Mini Project 1
Report 1
Parsa Rashtian - 40010823

1.1: The following csv data file includes data from consumers of a credit card in a bank. The manager of the bank really appreciates if we can find a way to predict who is gonna get churned so they can go to that customer to provide better services. Now, this dataset consists of 10,000 customers mentioning their age, salary, marital_status, credit card limit, credit card category, etc. There are nearly 18 features.

Here is all of it's columns which show us what information are featured in this dataset:

```
Index(['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
       'Income_Category', 'Card_Category', 'Months_on_book',
       'Total_Relationship_Count', 'Months_Inactive_12_mon',
       'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
       'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
       'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio',

'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon
_Dependent_count_Education_Level_Months_Inactive_12_mon_1',

'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon
_Dependent_count_Education_Level_Months_Inactive_12_mon_2'],
      dtype='object')
```

We can see it has 23 columns.

Here is an example of first 5 rows to see how is the data structure:

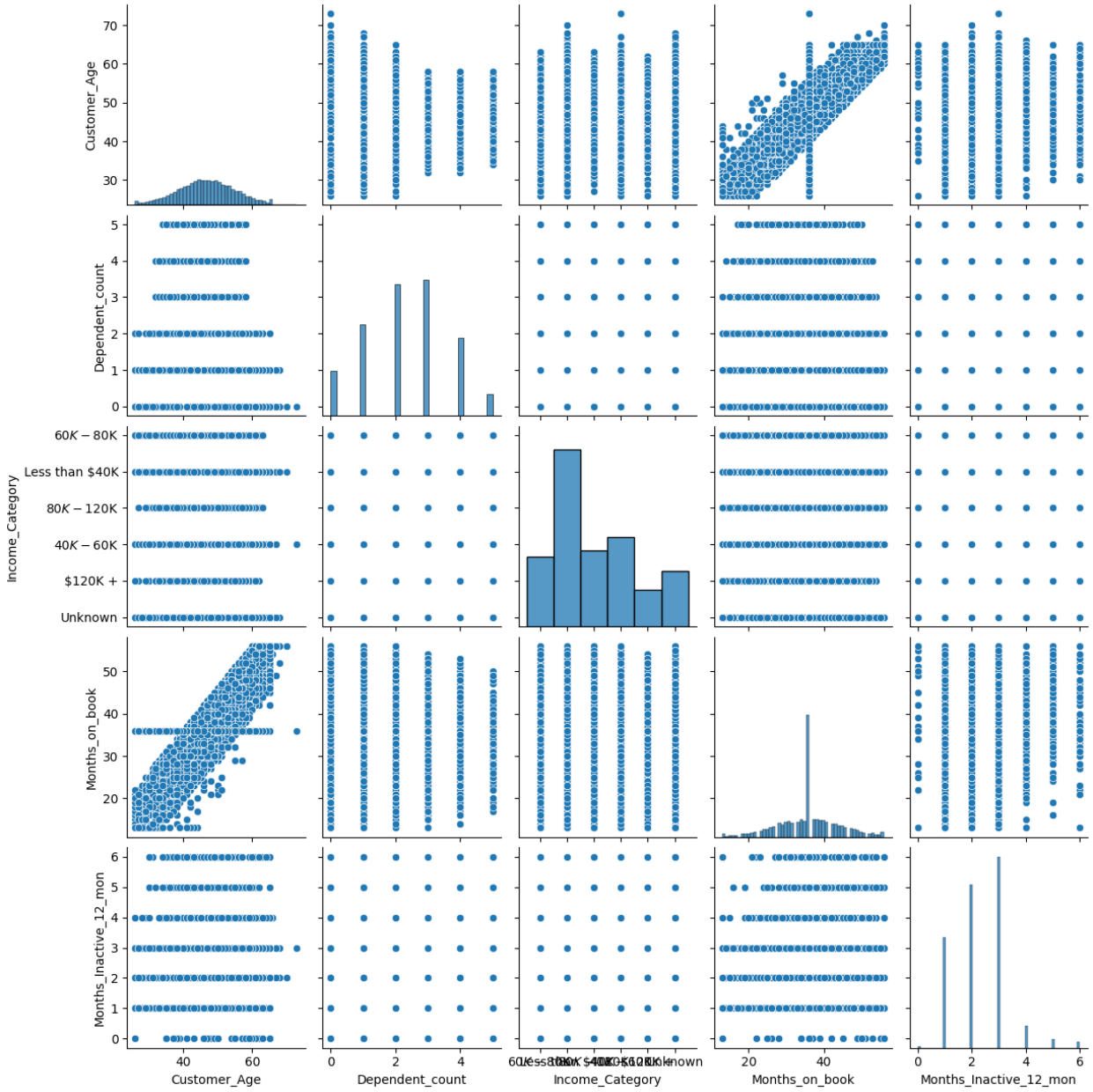
```
CLIENTNUM     Attrition_Flag  Customer_Age Gender  Dependent_count  \
0  768805383  Existing Customer            45      M                3
1  818770008  Existing Customer            49      F                5
2  713982108  Existing Customer            51      M                3
3  769911858  Existing Customer            40      F                4
4  709106358  Existing Customer            40      M                3

  Education_Level Marital_Status Income_Category Card_Category  \
0     High School        Married     $60K - $80K          Blue
1        Graduate         Single  Less than $40K          Blue
2        Graduate        Married    $80K - $120K          Blue
3     High School        Unknown  Less than $40K          Blue
```

```
4       Uneducated        Married      $60K - $80K          Blue
```

And if we print the whole data in python we can find out that it has got :

