let's apply the K-Nearest Neighbors (KNN) algorithm to this dataset. We will try various values of k to find the best k for our classification task.

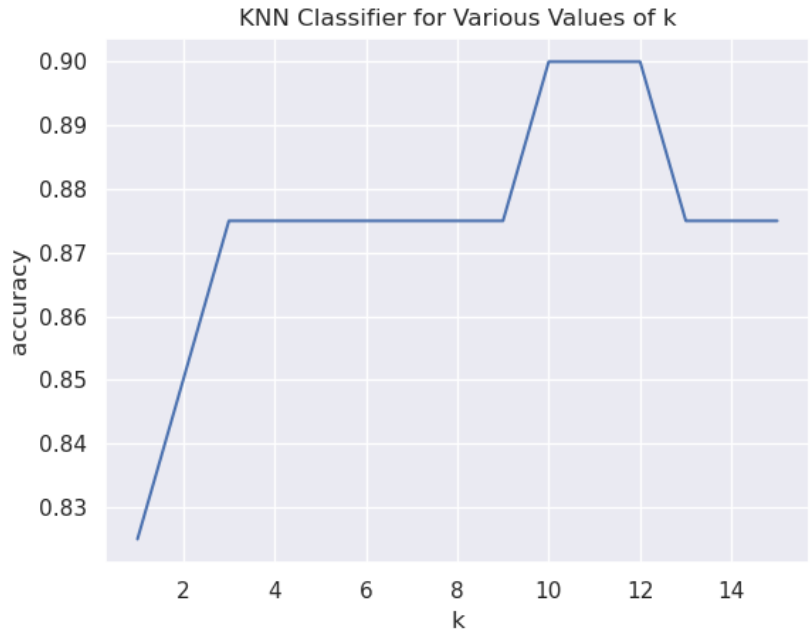

Figure 8.3: KNN Classifier for Various Values of k

As observed, the accuracy tends to increase as the value of k increases in the KNN algorithm. However, there comes a point where increasing k further leads to a decrease in accuracy. This occurs when the model becomes overfitted to the training data, resulting in poor performance on the test data.

To strike a balance between complexity and overfitting, it is advisable to choose a suitable value for k. In this case, k = 10 seems to be a good choice, as it captures sufficient complexity of the data without overfitting.

9 Linear Regression

In this section, our goal is to predict the number of customer purchases in a market. It consists of two phases. In the first phase, we will develop a **linear regression** model from scratch. Then, in the second phase, we will apply the **gradient descent** method.

9.1 Exploratory Data Analysis (EDA)

To gain a more comprehensive understanding of the dataset, we will examine the statistical properties of the features. This exploration will offer valuable insights into the data.

9.1.1 Missing Data Proportion

You can see the proportion of missing data in following table:

#	Column	Missing Data Count	Missing Data Proportion (%)
1	ID	0	0.00
2	Year_Birth	0	0.00
3	Education	0	0.00
4	Marital_Status	0	0.00
5	Income	223	9.96
6	Kidhome	0	0.00
7	Teenhome	0	0.00
8	Dt_Customer	0	0.00
9	Recency	0	0.00
10	MntCoffee	205	9.15
11	MntFruits	0	0.00
12	MntMeatProducts	0	0.00
13	MntFishProducts	0	0.00
14	MntSweetProducts	0	0.00
15	MntGoldProds	13	0.58
16	NumWebVisitsMonth	200	8.93
17	Complain	0	0.00
18	NumPurchases	0	0.00
19	UsedCampaignOffer	0	0.00

Table 9.1: Missing data proportion

A significant portion of the data, approximately 10 %, is missing for three features. We will address this issue in the following sections.

9.1.2 Histogram Plot

Histogram of features is as follows:

