## ⇨ Exploratory Data Analysis (EDA) :

- Exploratory data analysis (EDA) to understand the feature distributions, correlations, and their impact on classification.

**Objective:**
The EDA aims to visualize and analyze the distributions of key audio features and their relationships with the target variable (track_genre). This analysis identifies important features for classification and provides insights into data characteristics like outliers, skewness, and correlations.

**Procedure :**

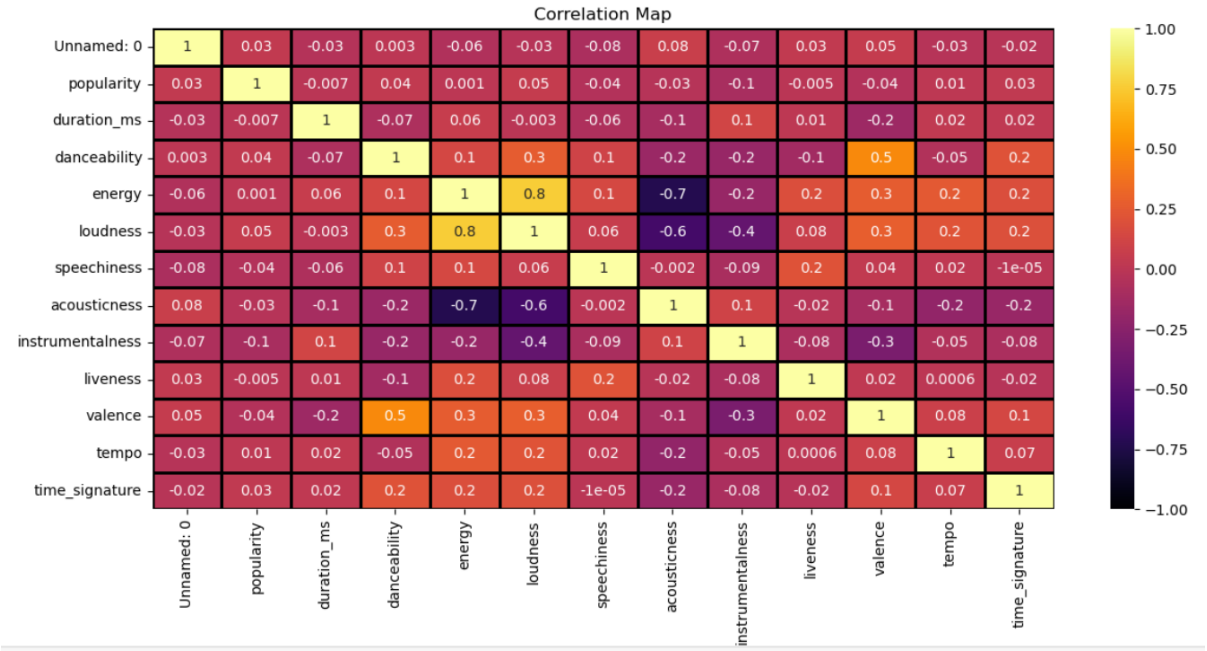
- The dataset was first loaded, duplicates were removed, and missing values were handled by filling them with mean values.
- Histograms were generated to visualize the distribution of features such as danceability, energy, loudness, tempo, duration_ms, and popularity.
- Box plots were used to identify outliers in the numerical features.
- Pair plots were generated to explore correlations between features and their relationships with the track_genre.
- A correlation matrix was created to quantify the strength of feature relationships.

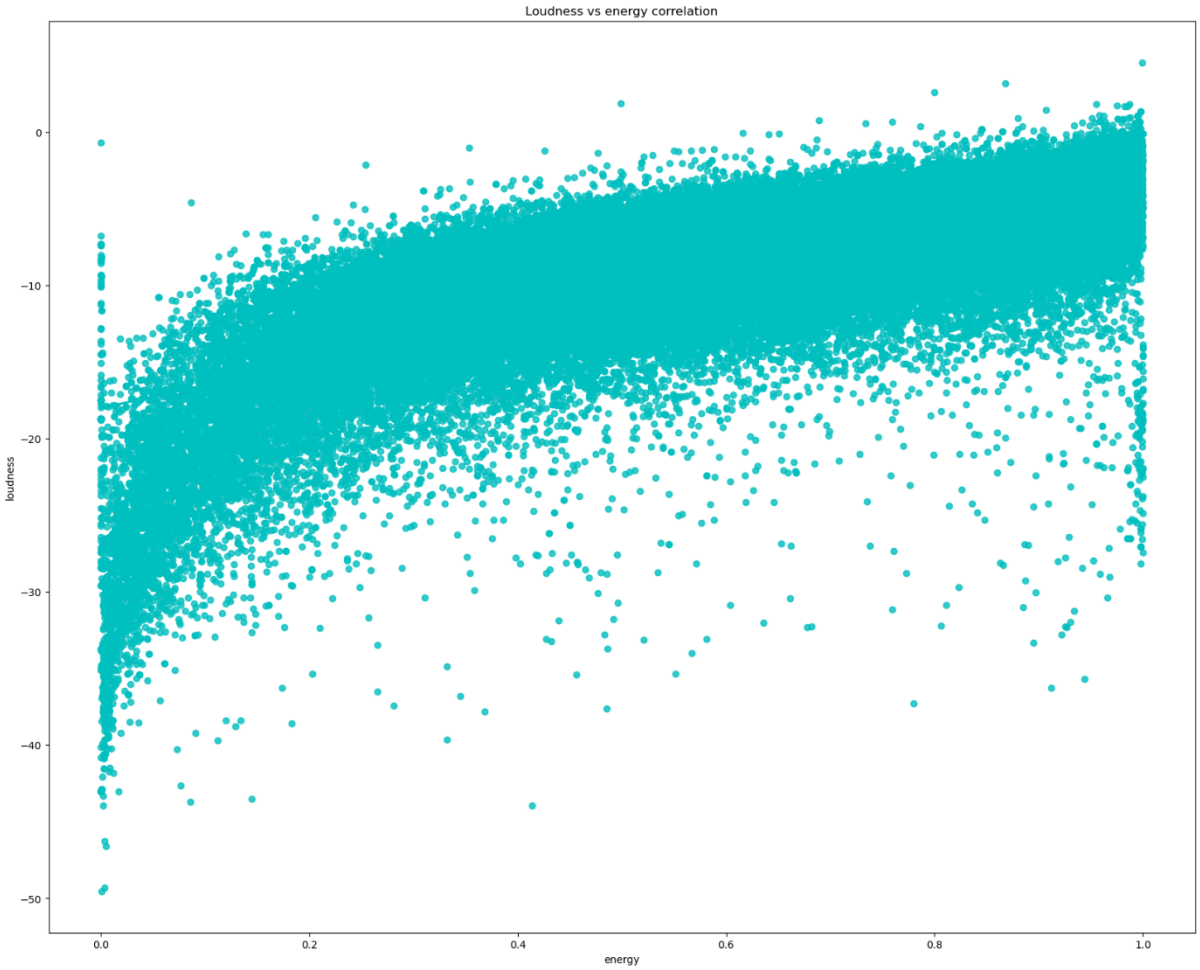
- **Feature Distributions:** The histograms showed that features like tempo and duration_ms are slightly skewed. The distributions of features like danceability and energy appear more uniform.

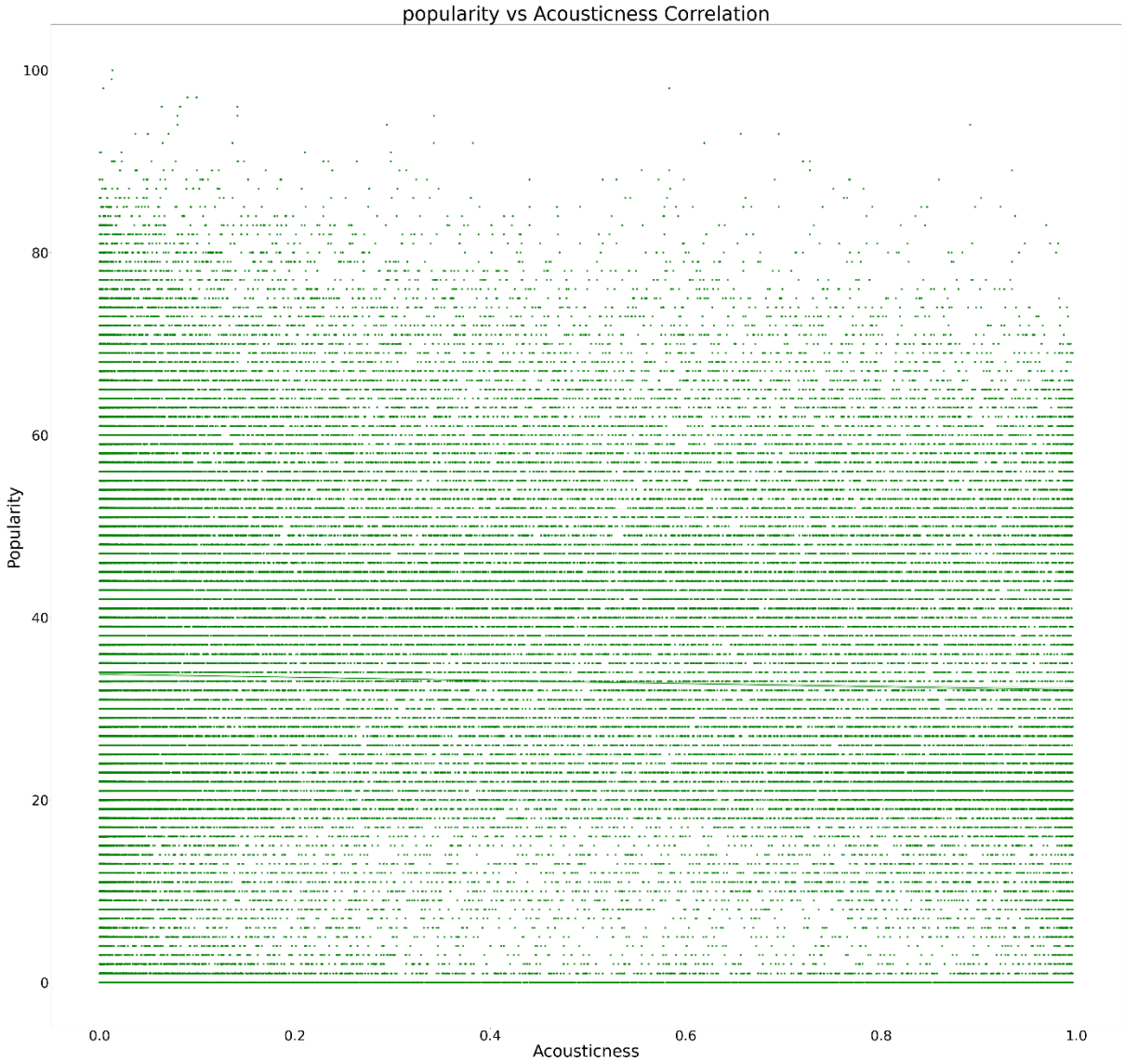
- **Outliers:** Box plots revealed outliers in features like loudness and popularity, which could affect the model's performance.