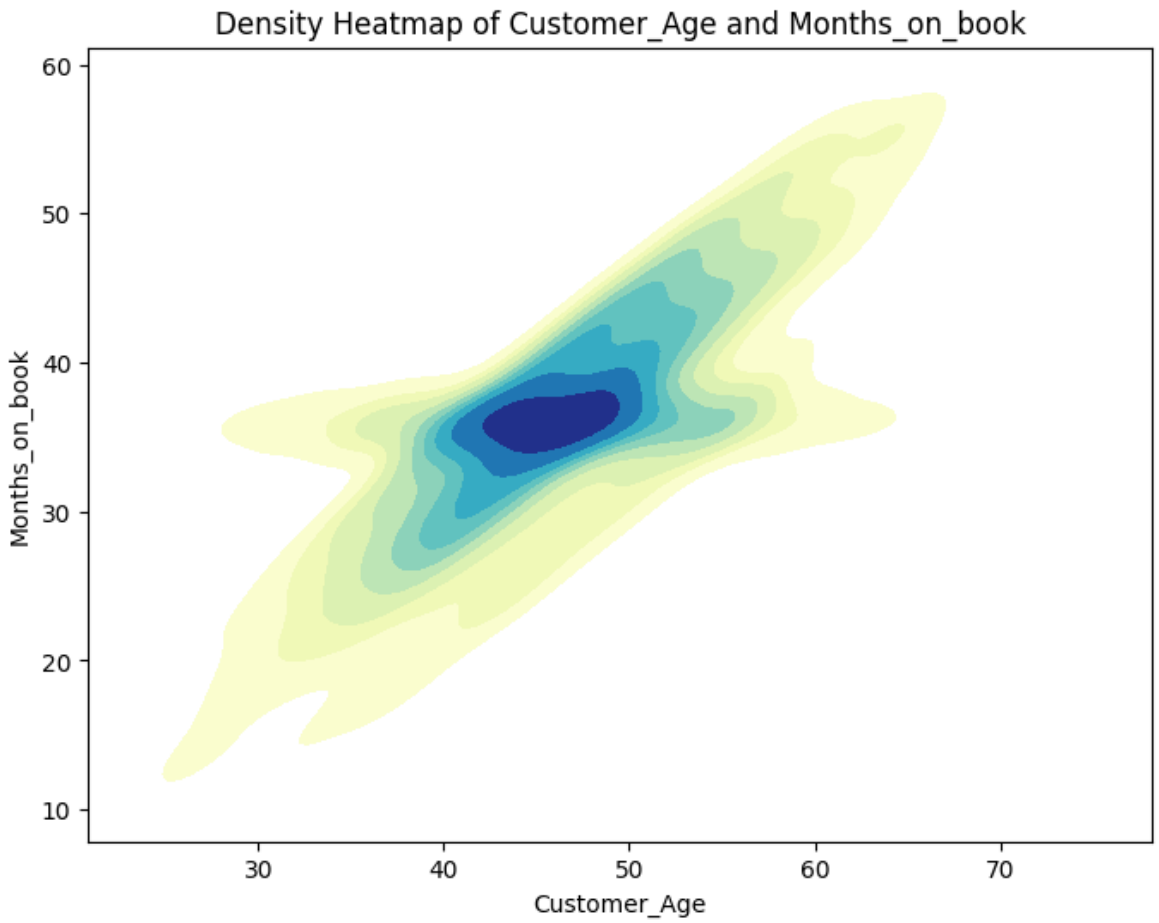
```
[10127 rows x 23 columns]
```

1.2: I've chosen 5 categories: `['Customer_Age', 'Dependent_count', 'Income_Category', 'Months_on_book', 'Months_Inactive_12_mon']`

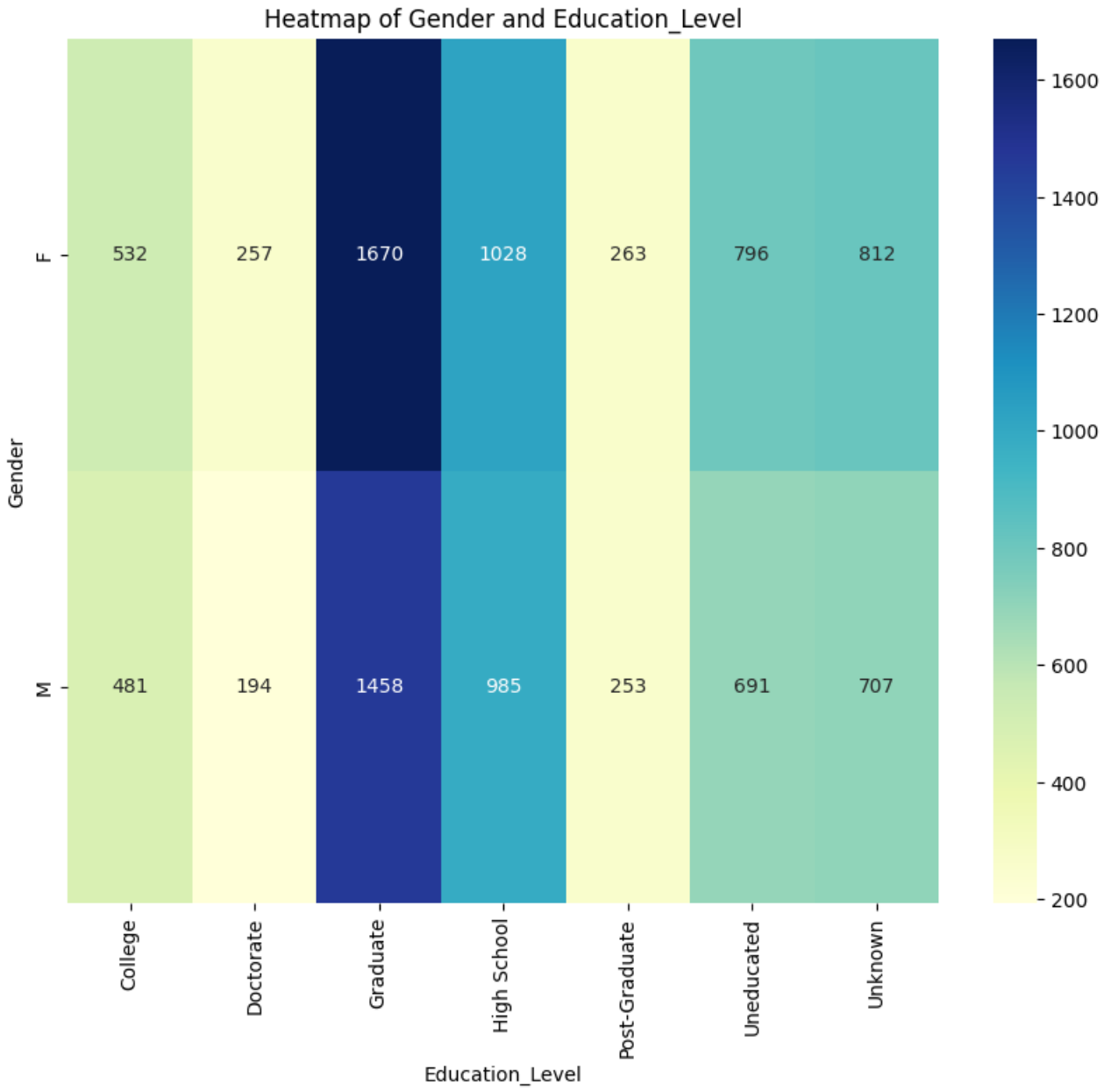


1.3 :