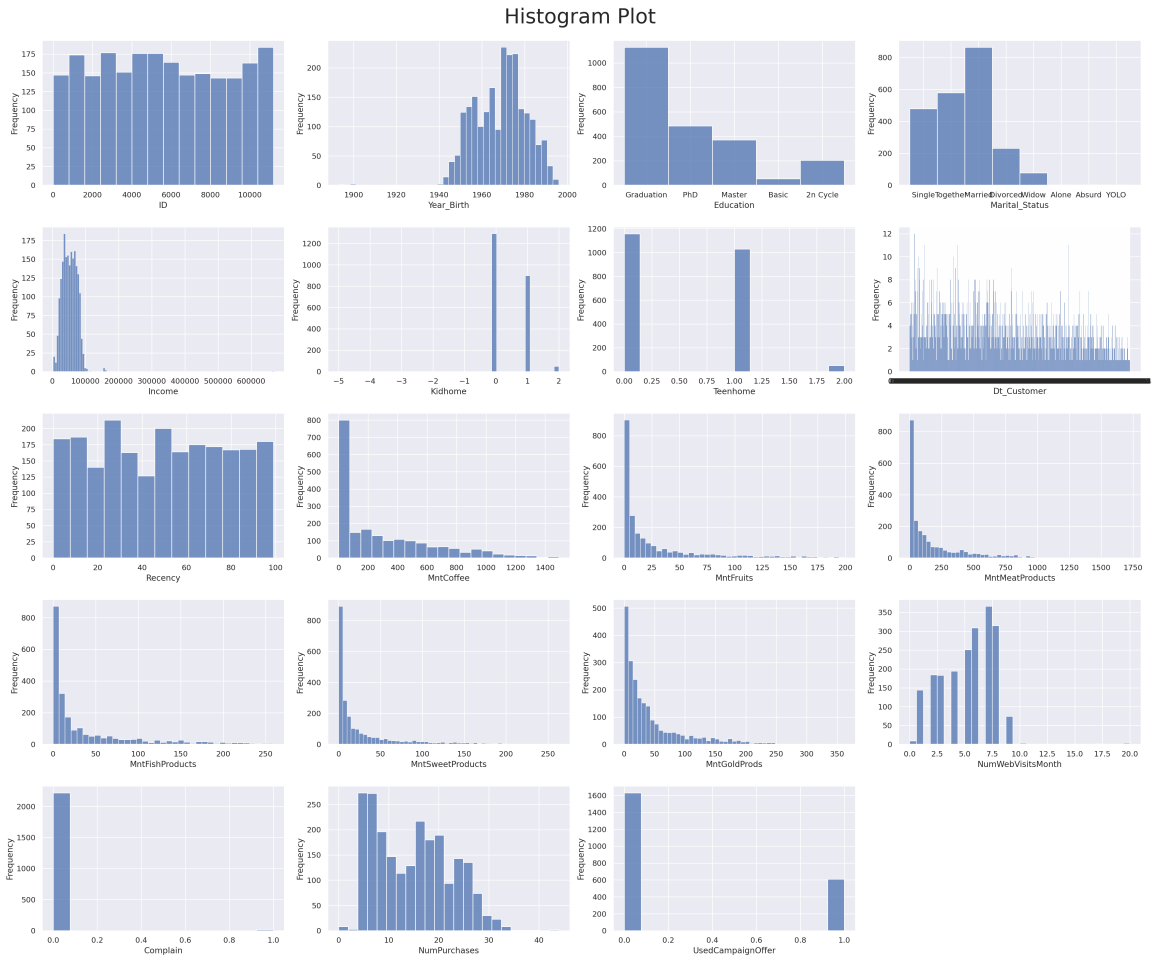

Figure 9.1: Histogram of features

As you can observe, the target feature (`NumPurchases`) follows a normal distribution, while many other features exhibit a gamma distribution(like `MntCoffee`, `MntFruits`, and `MntMeatProducts`). Additionally, some features are categorical and discrete in nature(like `Education`).

9.1.3 Scatter Plot

To understand the relationship between each feature and the target feature, we plot their scatter plot.