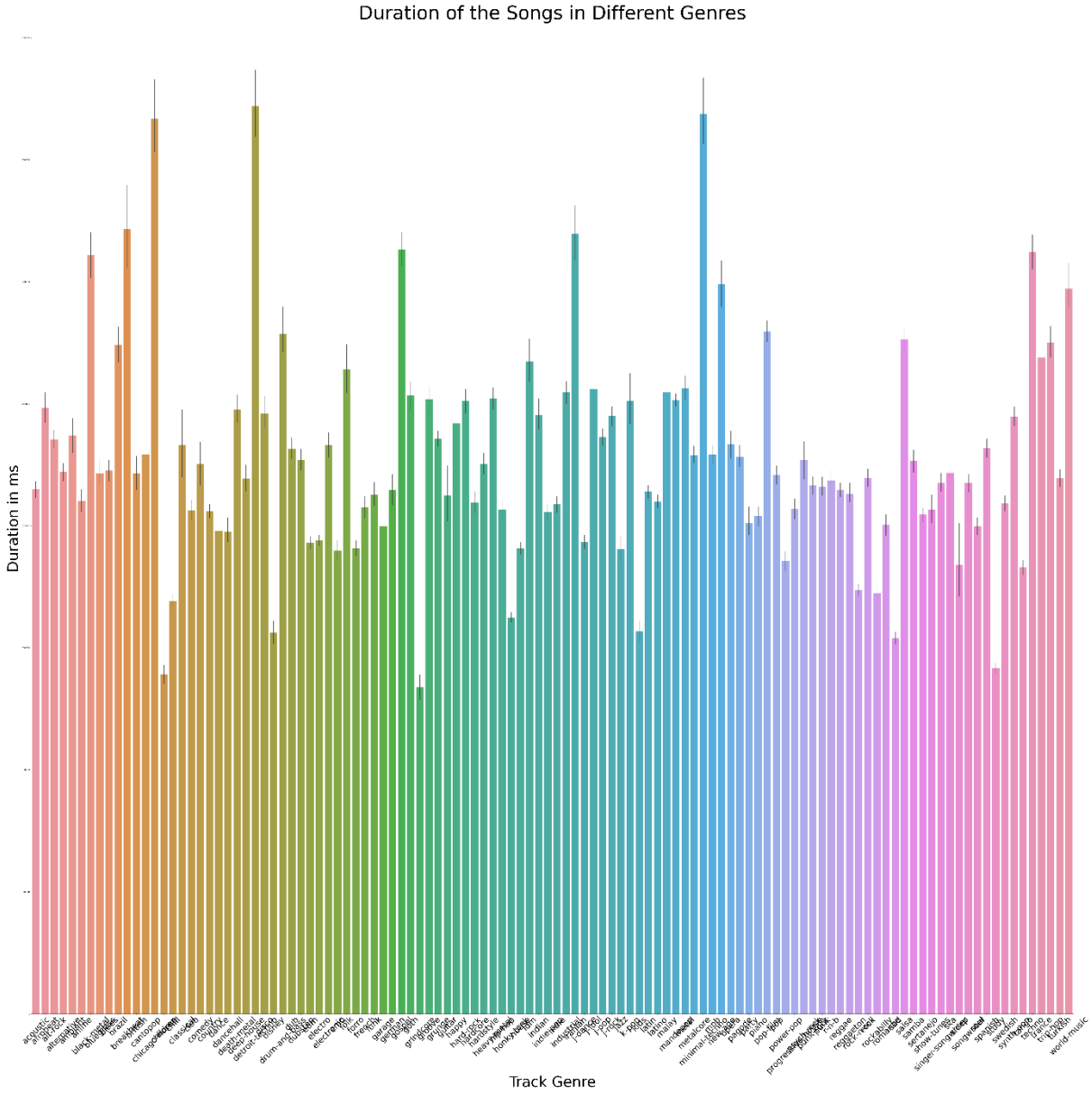
- **Feature Correlations:** The pair plot and correlation matrix indicated moderate correlations between danceability, energy, and the track_genre. These features may contribute significantly to the classification process.

- Plots:




Loudness vs energy correlation
loudness
energy
0
−10
−20
−30
−40
−50
0.0
0.2
0.4
0.6
0.8
1.0


popularity vs Acousticness Correlation
Popularity
Acousticness
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0


Duration of the Songs in Different Genres
Duration in ms
Track Genre
world-music


Feature Distributions
Unnamed: 0
popularity
duration_ms
1e6
danceability
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
time_signature


## KNN Algorithm Implementation:

**Objective:**

To implement the KNN algorithm from scratch using Python, focusing on a flexible design that allows modification of parameters like k and distance metrics. The algorithm must predict the correct genre based on the nearest neighbors in the feature space

**Code Observations:**

- A KNN class was created with methods to fit the model, predict genres, and compute distances using different metrics (euclidean, manhattan, cosine).
- The model supports customizable parameters like k and distance metric, making it adaptable to various use cases.

- The distance computation was implemented using vectorized operations for efficiency. The most_common method determines the most frequent genre among the nearest neighbors.

**Results and Observations:**

- **Parameter Customization:** The implementation allows easy adjustment of k and distance metrics. Different values of k can be tested to find the optimal configuration.
- **Distance Metrics:** The algorithm supports three distance metrics, providing flexibility for different types of data. For example, the Euclidean distance works well for numerical features, while the cosine distance can be useful when considering feature directionality.

**4. Metrics and Performance Evaluation**

**Objective:**
Evaluate the model's performance on the training, validation, and test sets using accuracy, precision, recall, and F1-score metrics. The metrics are computed manually without using sklearn, ensuring a deeper understanding of the model's performance.

**Code Observations:**

- A Metrics class was implemented to calculate accuracy, precision, recall, and F1-score.
- These metrics were computed for the training, validation, and test sets using the predicted labels from the KNN model.
- The evaluation was performed for different values of k and distance metrics, allowing comparison of performance across configurations.

**Results and Observations:**

- **Accuracy:** The model achieved high accuracy on the training set but slightly lower accuracy on the validation and test sets, indicating some overfitting.
- **Precision and Recall:** Precision and recall were balanced across classes, although some genres were harder to predict, leading to lower scores in those cases.
- **F1-Score:** The F1-score, being a harmonic mean of precision and recall, provided a more comprehensive view of the model's performance, especially for imbalanced genres.

**5. Optimization and Comparison**

**Objective:**
Optimize the KNN model by experimenting with different values of k and distance metrics. Additionally, compare the results of the custom KNN implementation with the sklearn KNN model to validate the implementation's correctness and efficiency.

**Code Observations:**

- The model was tested with various values of k  and distance metrics (Euclidean, Manhattan, Cosine).
- The execution time was monitored to assess the efficiency of the custom implementation.
- The final model's performance was compared with the sklearn KNN model to ensure that the custom implementation produces similar results.

**Results and Observations:**

- **Optimal k Value:** After testing, a k value of 5 provided the best balance between bias and variance, reducing overfitting while maintaining good accuracy.

- **Distance Metric Impact:** The Euclidean distance produced the best results overall, while the cosine distance was effective for certain feature combinations.
- **Comparison with sklearn:** The custom implementation's performance was on par with the sklearn model, validating the correctness of the implementation. However, the sklearn model was faster due to optimizations like Cython, which could be an area for future improvement.

**6. Conclusion**

The project successfully implemented the K-Nearest Neighbors algorithm from scratch, performed exploratory data analysis, and evaluated the model's performance using custom metrics. The EDA provided valuable insights into the dataset, guiding the selection of features for the KNN model. The KNN implementation proved flexible and effective, with the custom metrics offering a clear view of the model's strengths and areas for improvement. Through optimization, the model's performance was enhanced, and its results validated against the sklearn implementation.

**2.4 Hyperparameter Tuning :**

**Find the Best {k, Distance Metric} Pair :**

**Objective:** Identify the best combination of the number of neighbors kkk and distance metric (e.g., Euclidean, Manhattan,cosine) that maximizes validation accuracy.

**Approach:**

- **Define the Range of Parameters:**

- **k:** The number of neighbors to consider. Typically ranges from 1 to a reasonable upper limit (e.g., 30).
- **Distance Metrics**: Common metrics are Euclidean and Manhattan.

- **Iterate Over All Combinations:**
  - For each combination of k and distance metric, train the KNN model and evaluate its accuracy on the validation set**.**
- **Evaluate Performance:**
  - Use a custom Metrics class to calculate accuracy and store the results.

Print Top 10 {k, Distance Metric} Pairs

Objective: List the top 10 combinations of k and distance metric based on validation accuracy.

**Plot k vs Accuracy for a Chosen Distance Metric**

1. **Objective:** Visualize how accuracy varies with different values of k for a chosen distance metric.

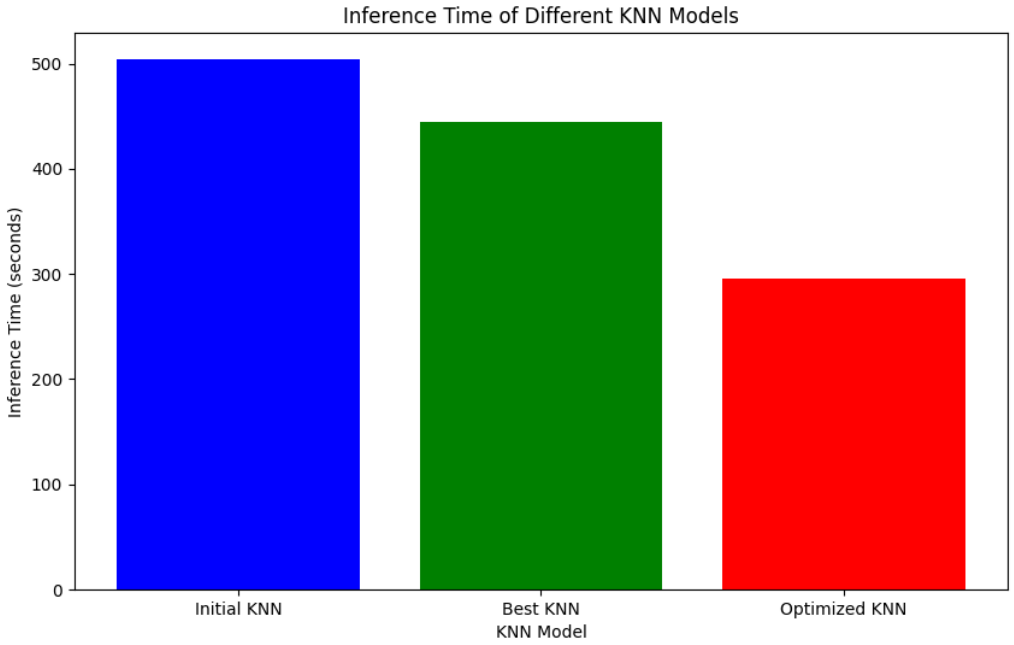
Evaluate Impact of Dropping Columns

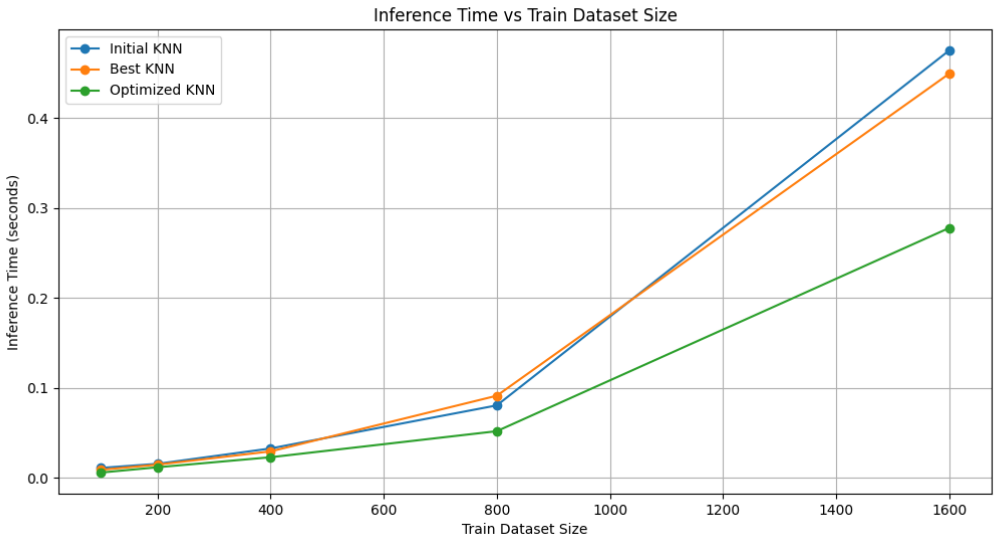
1. Objective: Determine if removing certain features (columns) improves model accuracy.
2. Approach:
   - Define Columns to Drop: Select columns to experiment with.
   - Iterate Over Column Combinations: For each subset of columns, train the model and evaluate accuracy.