For two continuous features : `'Customer_Age'` and `'Months_on_book'`



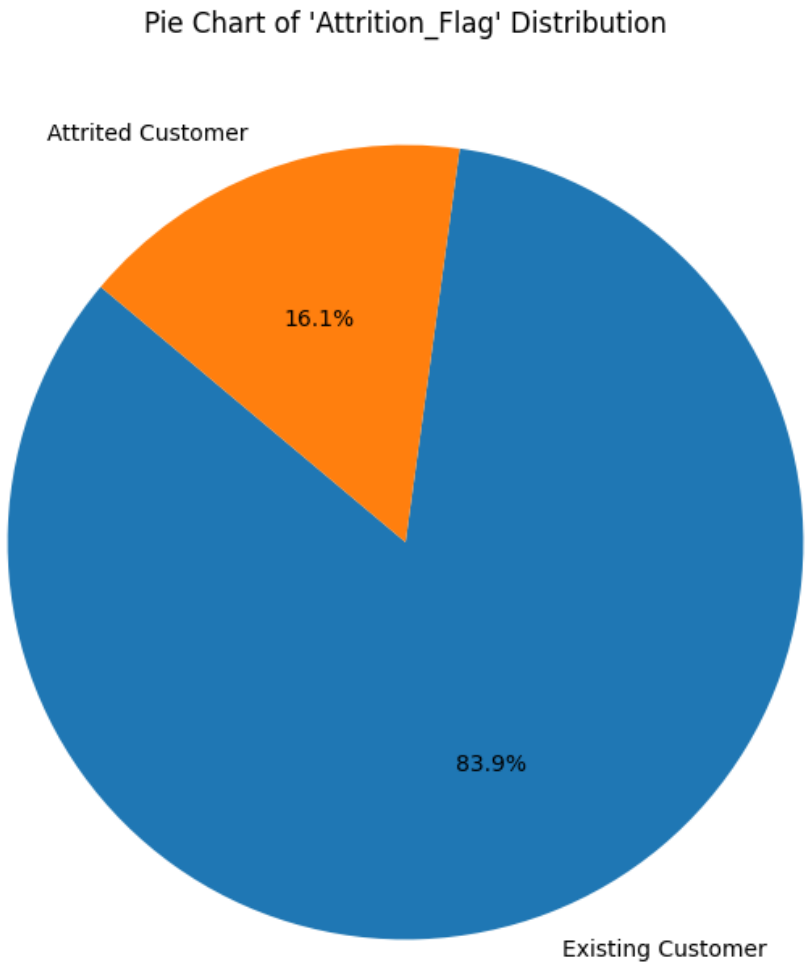
For two categorical features: `'Gender' and 'Education_Level'`



1.4 : It seems like we don’t have any NaN(Not a number) type data.

1.5:

```
Number of unique classes in 'Attrition_Flag': 2
Classes in 'Attrition_Flag': ['Existing Customer' 'Attrited Customer']
```



1.6 : Yes. When the data of a feature is unbalancing it can lead the final model to several problems:
1. model bias towards the majority class
2. poor performance on the minority class
3. reduced generalization
4. difficulty in learning minority class patterns

Ways we can handle imbalanced data :
1. resampling : balance the classes either ways: oversampling the minority or undersampling the majority.

2. class weights: we can set weighs for minority class, making the model more sensitive to it.

We have to balance it before learning cause it can lead us to face the problems we talked earlier and then we cannot correct them.

So as we mentioned earlier:

The imbalance between classes (327 vs. 1699) is impacting recall for the minority.

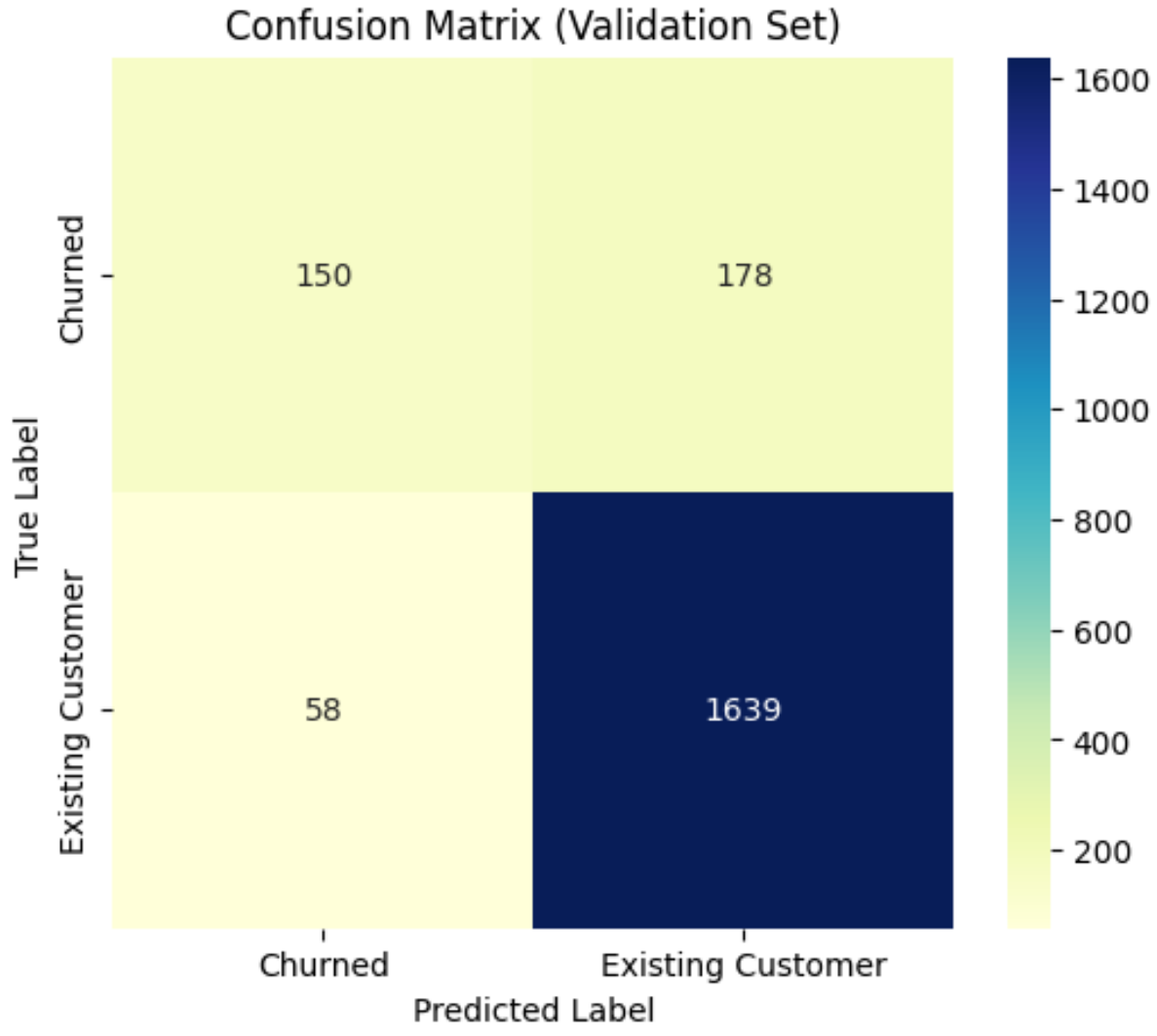
It is common with unbalanced data to make biased towards predicting the majority class.