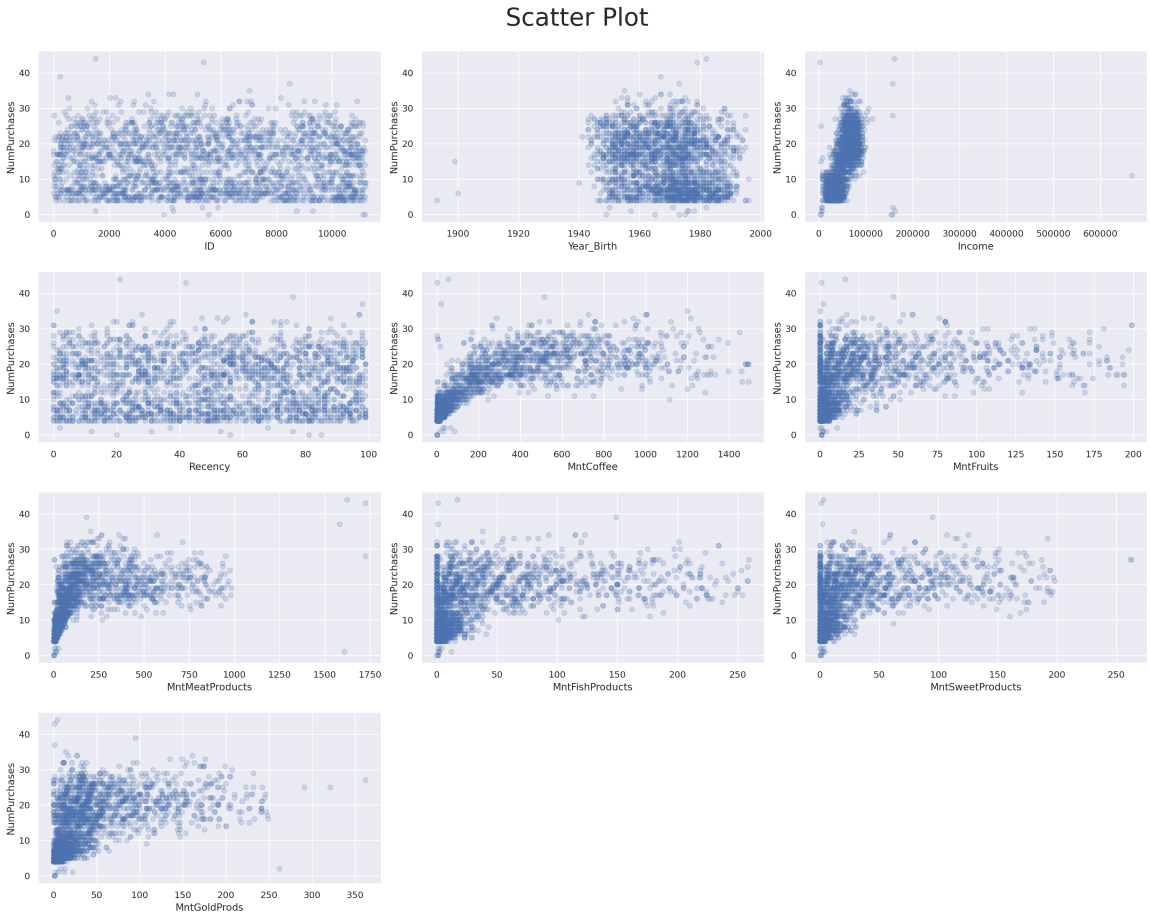

Figure 9.2: Scatter plot of features

Some features, such as `income` and `MntCoffee`, exhibit a strong linear relationship with the target feature. On the other hand, features like `MntMeatProducts` and `MntFruits` demonstrate a somewhat linear relationship, although not as prominent as `MntCoffee`.

This linear relationship is crucial for us because we aim to apply linear regression to this dataset, and the model heavily relies on the linear relationship between features.

9.1.4 Correlation Matrix

Let's examine the correlation between the features.