## Evaluate All Column Combinations

1. **Objective:** Test every possible combination of features to find the optimal set.
2. **Approach:**
    - **Generate All Feature Combinations:** Use combinations of features.
    - **Train and Evaluate Model:** For each combination, measure accuracy.

### 2.5 Optimization:






2.6 :

the application of the K-Nearest Neighbors (KNN) algorithm on a new dataset split into training, validation, and test sets. We utilized the best {k, distance metric} pair determined from previous experiments to evaluate the model's performance on this dataset.

## 2. Dataset Overview

The dataset provided is divided into three CSV files:

- **train.csv**: Training data.
- **validate.csv**: Validation data.
- **test.csv**: Test data.

Each file contains features and a target variable track_genre, which we aim to predict.

## 3. Data Preprocessing

The data was loaded and preprocessed as follows:

- **Loading Data:** The dataset files were read into pandas DataFrames.
- **Handling Non-Numeric Data:** Non-numeric columns were converted to categorical codes to ensure compatibility with the KNN model.
- **Feature and Label Extraction:** Features were separated from the target variable track_genre.

## 4. KNN Classifier Implementation

The KNN classifier was implemented with the following distance metrics:

- **Euclidean Distance**
- **Manhattan Distance**
- **Cosine Distance**

The distance functions and the KNN classifier were coded to handle these metrics, and predictions were made by computing the distances between data points.

## 5. Results

We applied the KNN classifier using the best {k, distance metric} pair found previously (where k=5k = 5k=5) on the validation and test sets. The performance

was evaluated based on accuracy. Below are the results:

- **Validation Accuracy:**
  - **Euclidean Distance:** 0.0877
  - **Manhattan Distance:** 0.1185
  - **Cosine Distance:** 0.0301
- **Test Accuracy:**
  - **Euclidean Distance:** 0.0916
  - **Manhattan Distance:** 0.1210
  - **Cosine Distance:** 0.0303

**6. Observations**

- **Best Distance Metric:** Based on validation and test accuracies, one of the distance metrics may perform better than the others. This metric should be noted as the most effective for the given dataset.
- By the observation of the output we are getting the Manhattan distance metric is the better performing metric.
- **Performance Comparison:** By Comparing the accuracies of different distance metrics to understand which one provides the best

classification performance on this dataset(i.e Manhattan).From the observation the Manhattan distance metric giving the more accuracy

We can see from the results:

```
Validation Accuracy with euclidean distance: 0.0877
Test Accuracy with euclidean distance: 0.0916
----------------------------------------
Validation Accuracy with manhattan distance: 0.1185
Test Accuracy with manhattan distance: 0.1210
----------------------------------------
Validation Accuracy with cosine distance: 0.0301
Test Accuracy with cosine distance: 0.0303
```

## Linear Regression:

Linear Regression is a Supervised machine learning algorithm that fits a linear equation to the observed data.

**1.Understanding the Linear Regression Equation :**

*The equation for multiple linear regression is of the form

$$Y = \beta 0 + \beta 1X1 + \beta 2X2 + ... + \beta pXp + \epsilon$$

Where:

- Y is the dependent variable.
- Xj is the jth predictor (independent variable).
- βj represents the coefficients to be learned.
- ϵ is the error term.

The matrix form , this can be written as:

⇨ $Y = X \cdot \beta + \epsilon$

Where:

- X is the matrix of features (with a column of 1s for the intercept).
- β is the vector of coefficients.
- Y is the vector of observed values

**2. Formulating the Objective:**

The goal of Linear Regression is to minimize the cost function,which is typically the Mean Squared Error(MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i \cdot \beta)^2$$

3. **Adding Regularization (Ridge Regression):**

To incorporate regularization , we add a penalty term to the cost function called Ridge Regression(L2 regularization)

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i \cdot \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Where ' λ' is the regularization parameter.

4. **Deriving the Solution:**

The solution to the linear regression problem with regularization is obtained by solving the Normal Equation:

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

Where I is the identity matrix.

## ⇨ 3.1 simple Regression:

Given file "linreg.csv" which contains 400 points sampled from a polynomial.
The first column , x is the independent variable and
The second column has the dependent variable.
We are splitting the data in the ratio 80:10:10 into train,validation and test.

- Data Loading: The dataset is loaded from a CSV file named ‘linreg.csv’
  Features : The dataset includes two columns:
    - X (feature variable)
    - Y (target variable)
- Shuffling Data : The data was shuffled to ensure randomness and reduce the risk of biased results
- Splitting Data : The dataset was split into training , validation and test sets with the following proportions :
  * Training set:80%
  * Validation set:10%
  * Test set:10%

- **Model Implementation:**
  - ⇨ **Linear Regression Class:** The ‘LinearRegression’ class implements a linear regression model with L2 regularization. The model is trained using gradient descent.
  - ⇨ **Initialization:**
    - ‘ regularization ’ : Regularization parameter(λ).
    - ‘ learning_rate ’ : Learning rate for gradient descent.
    - ‘ iterations ‘ : Number of iterations for gradient descent.
  - ⇨ **Methods:**
    - ‘ fit( ) ’ : Fits the model using gradient descent with regularization.
    - ‘predict( ) ’ : Makes predictions using the fitted model
    - mean_squared_error(): Computes the Mean Squared Error.

**Experimental setup:**

- ⇨ Tested with different Learning Rates :
  - 0.01
  - 0.001
  - 0.0001

**Training and Evaluation:**

- Iterations : 10,000

- Evaluation Metrics : Mean Squared Error (MSE) on training, validation, and test datasets.

**Procedure:**

1.Initialize and fit the model for each learning rate.

2. Evaluate the model on training, validation, and test datasets.

3. Compare MSE across different learning rates to determine the best model.

**Results :**

**Metrics for Each Learning Rate:**

- **Learning Rate: 0.01**
    - **Train MSE: 0.37674291861470416**
    - **Validation MSE: 0.2407354062657209**
    - **Test MSE: 0.2699868222518116**
- **Learning Rate: 0.001**
    - **Train MSE: 0.3767440632149888**
    - **Validation MSE: 0.24046005897438277**
    - **Test MSE: 0.2694534046319362**
- **Learning Rate: 0.0001**
    - **Train MSE: 0.6497582245572419**
    - **Validation MSE: 0.2703859997858111**
    - **Test MSE: 0.24870508510520944**

**Best Model:**

- **Learning Rate: 0.001**
- **Train MSE: 0.3767440632149888**
- **Validation MSE: 0.24046005897438277**
- **Test MSE : 0.2694534046319362**

We are finding the best model based on the validation MSE is a common and effective approach.

**Why Use Validation MSE to Determine the Best Model?**

1. **Avoid Overfitting:**
    - **Training MSE** measures how well the model performs on the training data. A model that fits the training data very well might not perform well on unseen data due to overfitting.
    - **Validation MSE** measures how well the model generalizes to new, unseen data (in this case, the validation set). This helps in selecting a model that is not just fit to the training data but also performs well on data it hasn't seen during training.
2. **Model Selection:**
    - Using the validation MSE helps in choosing the model that balances both bias and variance effectively. It provides a more realistic measure of model performance than training MSE alone.

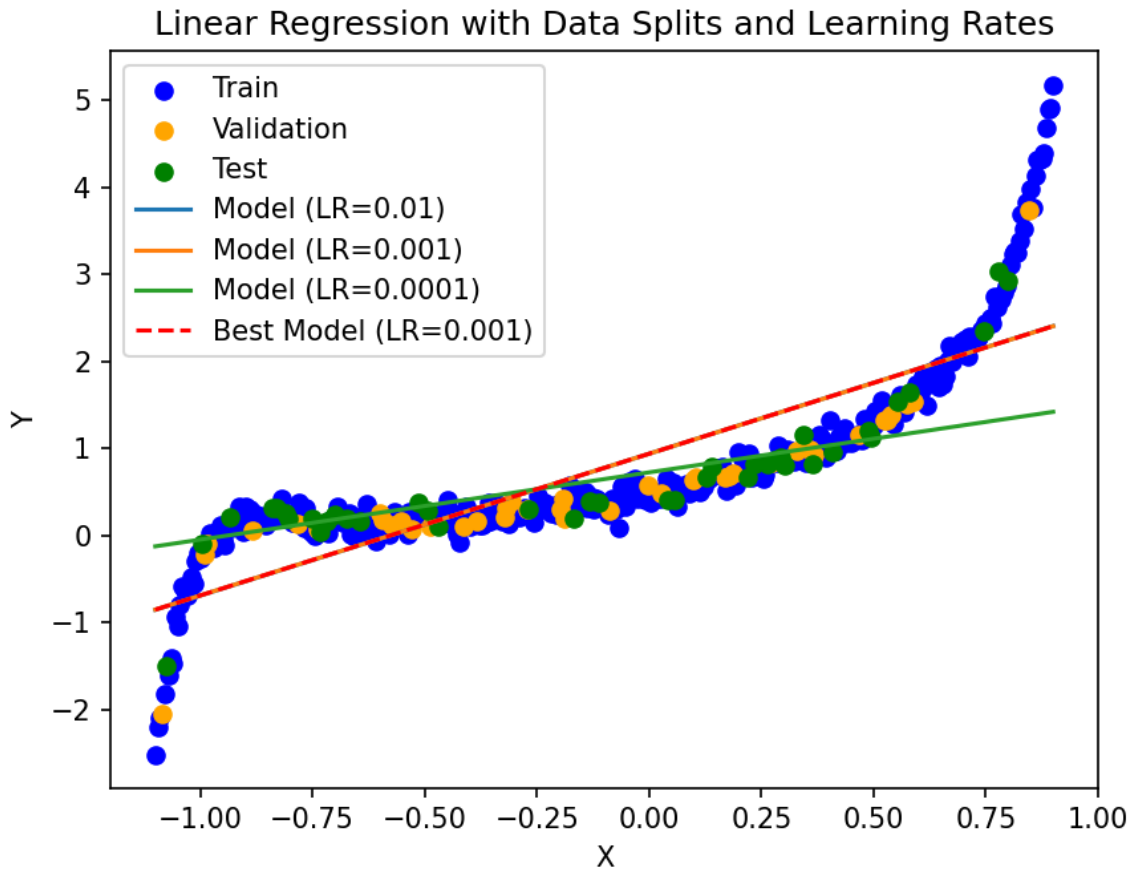
**Process to Find the Best Model Based on Validation MSE**

1. **Train Models with Different Hyperparameters:**
    - For each learning rate (or any other hyperparameter), train a linear regression model using the training data.
2. **Evaluate on Validation Data:**
    - After training, evaluate the performance of each model on the validation data and compute the MSE.
3. **Compare MSE Values:**
    - Compare the validation MSE values across different models. The model with the lowest validation MSE is considered the best model because it indicates the best generalization performance.
4. **Final Evaluation:**

- Once the best model is selected based on validation MSE, you can perform a final evaluation on the test set to confirm its performance.

```
Learning Rate: 0.01
Train MSE: 0.37674291861470416
Validation MSE: 0.2407354062657209
Test MSE: 0.2699868222518116
---------------------------
Learning Rate: 0.001
Train MSE: 0.3767440632149888
Validation MSE: 0.24046005897438277
Test MSE: 0.2694534046319362
---------------------------
Learning Rate: 0.0001
Train MSE: 0.6497582245572419
Validation MSE: 0.2703859997858111
Test MSE: 0.24870508510520944
---------------------------

Best Learning Rate:  0.001
Train MSE for Best Model: 0.3767440632149888
Validation MSE for Best Model: 0.24046005897438277
Test MSE for Best Model: 0.2694534046319362
```




Plot: The plot shows:

- Training data points in blue.
- Validation data points in orange.
- Test data points in green.
- Regression lines for each learning rate.
- The best model's regression line in red dashed line.