So, I tried some ways to reduce this bias towards majority class bus SMOTE was the best one. Here are the results with SMOTE and without it:

```
Validation Accuracy: 0.8834567901234568
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.46      0.56       328
           1       0.90      0.97      0.93      1697

    accuracy                           0.88      2025
   macro avg       0.81      0.71      0.75      2025

weighted avg       0.87      0.88      0.87      2025
Test Accuracy: 0.8923988153998026
```

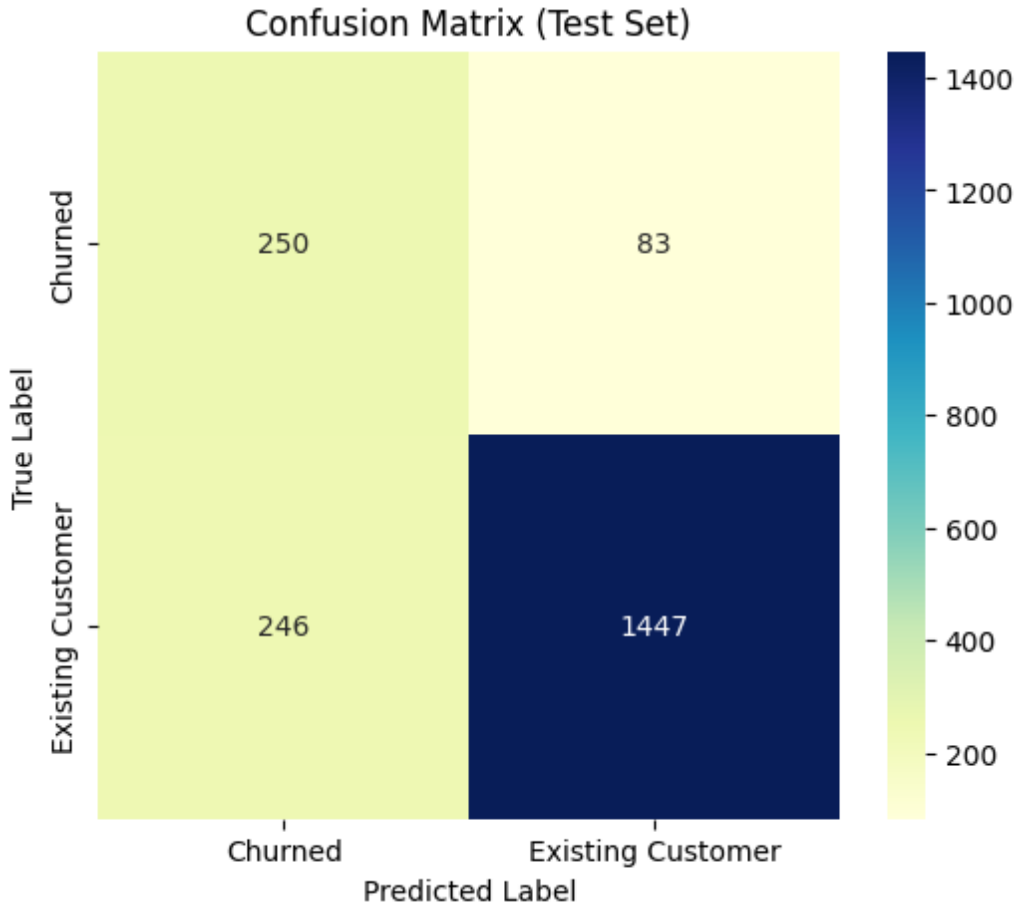
Test Classification Report:

	precision	recall	f1-score	support
0	0.78	0.48	0.59	333
1	0.90	0.97	0.94	1693
accuracy			0.89	2026
macro avg	0.84	0.73	0.77	2026
weighted avg	0.88	0.89	0.88	2026

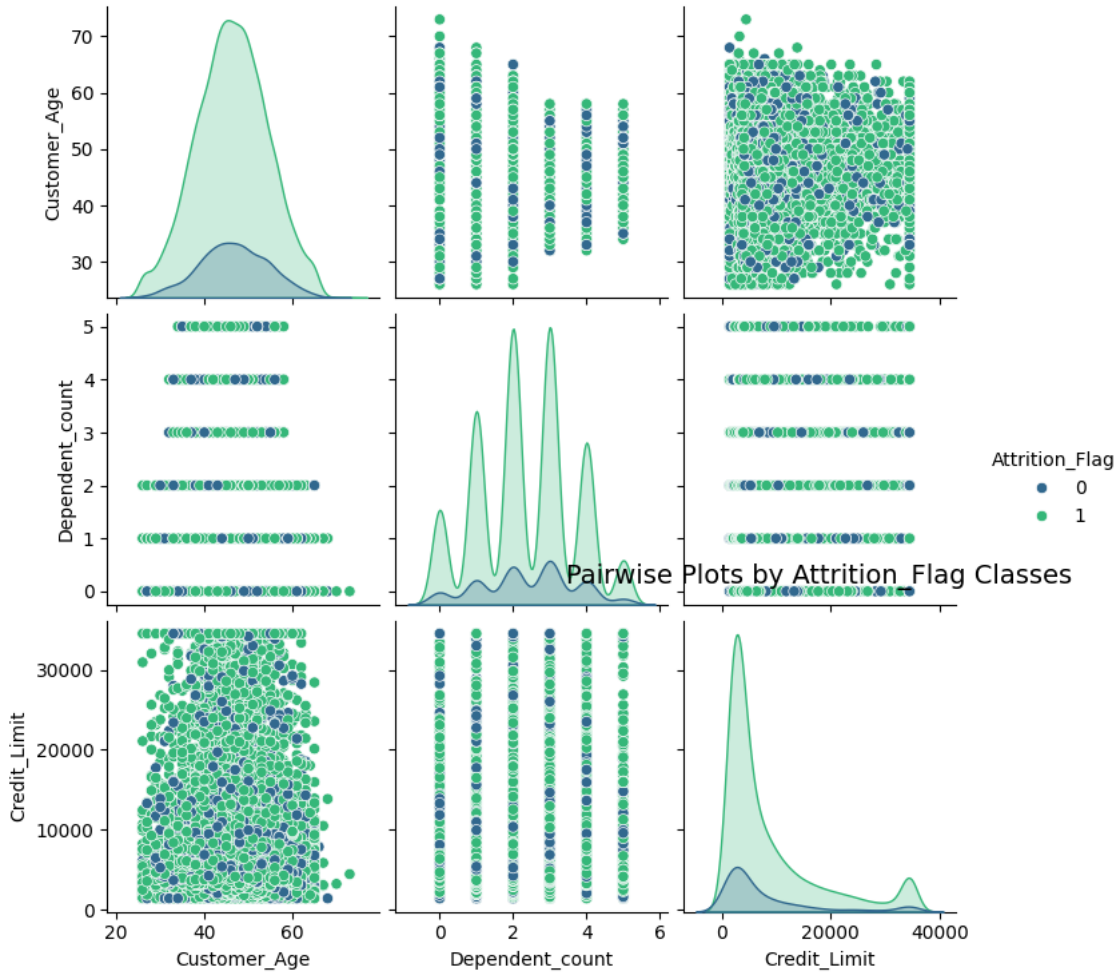


and with SMOTE and some optimizations:

```
Validation Accuracy with Adjusted Threshold: 0.8276543209876543
Validation Classification Report with Adjusted Threshold:
              precision    recall  f1-score   support

           0       0.48      0.74      0.58       328
           1       0.94      0.84      0.89      1697

    accuracy                           0.83      2025
   macro avg       0.71      0.79      0.74      2025
weighted avg       0.87      0.83      0.84      2025

Test Accuracy with Adjusted Threshold: 0.8376110562685094
Test Classification Report with Adjusted Threshold:
              precision    recall  f1-score   support

           0       0.50      0.75      0.60       333
           1       0.95      0.85      0.90      1693

    accuracy                           0.84      2026
   macro avg       0.72      0.80      0.75      2026
weighted avg       0.87      0.84      0.85      2026
```