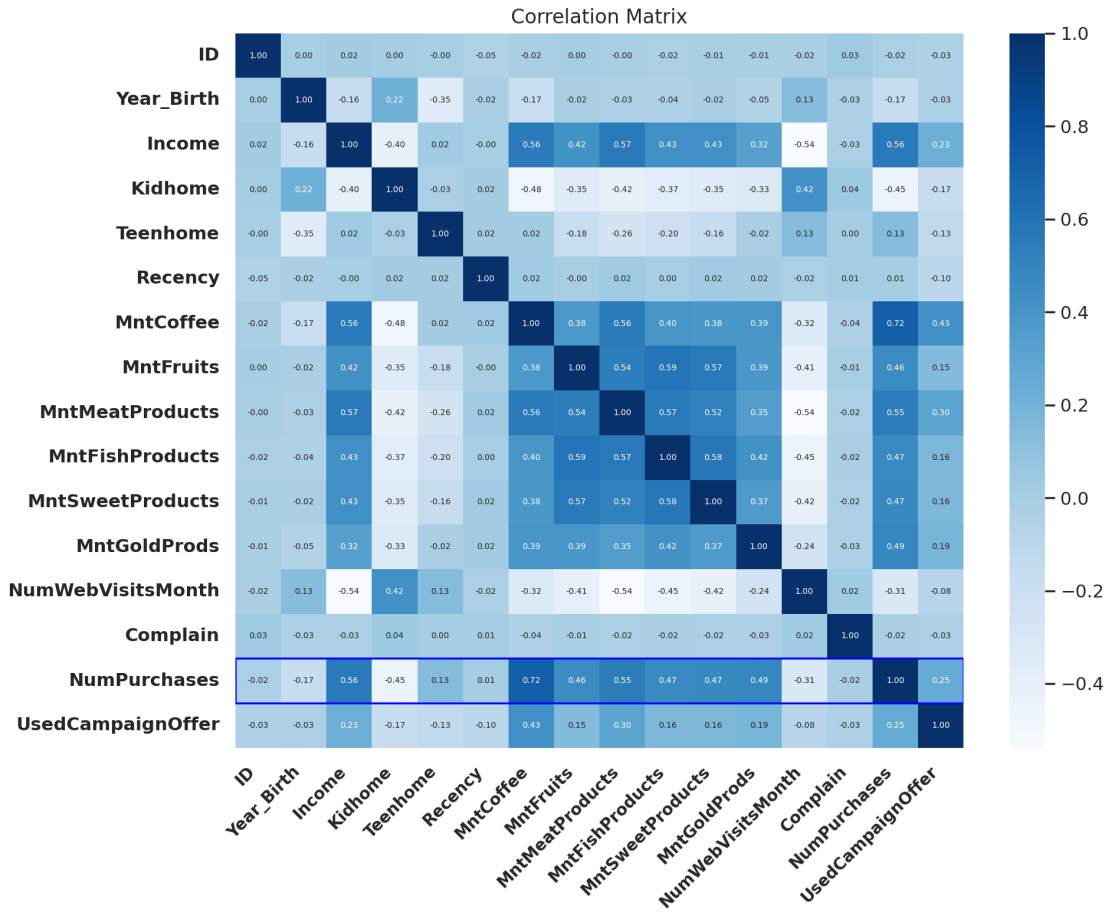

Figure 9.3: Correlation matrix

There is a black box around target feature in Figure. 9.3 and the most correlated features are as follows:

Rank	Feature	Correlation
1	MntCoffee	0.72
2	Income	0.56
3	MntMeatProducts	0.55
4	MntGoldProds	0.49
5	MntSweetProducts	0.47

Table 9.2: Top features with the highest correlation to the target feature

We will utilize this table in subsequent sections when selecting specific features for model training.

9.2 Preprocessing and Normalization

In this section we try to make the dataset more clean and completed, such as fill missing values and converting the non-numerical values to numerical.

9.2.1 Handling Missing Value

Here are some common methods for handling missing values:

1. **Dropping rows or columns**: If the missing values are relatively few we can simply remove the rows or columns containing missing values. However, this approach may result in a loss of valuable information.

2. **Mean, median, or mode imputation**: This approach assumes that the missing values follow a similar distribution as the observed values.

3. **Advanced imputation methods**: Machine learning algorithms, such as k-nearest neighbors (KNN) or regression models, can be used to predict missing values based on other features in the dataset.

Here we use method 2 and replace the missing values with their mode.

9.2.2 Handling Non-numerical Features

In our dataset there are 3 non-numerical features, named, `Education`, `Marital_Status`, and `Dt_Customer`. We apply three different ways for converting them to numerical value.

In general some ways for solving this problem are as follows:

1. **Label Encoding**: Assigning a unique numeric label to each unique category or class in the data.

2. **One-Hot Encoding**: Creating binary columns to represent each category. Each category is encoded as a separate binary feature, with a value of 1 indicating the presence of that category and 0 otherwise. One-hot encoding is suitable for nominal categorical variables where there is no inherent order or hierarchy.(we are going to use it for `Marital_Status`)

3. **Ordinal Encoding**: Assigning numerical values to categories based on their order or rank. (we are going to use it for `Education` which obviously has an order.)

9.2.3 Normalization

We normalize each feature to have a mean of 0 and a standard deviation of 1.

9.2.4 Train Test Split

We split the dataset into a training set and a test set, maintaining a proportion of 80% for training and 20% for testing.