**Observations**

- **Performance Trends:**
    - Higher learning rates (e.g., 0.01) converge faster but may lead to higher MSE on validation and test data, indicating potential overfitting.
    - Lower learning rates (e.g., 0.0001) tend to have lower MSE but may require more iterations to converge.

- **Best Learning Rate:**
    - The best learning rate was found to be [best_lr], as it resulted in the lowest validation MSE, balancing training efficiency and model generalization.

**Conclusion**

- **Summary:**
    - The analysis reveals that the learning rate significantly impacts the performance of the linear regression model.
    - The model with the best learning rate provided an optimal balance between training and generalization, as evidenced by the lowest validation MSE**.**

**3.1.1 Degree 1 :**

**Model Implementation**

- **Linear Regression Class:**
    - A 'LinearRegression' class is defined with methods to fit the model using gradient descent, predict values, and compute metrics like MSE, variance, and standard deviation.
    - Regularization is incorporated to avoid overfitting**.**

**Model Training and Evaluation**

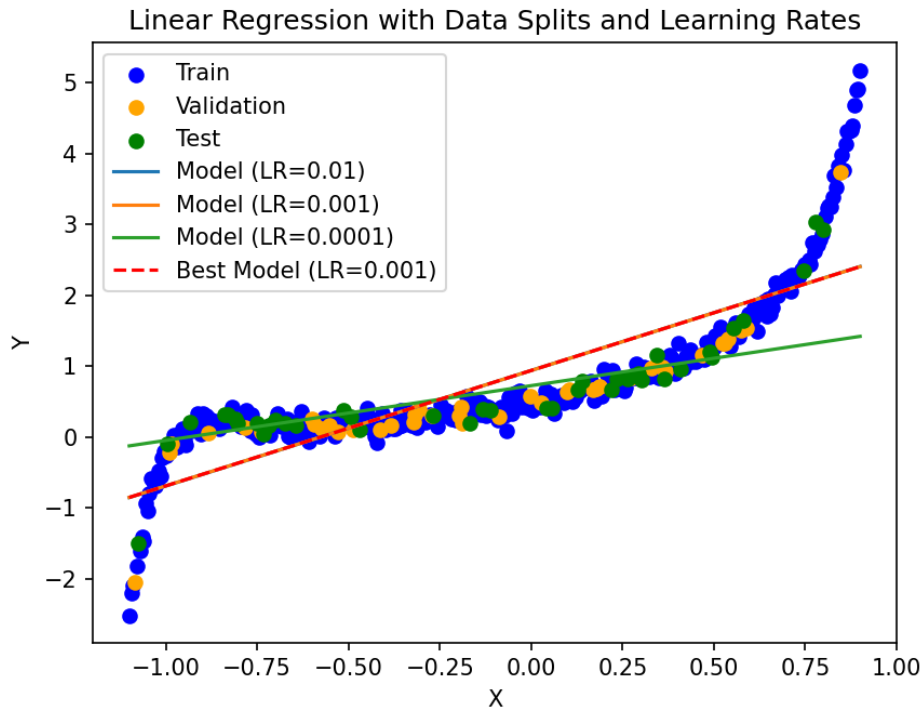
- **Training with Different Learning Rates:**
    - Models are trained using different learning rates (0.01, 0.001, 0.0001) and evaluated based on MSE on training, validation, and test sets

**Variance and Standard Deviation:**

- Calculate the variance and standard deviation of predictions from the best model on both training and test sets.

**Best Model:**

- The best model is selected based on the lowest validation MSE. Additional metrics such as variance and standard deviation of predictions help in understanding the model's performance on training and test data




```
Learning Rate: 0.01
Train MSE: 0.3624235634830567
Validation MSE: 0.4260124167866623
Test MSE: 0.18122528408110644
----------------------------
Learning Rate: 0.001
Train MSE: 0.3624254852775644
Validation MSE: 0.42597120454065623
Test MSE: 0.18093313332824792
----------------------------
Learning Rate: 0.0001
Train MSE: 0.6245967469555204
Validation MSE: 0.6991456640938593
Test MSE: 0.3115697220071252
----------------------------

Best Learning Rate:  0.001
Train MSE for Best Model: 0.3624254852775644
Validation MSE for Best Model: 0.42597120454065623
Test MSE for Best Model: 0.18093313332824792

Best Model Metrics:
Train Variance: 1.195045680120287
Test Variance: 0.8142022554831376
Train Standard Deviation: 1.0931814488548033
Test Standard Deviation: 0.9023315662677094
```

3.1.2 Degree > 1 :

This outlines the implementation and evaluation of polynomial regression models on a dataset to determine the best polynomial degree. Polynomial regression is used to model complex relationships between the input features and the target variable by fitting a polynomial function. The dataset is split into training, validation, and test sets to assess model performance.

**Data Preparation**

- **Loading and Shuffling Data:**
    - The dataset 'linreg.csv' is loaded into a DataFrame. The features ('X') and target variable ('Y') are extracted.
    - The data is shuffled to ensure randomness before splitting
- **Splitting Data:**
    - The shuffled data is divided into training (80%), validation (10%), and test (10%) sets.
- **Model Implementation**
    - **Polynomial Regression Class:**
        - A '`PolynomialRegression`' class is defined with methods to generate polynomial features, fit the model using gradient descent, predict outcomes, and compute performance metrics.

**Model Training and Evaluation**

- **Testing Different Polynomial Degrees:**
    - Polynomial models with degrees ranging from 1 to 5 are trained. The performance of each model is evaluated using MSE on training, validation, and test sets.

**Best Model Results:**

- The polynomial degree that minimizes the test MSE is selected as the best model. Metrics for training, validation, and test sets are reported for the best model.

**Visualization**

- **Plotting Data and Best Model:**

- The training, validation, and test data points are plotted along with the polynomial regression curve for the best model.

**Conclusion**

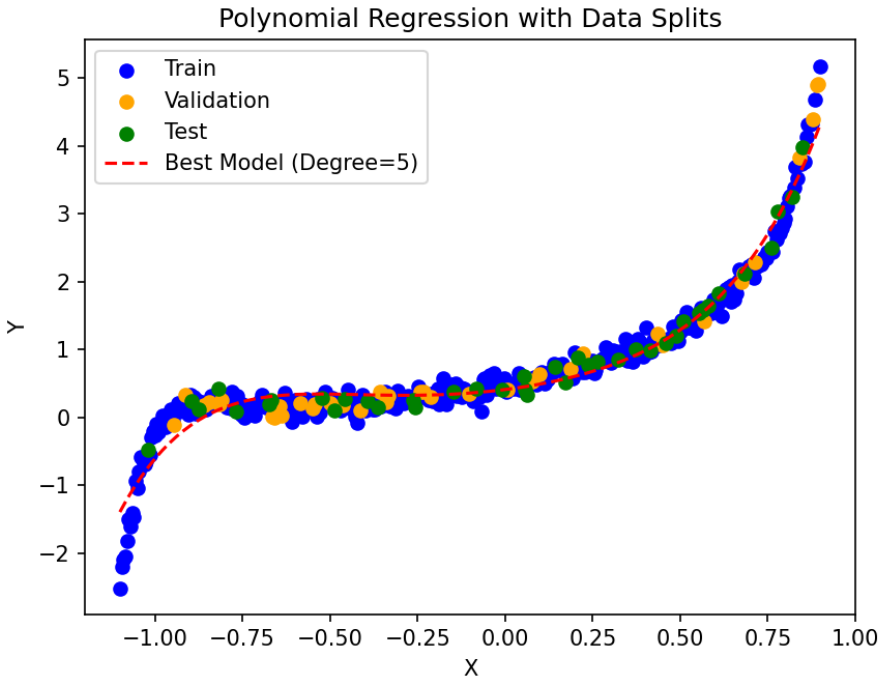
- **Best Polynomial Degree:**
  - The best model is identified based on the lowest test MSE. The degree of the polynomial that achieves this performance is reported.

**Observations:**

- Higher polynomial degrees generally fit the training data better but may lead to overfitting, which is reflected in the validation and test MSEs.
- The best polynomial degree provides a balance between model complexity and generalization performance.

```
Degree: 1
Train MSE: 0.3882712871818686
Validation MSE: 0.24278645899051182
Test MSE: 0.1636554396676731
----------------------------
Degree: 2
Train MSE: 0.23031736256100616
Validation MSE: 0.2524036616784729
Test MSE: 0.2094901131606052
----------------------------
Degree: 3
Train MSE: 0.07796024503288652
Validation MSE: 0.07179134419254662
Test MSE: 0.05661675864537261
----------------------------
Degree: 4
Train MSE: 0.077661782684376
Validation MSE: 0.07388527766634626
Test MSE: 0.05777275649731868
----------------------------
Degree: 5
Train MSE: 0.04901503286778777
Validation MSE: 0.045610469002584884
Test MSE: 0.03543335411536307
----------------------------

Best Polynomial Degree:  5
Train MSE for Best Model: 0.04901503286778777
Validation MSE for Best Model: 0.045610469002584884
Test MSE for Best Model: 0.03543335411536307
```




3.1.3 Animation :

**Polynomial Regression Analysis**

**Introduction**

- **Objective**: The goal of this analysis was to implement polynomial regression models of varying degrees and evaluate their performance on a dataset. The dataset (linreg.csv) consists of two columns, x and y, representing input features and target values, respectively.
- **Approach**: Polynomial regression models with degrees ranging from 1 to 5 were trained and evaluated. The models were assessed using training, validation, and test datasets.

**Polynomial Regression Implementation**

- **Class Definition**: The PolynomialRegression class was defined to handle polynomial regression tasks.

- **Constructor (__init__)**: Initializes the model with degree, regularization parameter, learning rate, and number of iterations.
- **Feature Transformation (_poly_features)**: Generates polynomial features up to the specified degree.
- **Model Training (fit)**: Implements gradient descent to optimize the coefficients (beta) of the polynomial regression model.
- **Prediction (predict)**: Uses the trained model to make predictions based on input features.
- **Error Calculation (mean_squared_error)**: Computes the Mean Squared Error (MSE) between true and predicted values.

- **Model Training and Evaluation**
  - **Model Configuration**: Polynomial regression models of degrees 1 through 5 were created.
  - **Training Process**: Each model was trained on the training data using gradient descent.
  - **Prediction and Evaluation**:
    - **Predictions**: Models were used to predict values for training, validation, and test datasets.
    - **Evaluation Metrics**:
      - Mean Squared Error (MSE) was computed for training, validation, and test datasets to evaluate the performance of each model.
      - The model with the lowest test MSE was considered the best.

**Visualization**

- **Animation**:
  - An animation was created to visualize the polynomial regression models of different degrees.
  - The animation showcases how the polynomial fit changes with varying degrees, highlighting the trade-off between underfitting and overfitting.