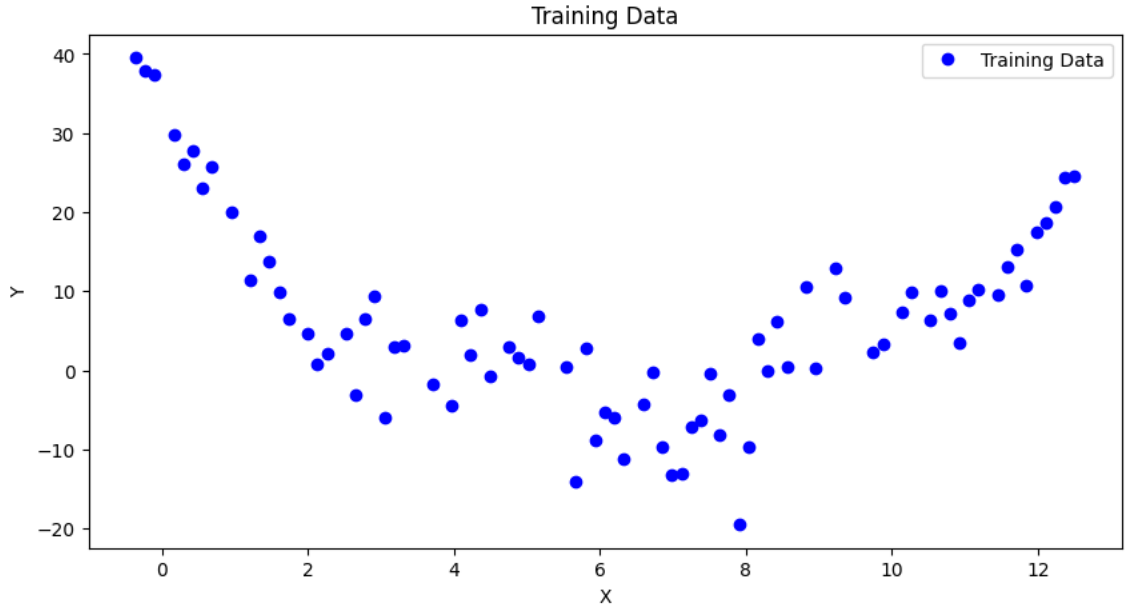


امتیازی سوال ۱:

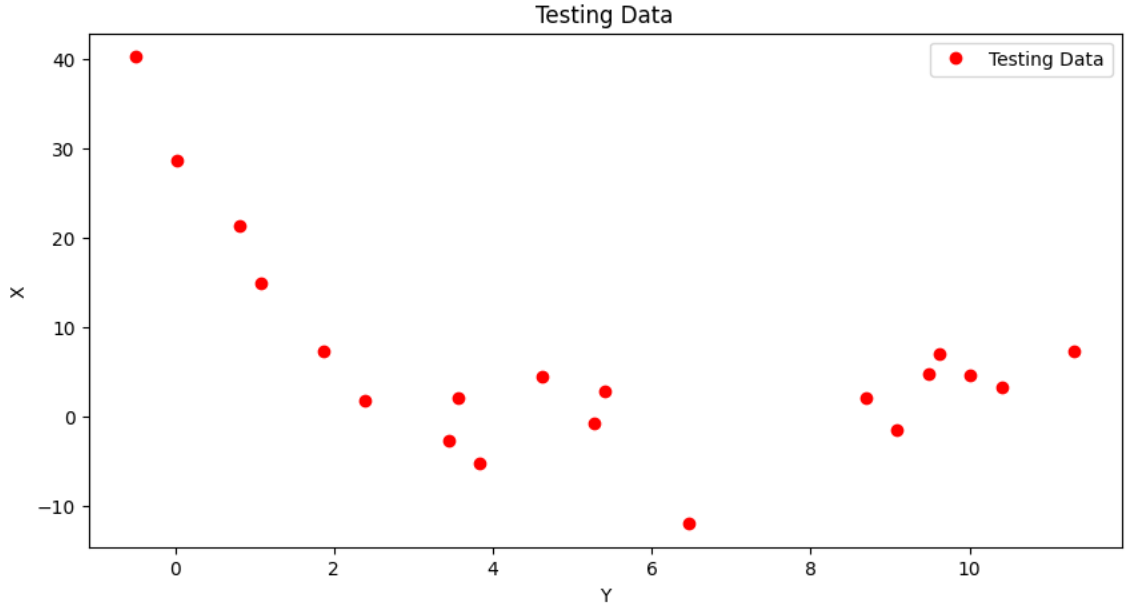


2.1:

Training data.



Testing data.



2.2

a) Mean Absolute Error (MAE):

- Measures the average absolute difference between predicted and actual values.

- Formula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

b) Mean Squared Error (MSE):

- Measures the average squared difference between predicted and actual values (penalizes larger errors more than MAE).

- Formula:

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

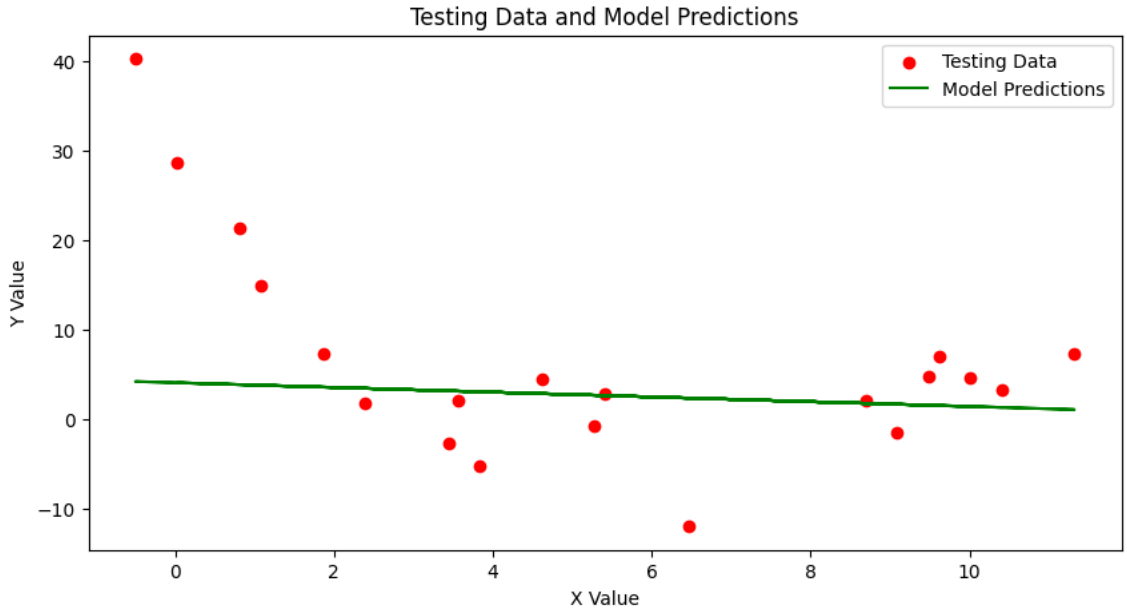
c) R-squared Score (R^2):