9.3 Modeling

9.3.1 Simple Linear Regression

Main form of simple linear regression function:

$$f(x) = \alpha x + \beta$$

Here we want to find the bias (α) and slope(β) by minimizing the derivation of the Root Mean Square Error (RMSE) function:

- step 1: Compute RMSE of the training data

$$RMSE = \sqrt{\frac{\Sigma(y_i - (\hat{\beta} + \hat{\alpha} * x_i))^2}{N}}$$

- step 2: Compute the derivatives of the RMSE function in terms of α and β, and set them equal to 0 to find the desired parameters

$$\begin{aligned}
\frac{\partial RMSE}{\partial \beta} = 0 \rightarrow \Sigma(-y_i + \hat{\beta} + \hat{\alpha} * x_i) &= 0 \\
-\Sigma y_i + \Sigma\hat{\beta} + \hat{\alpha}\Sigma x_i &= 0 \\
\Sigma y_i - \hat{\alpha}\Sigma x_i &= m\hat{\beta} \\
\bar{y_i} - \hat{\alpha}\bar{x_i} &= \hat{\beta} \\
\beta = \bar{y} - \hat{\alpha}\bar{x} &
\end{aligned} \tag{9.1}$$

$$\begin{aligned}
\frac{\partial RMSE}{\partial \alpha} = 0 \rightarrow \Sigma(-2x_i y_i + 2\hat{\beta}x_i + 2\hat{\alpha}x_i^2) &= 0 \\
-\Sigma x_i y_i + \hat{\beta}\Sigma x_i + \hat{\alpha}\Sigma x_i^2 &= 0 \\
\text{Equation. (9.1)} \rightarrow -\Sigma x_i y_i + (\bar{y} - \hat{\alpha}\bar{x})\bar{x_i} + \hat{\alpha}\Sigma x_i^2 &= 0 \\
\rightarrow \alpha = \frac{\Sigma x_i y_i - \bar{x}\bar{y}}{\Sigma x_i^2 + \bar{x}^2} &
\end{aligned}$$

Finally,

$$\begin{aligned}
\hat{\alpha} &= \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2} \\
\hat{\beta} &= \bar{y} - \hat{a}\bar{x}
\end{aligned}$$

Based on the above formula, we implement the functions to compute the parameters of a simple linear regression.

9.3.1.1 Feature Selection Here we select `MntCoffee` because it exhibits a stronger linear relationship with the target feature, since we are training a linear regression model, also it has the highest correlation with target feature which means can provide valuable information about the target feature.

Regardless of the previous paragraph, let's try all features individually and compare the results with our expectations.

#	Feature	RMSE	R2 Score
0	ID	7.55	0.01
1	Year$_{B}irth$	7.55	0.01
2	Education	7.46	0.01
3	Income	6.61	0.23
4	Kidhome	6.94	0.15
5	Teenhome	7.52	0.00
6	Recency	7.55	0.01
7	MntCoffee	5.62	0.44
8	MntFruits	6.99	0.14
9	MntMeatProducts	6.37	0.28
10	MntFishProducts	7.03	0.13
11	MntSweetProducts	6.82	0.18
12	MntGoldProds	6.76	0.19
13	NumWebVisitsMonth	7.34	0.05
14	Complain	7.55	0.01
15	UsedCampaignOffer	7.34	0.05
16	Marital$_{S}tatus_{A}bsurd$	7.55	0.01
17	Marital$_{S}tatus_{A}lone$	7.55	0.01
18	Marital$_{S}tatus_{D}ivorced$	7.55	0.01
19	Marital$_{S}tatus_{M}arried$	7.55	0.01
20	Marital$_{S}tatus_{S}ingle$	7.54	0.01
21	Marital$_{S}tatus_{T}ogether$	7.55	0.01
22	Marital$_{S}tatus_{W}idow$	7.55	0.01
23	Marital$_{S}tatus_{Y}OLO$	7.55	0.01

Table 9.3: Simple linear regression result based on each feature

As observed, the feature `MntCoffee` yields the highest R2 score.

9.3.2 Multiple Regression

Now we will performs a gradient descent. The basic premise is simple. Given a starting point we update the current weights by moving in the negative gradient direction of cost function. Recall that the gradient is the direction of increase and therefore the negative gradient is the direction of decrease and we're trying to minimize a cost function.

The amount by which we move in the negative gradient direction is called the step size. We stop when we are sufficiently close to the optimum. We define this by requiring that the magnitude (length) of the gradient vector to be smaller than a fixed tolerance.

9.3.2.1 Features Selection We will choose three features based on their strong linear relationships and high correlation with the target feature. In the EDA section, we identified the features `MntCoffee`, `Income`, and `MntMeatProducts` as the most suitable.

The result of this model is as follows:

RMSE	R2 Score
5.29	0.50

Table 9.4: Multiple regression result

9.3.3 Compare Results

The accuracy of multiple regression is superior to that of linear regression, as expected. This improvement stems from the utilization of a greater number of features for prediction, enabling us to perceive our samples in a more comprehensive and refined manner. But multiple regression takes more time to train, which will be noticeable when the number of selected features increases.