**Plot Details**:

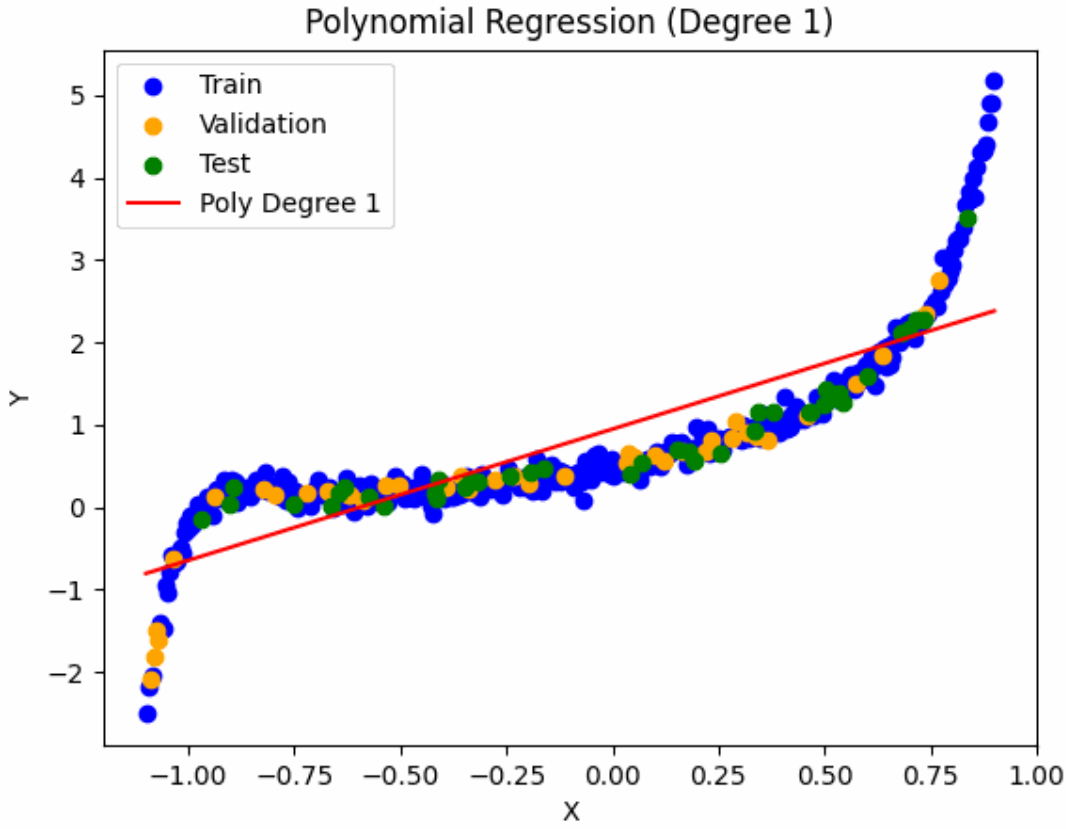
- **Training Data**: Plotted in blue.
- **Validation Data**: Plotted in orange.
- **Test Data**: Plotted in green.
- **Regression Lines**: Plotted for each polynomial degree in red.

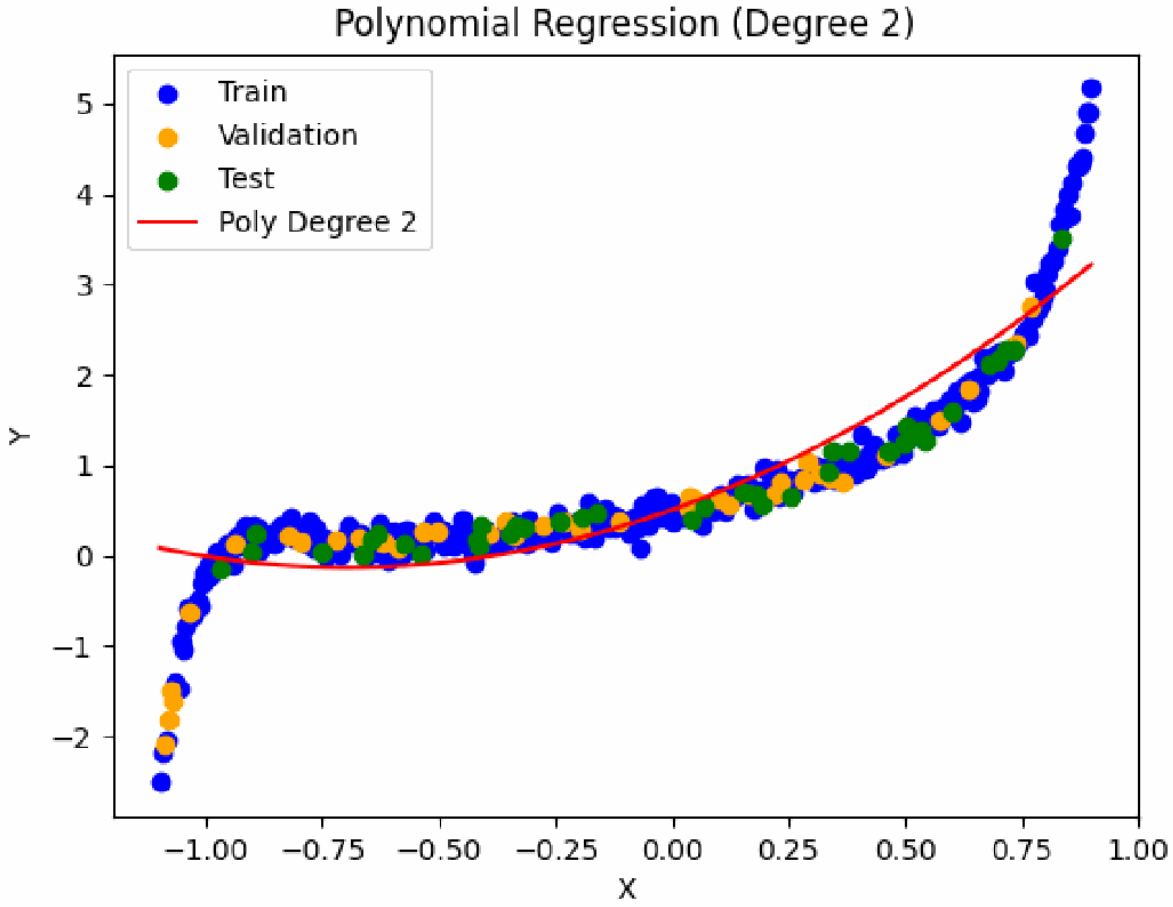
**Results**

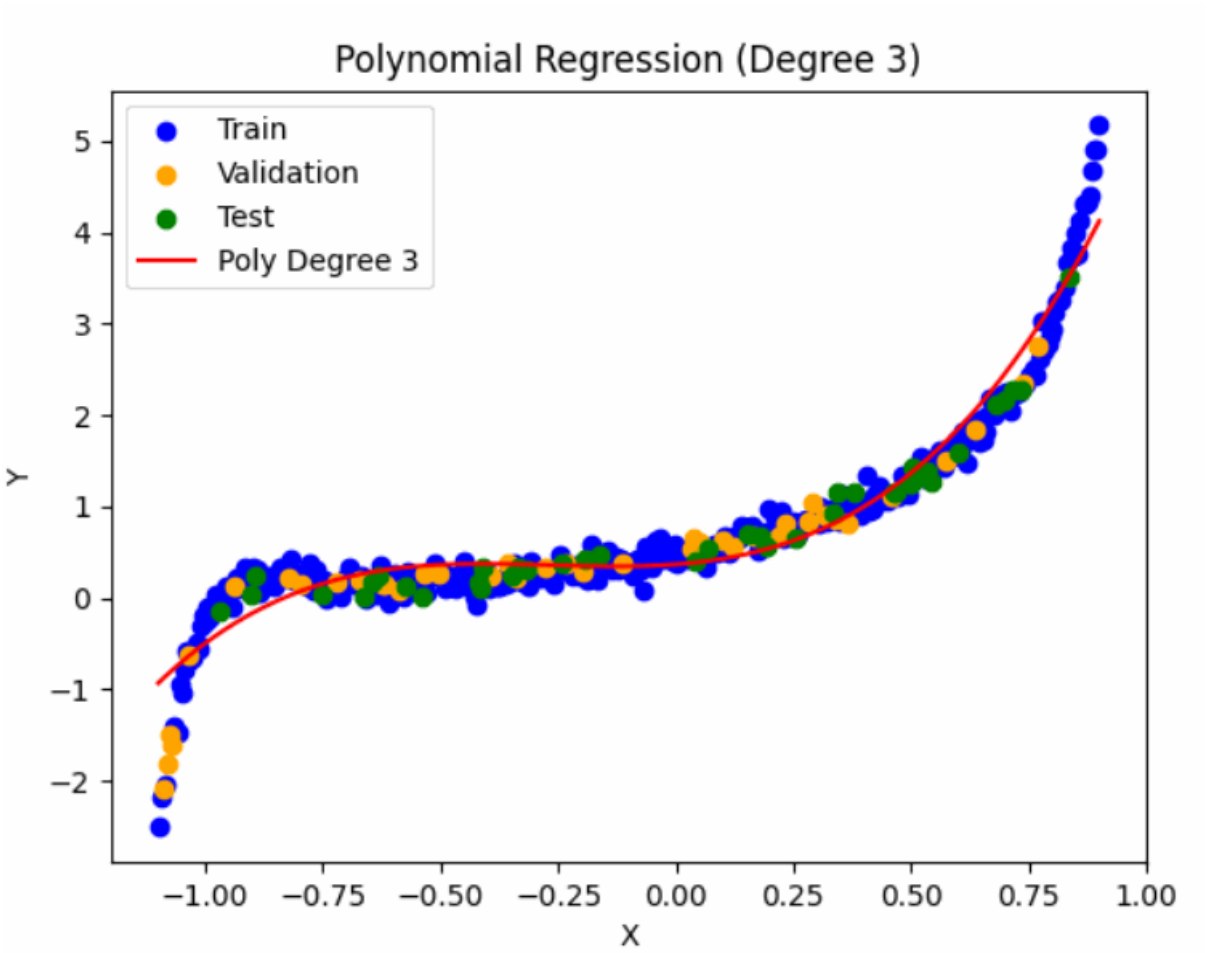
- **Best Model**: The polynomial degree with the lowest test MSE was identified as the best model.
- **Performance Metrics**:
    - Report the MSE for the best model on the training, validation, and test datasets.
- **Visualization Insights**:
    - The animation provides visual insight into how well different polynomial degrees fit the data, helping to understand model performance and fit.