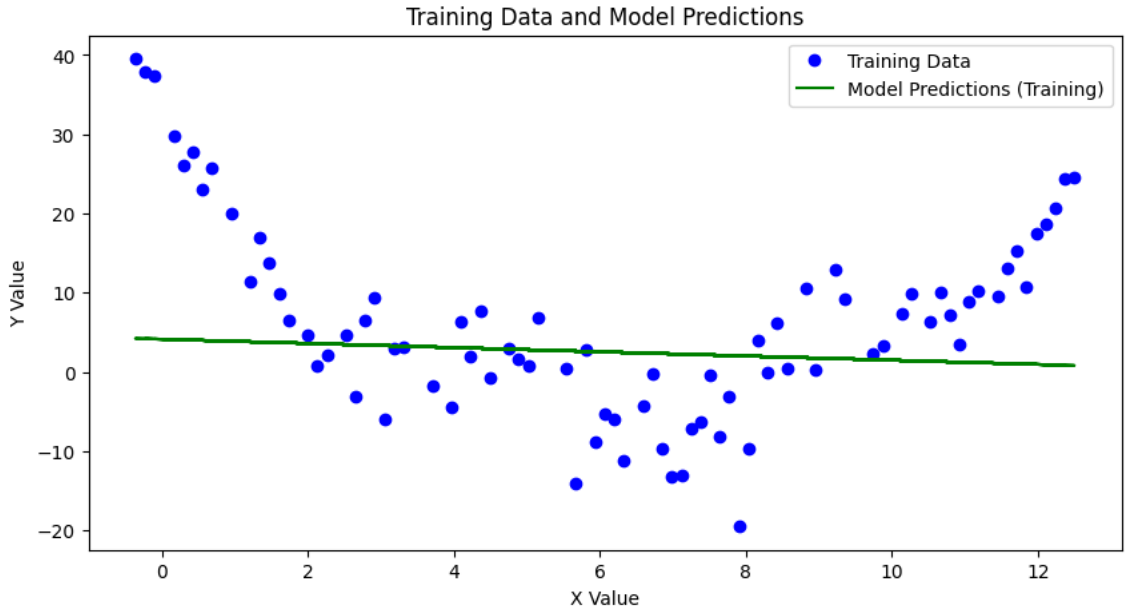
- Represents the proportion of variance in the dependent variable explained by the model.

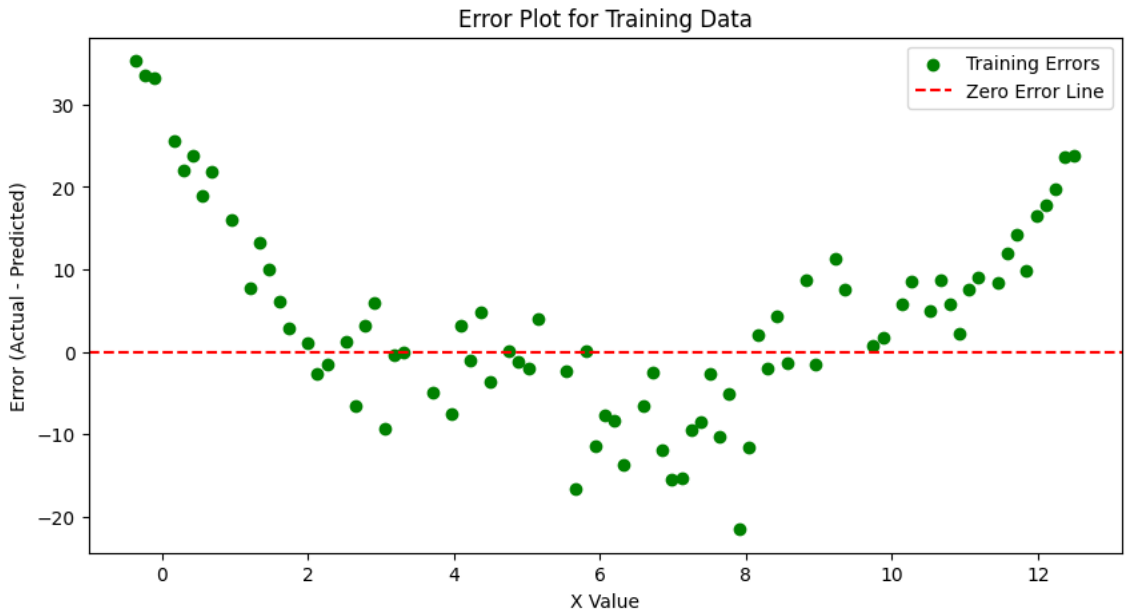
- Formula:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

2.3 No. it can't really fit the actual data. As we can see with the results :


Testing Data and Model Predictions
Testing Data
Model Predictions
Y Value
X Value
40
30
20
10
0
−10
0
2
4
6
8
10


Training Data and Model Predictions
Training Data
Model Predictions (Training)
Y Value
X Value
40
30
20
10
0
−10
−20
0
2
4
6
8
10
12




```
Mean Absolute Error (Testing): 7.625470122582359
Mean Squared Error (Testing): 138.3476415730294
R-squared (Testing): -0.027895749736724484
Mean Absolute Error (Training): 9.455722466135743
Mean Squared Error (Training): 157.5139517682842
R-squared (Training): -0.06916273117870686
```

The resons why it can't fit =>

1. Linear Assumption

Linear regression assumes that the relationship between the input (independent variable) and the output (dependent variable) is linear. In your graph:

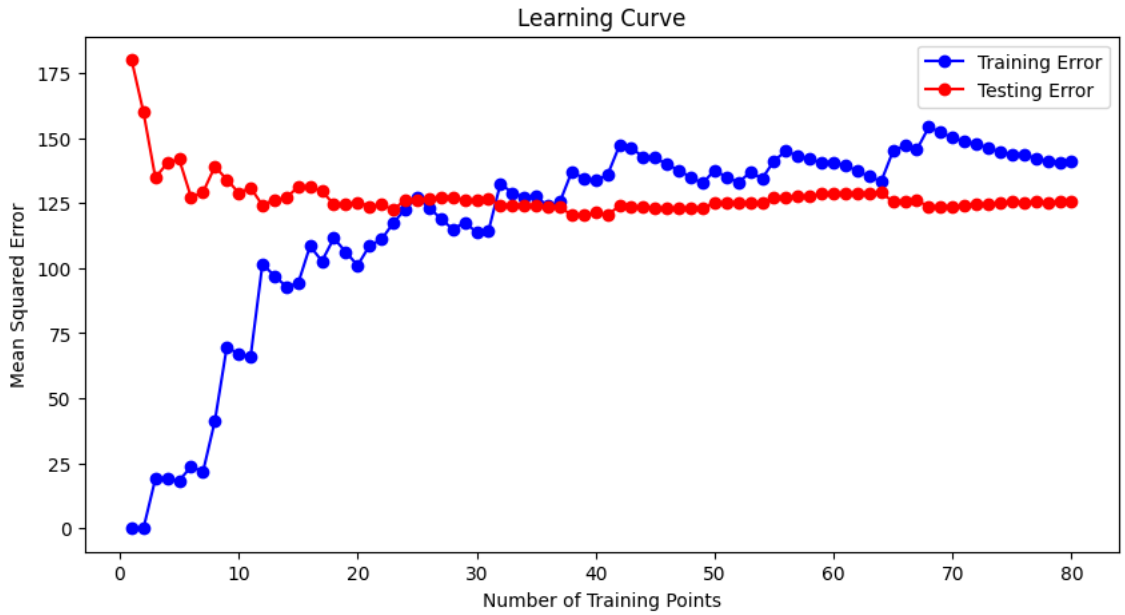
- The data clearly exhibits a **non-linear pattern**, resembling a curve or a sinusoidal wave.
- A linear model, represented by a straight line, cannot capture this curvature or oscillation.

2. Lack of Flexibility in Linear Models

Linear regression uses a fixed functional form ($y = mx + b$, or a hyperplane in multiple dimensions). This makes it unsuitable for modeling complex, non-linear relationships like:

- Polynomial patterns.
- Sinusoidal trends.
- Exponential growth or decay.

2.4:



as we can see, testing error decreased but then continue on a same level as we increase the number of training points. On the other hand the training error is oscillating. First it is at a low level and then it goes up and waves.

2.5:

1.**Intrinsic Limitations of the Model**: The model's error of 10 indicates that it may not be well-aligned with the underlying task. Simply adding more data may help the model learn more patterns or generalize better, but if the model's structure or capacity is not sufficient to capture the complexities of the task, its error will remain high. This is often the case if the model is too simple or inappropriate for the problem.

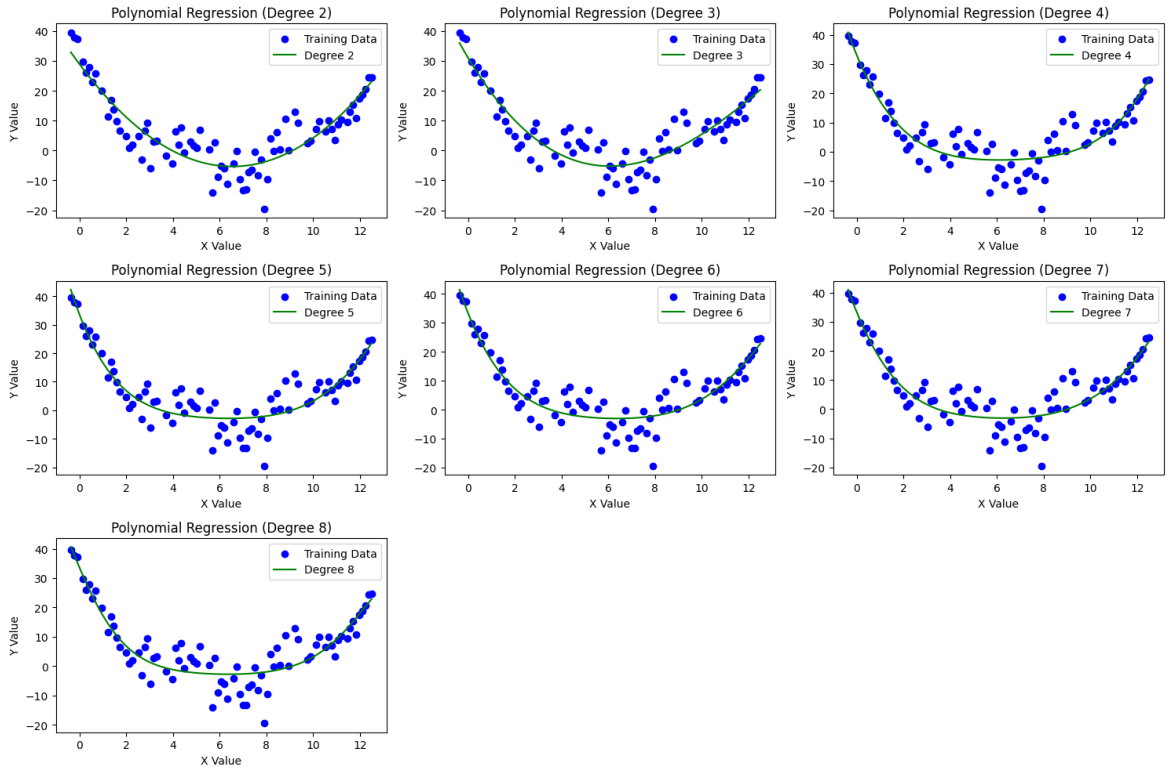
2.**Data Quality and Relevance**: More data can improve model performance, but the data must be relevant and diverse enough to cover various scenarios the model will encounter in practice. If the data is noisy or not representative of the task, simply increasing its quantity might not lead to better performance.