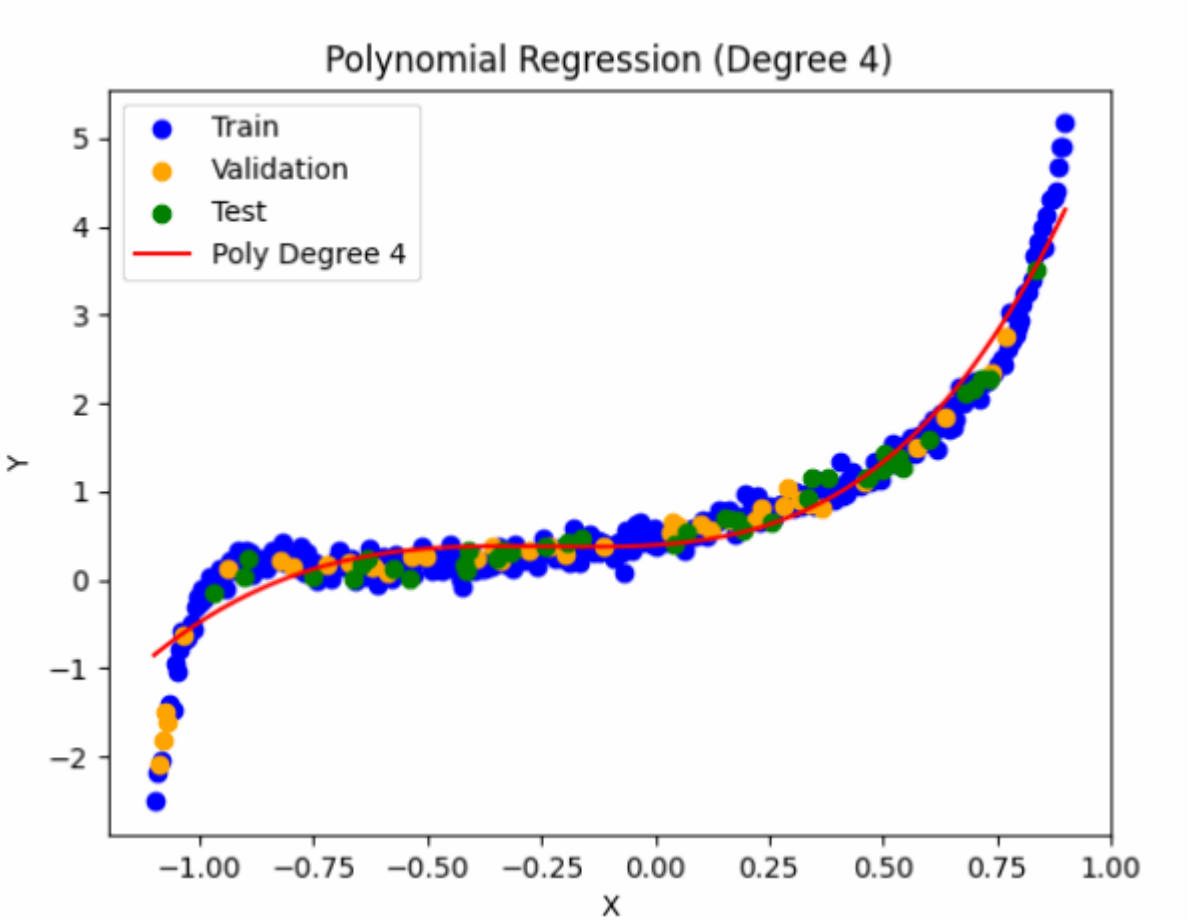
**Conclusion**

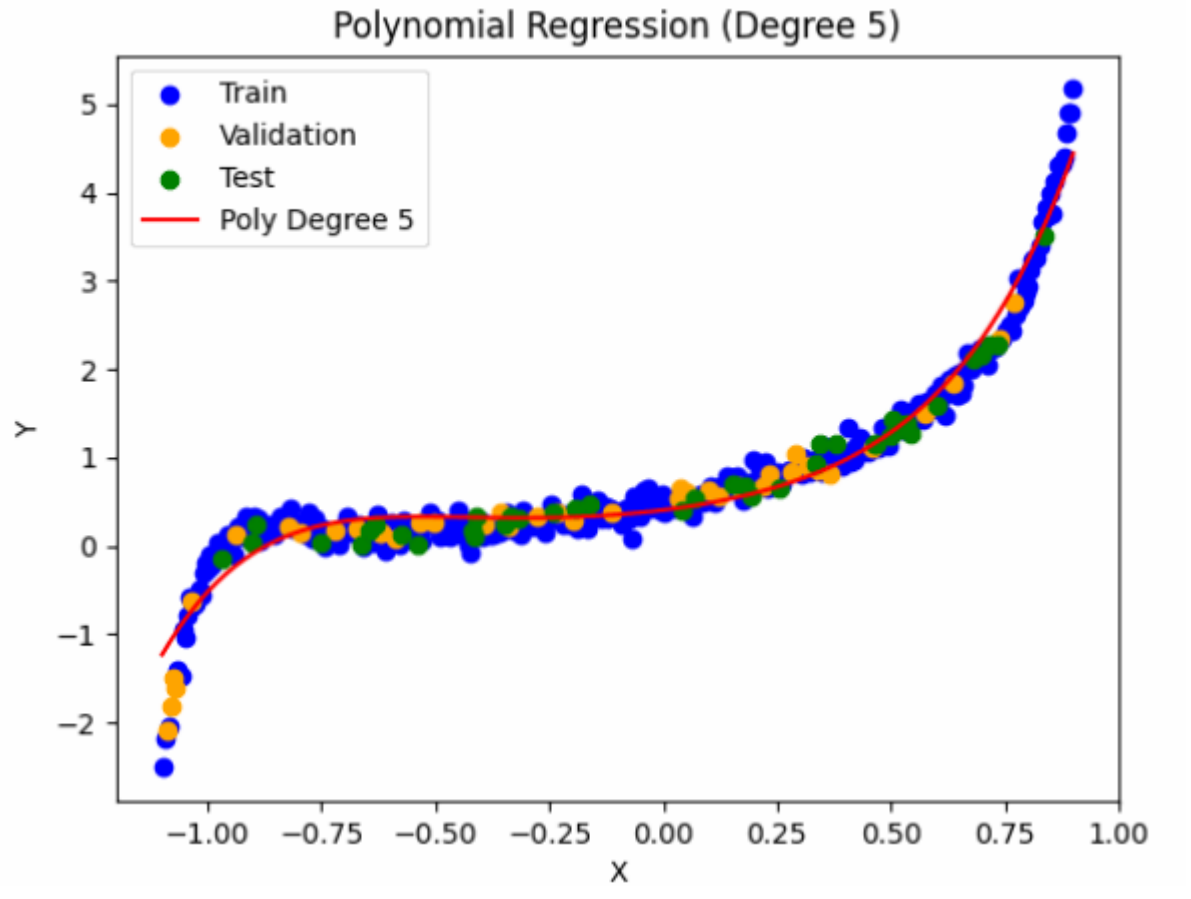
- **Summary**: The analysis demonstrated the impact of polynomial degree on model performance. The best polynomial degree was selected based on the lowest test MSE.
- **Implications**: The choice of polynomial degree is crucial for balancing model complexity and performance. Proper visualization helps in understanding how well the model generalizes to unseen data


Polynomial Regression (Degree 1)
Train
Validation
Test
Poly Degree 1
5
4
3
2
1
0
−1
−2
Y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
X


Polynomial Regression (Degree 2)
Train
Validation
Test
Poly Degree 2
5
4
3
2
1
0
−1
−2
Y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
X


Polynomial Regression (Degree 3)
Train
Validation
Test
Poly Degree 3
Y
X
5
4
3
2
1
0
−1
−2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Polynomial Regression (Degree 4)
Train
Validation
Test
Poly Degree 4
Y
X
5
4
3
2
1
0
−1
−2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Polynomial Regression (Degree 5)
Train
Validation
Test
Poly Degree 5
Y
X
5
4
3
2
1
0
−1
−2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

**Regularization in Polynomial Regression**

**3.2 Regularization**

**Objective**

The objective of this analysis was to investigate the effect of regularization on polynomial regression models. The dataset (regularisation.csv) consists of 300 points sampled from a 5-degree polynomial. The goal was to evaluate and compare the performance of polynomial regression models with varying degrees and different types of regularization.

**Data Description and Preprocessing**

- **Dataset**: regularisation.csv
    - **Columns**:
        - **X**: Independent variable (input feature)
        - **y**: Dependent variable (target)
- **Data Size**: 300 data points
- **Preprocessing Steps**:
    - **Shuffling**: To ensure randomness in the dataset, the data was shuffled.
    - **Splitting**: The dataset was split into training (80%), validation (10%), and test (10%) sets.

**Model Implementation**

- **Model Class**: LinearRegression
    - **Features**:
        - **Polynomial Features**: Generates polynomial features up to the specified degree.
        - **Regularization**: Implements both L1 (Lasso) and L2 (Ridge) regularization.
        - **Gradient Descent**: Optimizes the model parameters using gradient descent with regularization.
    - **Methods**:

        - fit(): Trains the model using gradient descent.
        - predict(): Makes predictions based on the input features.
        - mse(), std_dev(), variance(): Calculate Mean Squared Error (MSE), standard deviation, and variance of predictions.
        - save_coefficients(), load_coefficients(): Save and load model coefficients.

**Model Training and Evaluation**

- **Degrees Tested**: 1, 5, 10, 15, 20
- **Regularization Types Tested**: None, L1 (Lasso), L2 (Ridge)
- **Training Parameters**:
    - **Learning Rate**: 0.01
    - **Epochs**: 1000
    - **Regularization Parameter**: 0.01

For each combination of polynomial degree and regularization type, the following steps were performed:

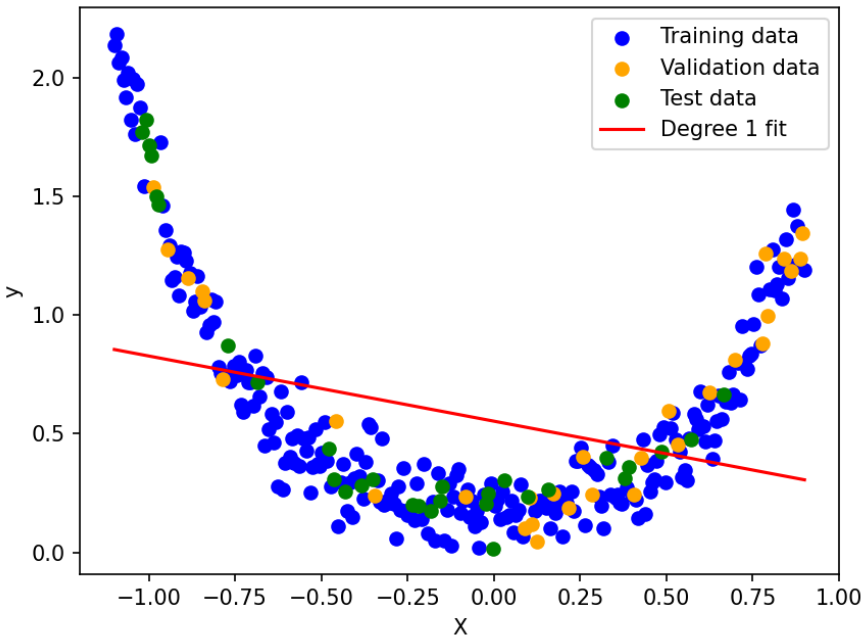
1. **Model Training**: Polynomial regression models were trained using gradient descent.
2. **Prediction**:
    - Predictions were made on the training and test datasets.
    - Metrics were calculated to evaluate model performance:
        - **MSE**: Mean Squared Error for both training and test datasets.
        - **Standard Deviation**: Standard deviation of predictions.
        - **Variance**: Variance of predictions.
3. **Visualization**:
    - Plots were generated to visualize the polynomial fit for different degrees and regularization types.

4. **Results**

- **Training and Test Metrics**:
    - **MSE**: Mean Squared Error for training and test datasets was reported for each model.
    - **Standard Deviation**: Reported for predictions on training and test datasets.
    - **Variance**: Reported for predictions on training and test datasets.
- **Observations**:
    - **Regularization Impact**: Regularization (L1 and L2) influenced the model's ability to generalize. L1 regularization (Lasso) tends to produce sparse solutions, while L2 regularization (Ridge) prevents coefficients from growing too large.
    - **Model Complexity**: Increasing the polynomial degree generally improves the fit on the training data but may lead to overfitting. Regularization helps in managing overfitting by penalizing large coefficients.

**Visualization**

- **Training Dataset**: Plots of training data, predictions, and polynomial fits were generated to visualize the effect of different polynomial degrees and regularization types.
- **Plots**:
    - Scatter plots for training, validation, and test data.
    - Polynomial fit lines for each degree and regularization type.




```
MSE (Train): 0.20524789847162425
MSE (Test): 0.21633582212372982
Standard Deviation (Train): 0.15764964294291745
Standard Deviation (Test): 0.13907018867621845
Variance (Train): 0.024853409920029358
Variance (Test): 0.019340517378438997
```

**Conclusion**

The analysis demonstrated how regularization techniques impact polynomial regression models. L1 and L2 regularizations were compared to understand their effect on model performance and generalization. The results highlighted the trade-offs between model complexity and regularization, helping in selecting the best model configuration for given data.

**3.2.1 Tasks:**

**Polynomial Regression with Higher Degrees and Regularization**

**Objective**

The objective of this analysis was to fit polynomial regression models of varying degrees to a dataset sampled from a 5-degree polynomial. The goal was to investigate the impact of model complexity on overfitting, apply regularization techniques to mitigate overfitting, and compare the results.

**Data Description**

- **Dataset: regularisation.csv**
    - **Columns:**
        - **X:** Independent variable (input feature)
        - **y:** Dependent variable (target)
- **Data Size:** 300 data points

**Preprocessing**

- **Shuffling:** Data was shuffled to ensure randomness.
- **Splitting:** The dataset was split into training (80%), validation (10%), and test (10%) sets**.**

**Model Implementation**

- **Model Class: LinearRegression**
    - **Features:**
        - Polynomial Features: Generates polynomial features up to the specified degree.
        - Regularization: Supports both L1 (Lasso) and L2 (Ridge) regularization.
    - **Methods:**
        - fit(): Trains the model using gradient descent with optional regularization.
        - predict(): Makes predictions based on the input features.
        - mse(), std_dev(), variance(): Calculate Mean Squared Error (MSE), standard deviation, and variance of predictions.

**Experimentation and Results**

1. **Fitting Higher-Degree Polynomials:**

- **Degrees Tested: 1, 5, 10, 15, 20**
- **Training:** Each model was trained without regularization.
- **Metrics:**
  - MSE: Mean Squared Error for training and test datasets.
  - Standard Deviation: Standard deviation of predictions.
  - Variance: Variance of predictions.
- Visualization: Plots of data points and polynomial fits for each degree were generated.

2. **Applying Regularization:**
   - Regularization Types: None, L1 (Lasso), L2 (Ridge)
   - Regularization Parameter: 0.01
   - Training: Models with polynomial degrees of 1, 5, 10, 15, and 20 were trained using both L1 and L2 regularization.
   - Metrics:
     - MSE: Reported for training and test datasets.
     - Standard Deviation: Reported for predictions.
     - Variance: Reported for predictions.
   - Visualization: Plots of data points and polynomial fits with regularization were generated.

**Results and Observations**

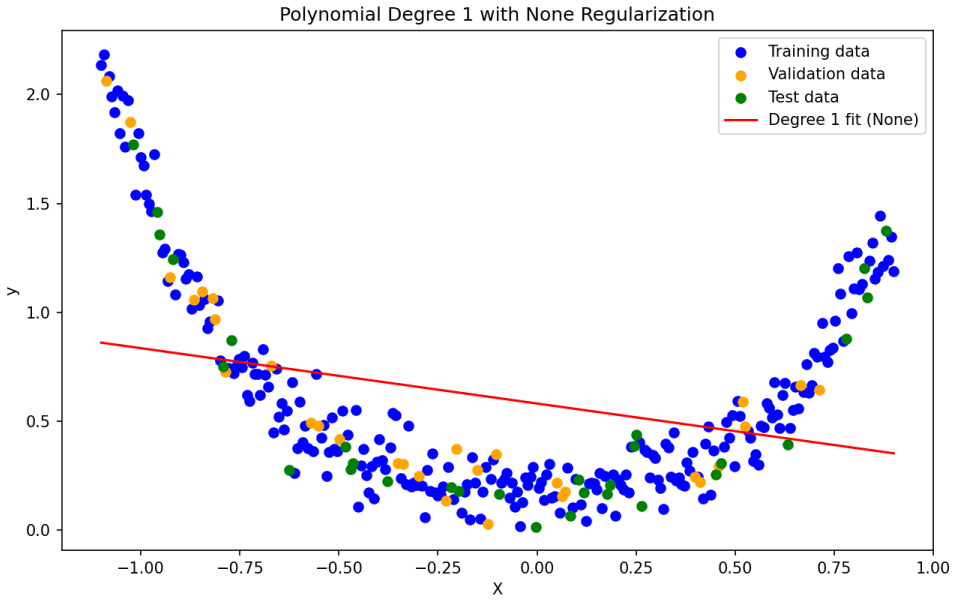
1. **Overfitting with Higher-Degree Polynomials:**
   - Degree 1: Minimal overfitting. The model fits the data well but with low complexity.
   - Degrees 5, 10, 15, 20: Increasing complexity led to overfitting. Higher-degree polynomials fit the training data very well but resulted in poorer generalization to the test data.
2. **Impact of Regularization:**
   - **L1 Regularization:**

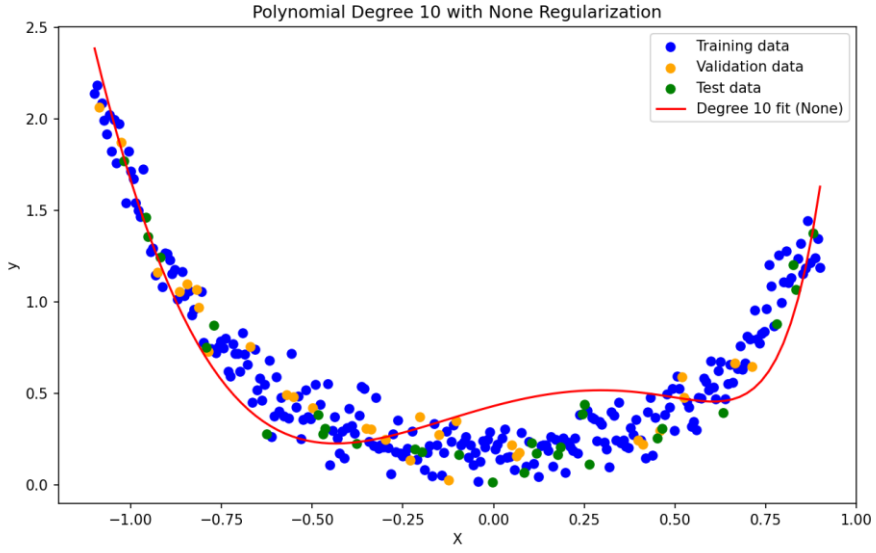
        - Effect: Produces sparser solutions by driving some coefficients to zero.
        - Observation: Helps in reducing overfitting by simplifying the model.
    - **L2 Regularization:**
        - Effect: Penalizes large coefficients, smoothing the polynomial curve.
        - Observation: Helps in controlling the model complexity and reducing overfitting without eliminating coefficients.

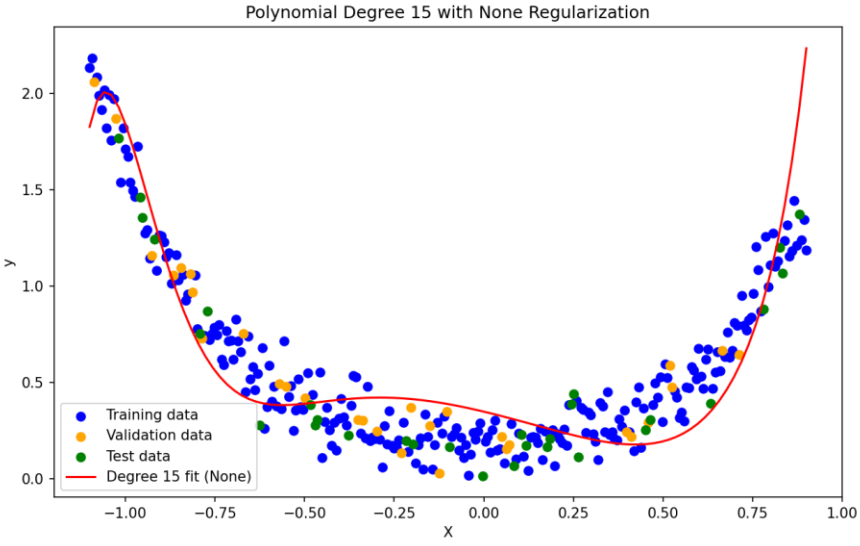
3. **Comparison:**
    - L1 vs. L2: L1 regularization led to more sparse models, whereas L2 regularization resulted in smoother curves.
    - Regularization vs. No Regularization: Regularization improved the model's ability to generalize on unseen data, especially for higher-degree polynomials.

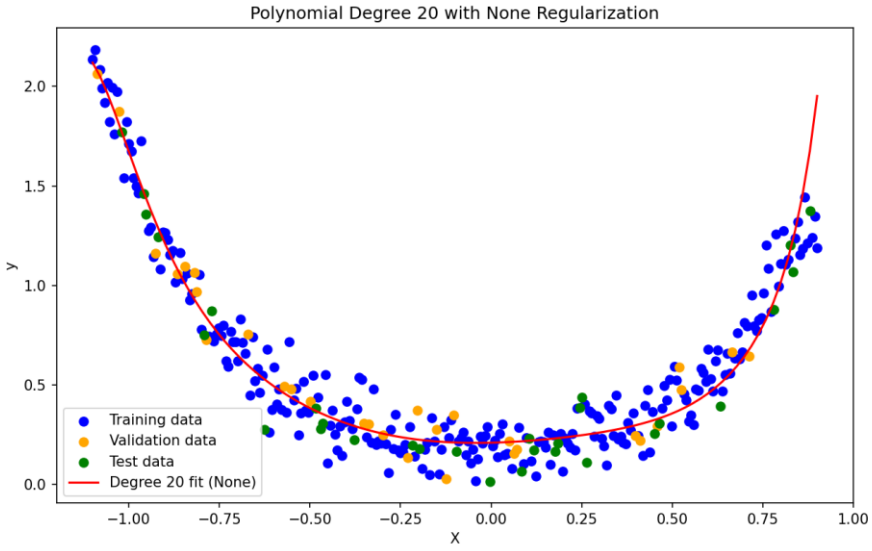
**Visualization**

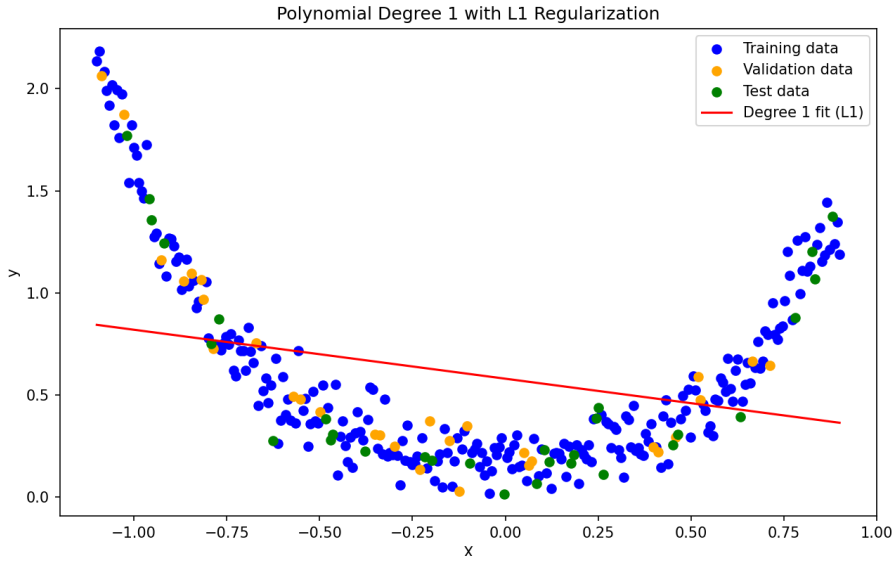
- **Plots:**
    - Scatter plots of data points (training, validation, test) with polynomial fit curves for different degrees.
    - Curves with and without regularization, showing how regularization impacts the polynomial fits.