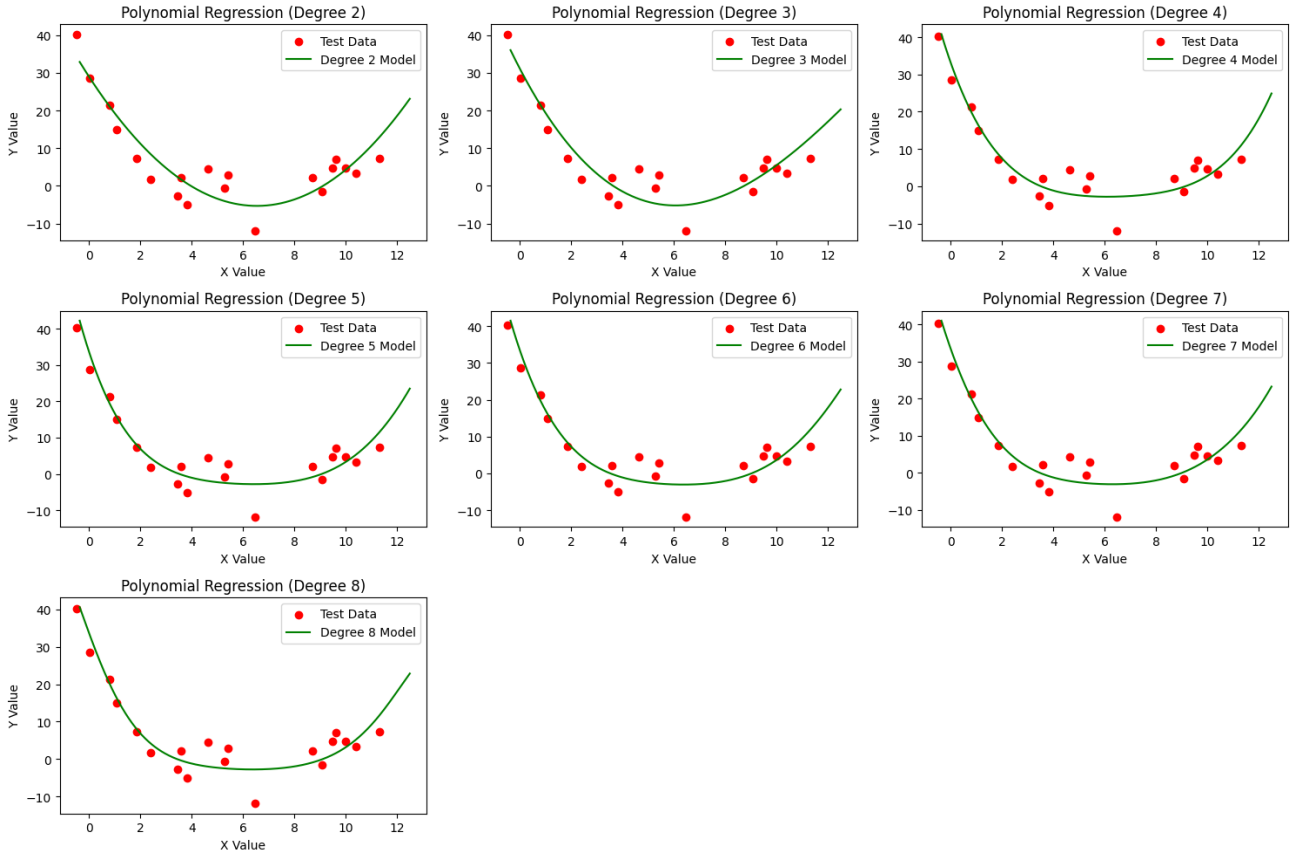
3.**Model Overfitting or Underfitting**: Adding more data might help if the model is underfitting, as it can learn more patterns from the additional data. However, if the model is overfitting (i.e., it has already memorized the training data), adding more data could help generalize better, but it may still struggle to reach the human level error.

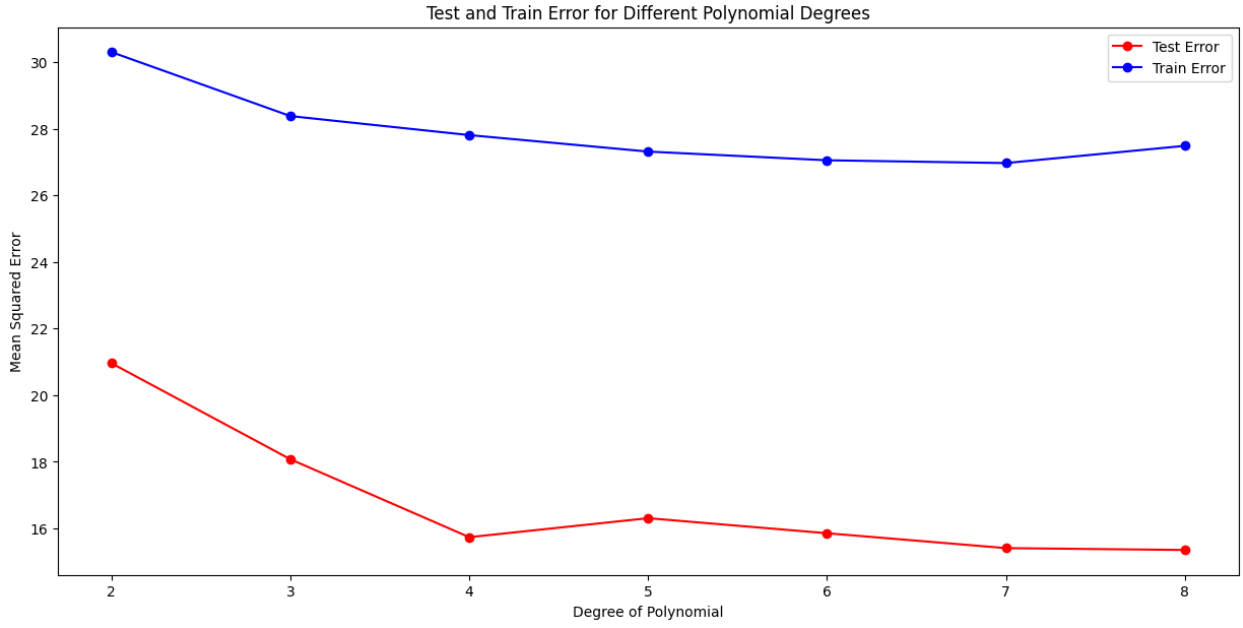
4.**Human Error vs. Machine Error**: The human error of 1 represents the performance of a human performing the task under normal conditions. The difference between the model's error (10) and human error (1) may also reflect the fundamental limitations of the task itself, which humans may handle with experience, intuition, or context that a machine cannot easily replicate.

In conclusion, while more data can improve the model's performance, it will not necessarily reduce its error to the human level of 1 unless the model has the capacity and structure to understand the task as well as a human does. The model may eventually get closer to the human error with enough data and appropriate adjustments, but there are no guarantees.

2.6:







as we see the error is increasing from 4th degree to …
because the original data doesn't have 5th or 6th degree and the model tries to fit with complex equations which lead to more error.

2.7

1. Decision Tree Regression

- **Description**: Uses a tree-like structure to make predictions by splitting data into regions based on feature values.
- **Usage**: Works well for non-linear relationships and datasets with complex patterns.

2. K-Nearest Neighbors Regression (KNN Regression)

- **Description**: Predicts the target value by averaging the values of the nearest k neighbors.
- **Usage**: Useful for non-linear and local patterns in data; performance depends on the choice of k and distance metric.
- **Parameter**: n_neighbors controls how many neighbors to consider.

3. Random Forest Regression

- **Description**: An ensemble method that builds multiple decision trees and averages their outputs to improve accuracy and reduce overfitting.
- **Usage**: Suitable for non-linear relationships and larger datasets.

The results:

Decision Tree Regression :

```
MAE: 4.603295760625363
MSE: 31.870745820568185
R^2 Score: 0.763206630793134
```

K-Nearest Neighbors Regression :

```
MAE: 2.8458251070693126
MSE: 14.843034901860367
R^2 Score: 0.88971917182439
```

Random Forest Regression:

```
MAE: 3.5122833776238473
MSE: 18.002214678954907
R^2 Score: 0.866247087814808
```