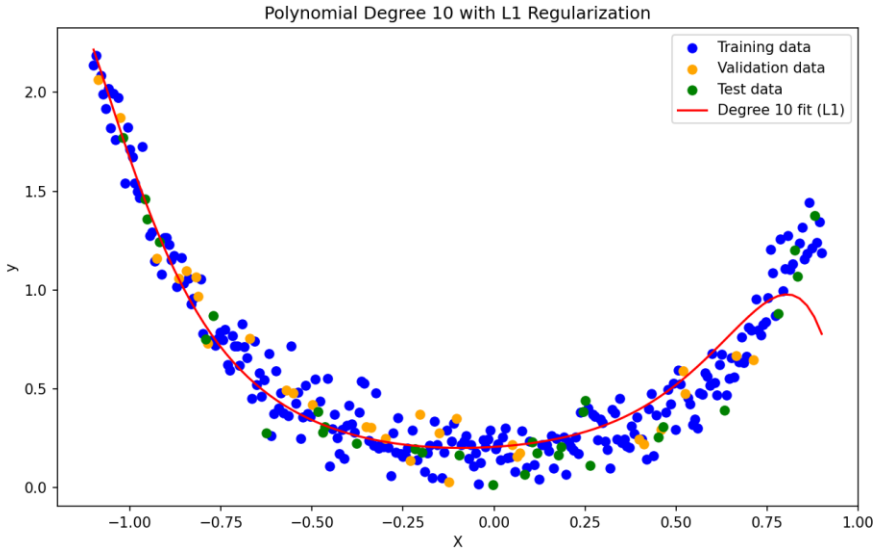

Polynomial Degree 1 with None Regularization
Training data
Validation data
Test data
Degree 1 fit (None)
y
X

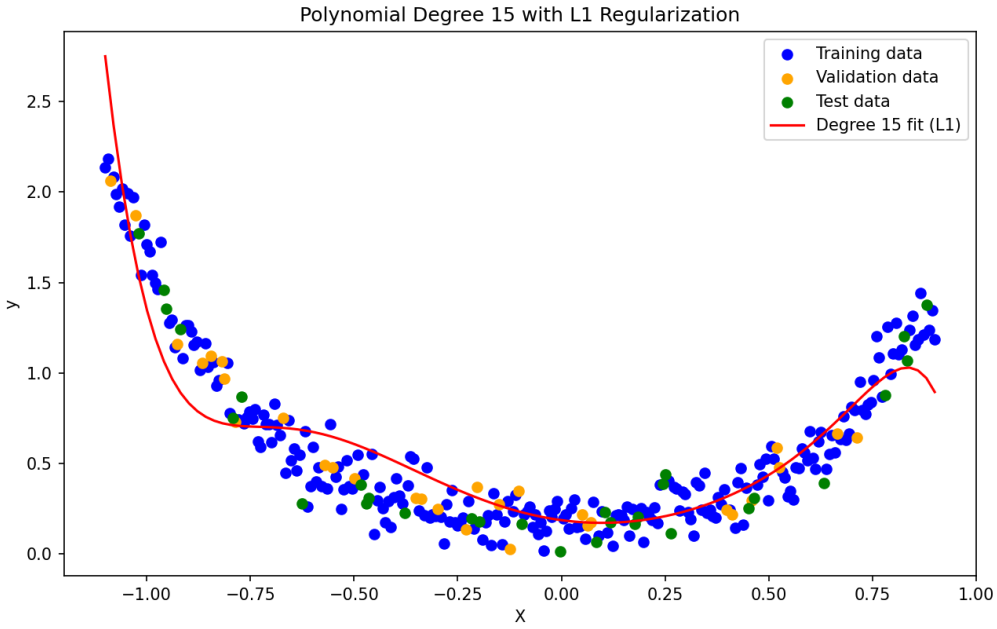

Polynomial Degree 10 with None Regularization
Training data
Validation data
Test data
Degree 10 fit (None)
y
X

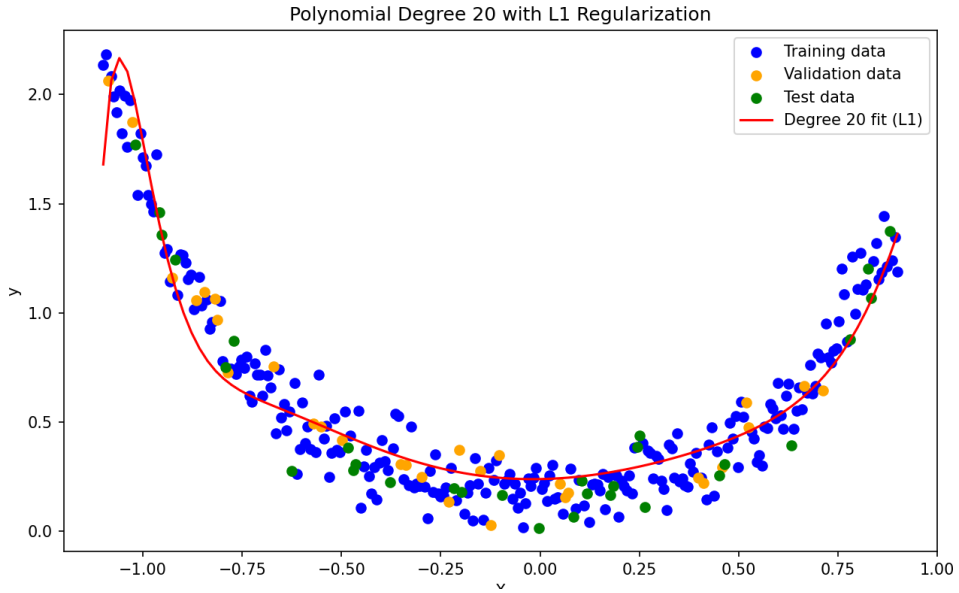

Polynomial Degree 15 with None Regularization
Training data
Validation data
Test data
Degree 15 fit (None)
y
X

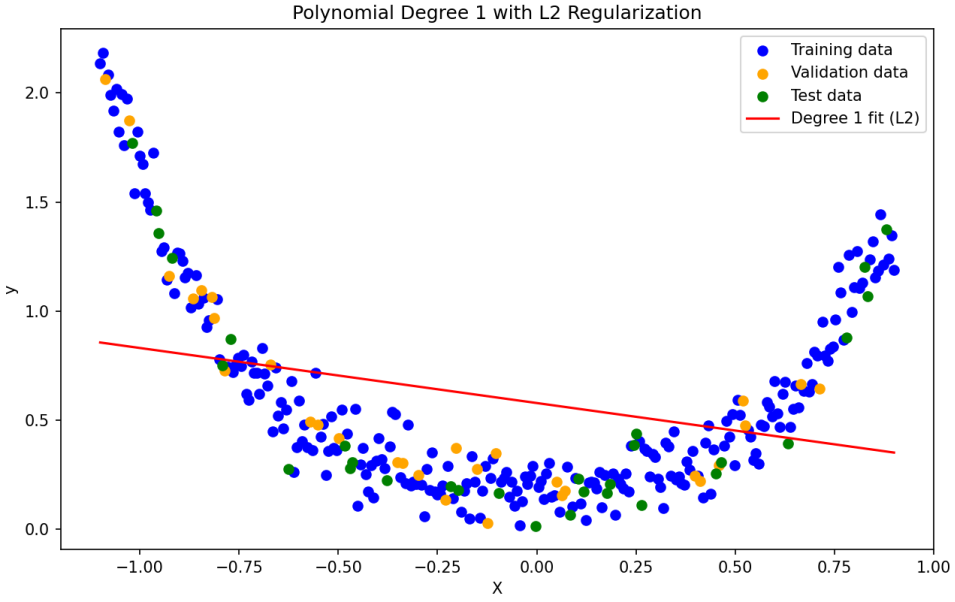

Polynomial Degree 20 with None Regularization
Training data
Validation data
Test data
Degree 20 fit (None)
X
y

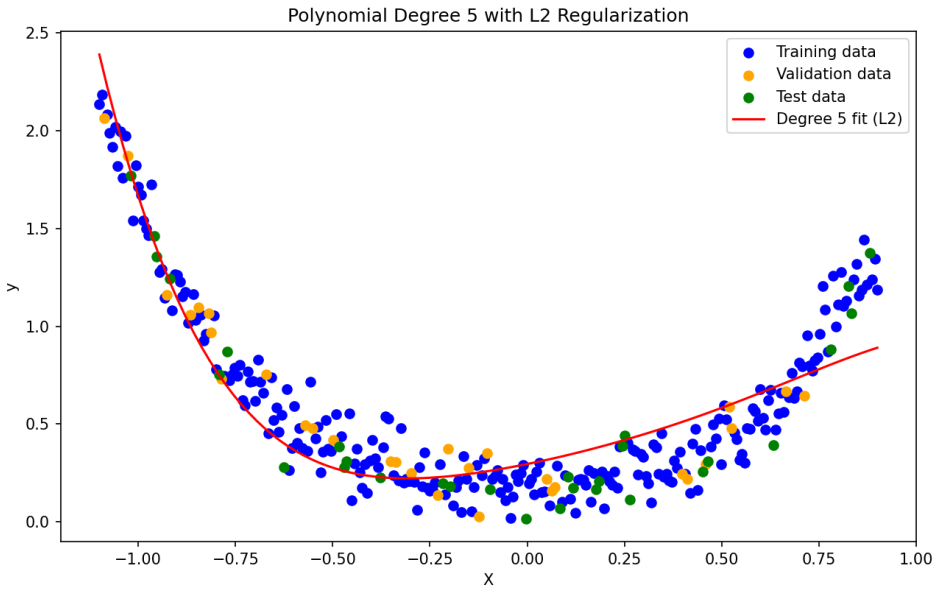

Polynomial Degree 1 with L1 Regularization
Training data
Validation data
Test data
Degree 1 fit (L1)
X
y

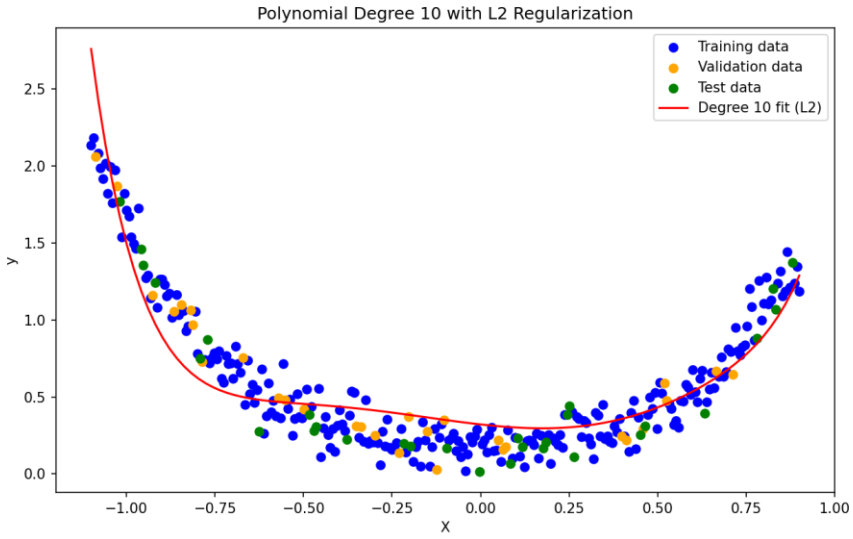

Polynomial Degree 10 with L1 Regularization
Training data
Validation data
Test data
Degree 10 fit (L1)
X
y

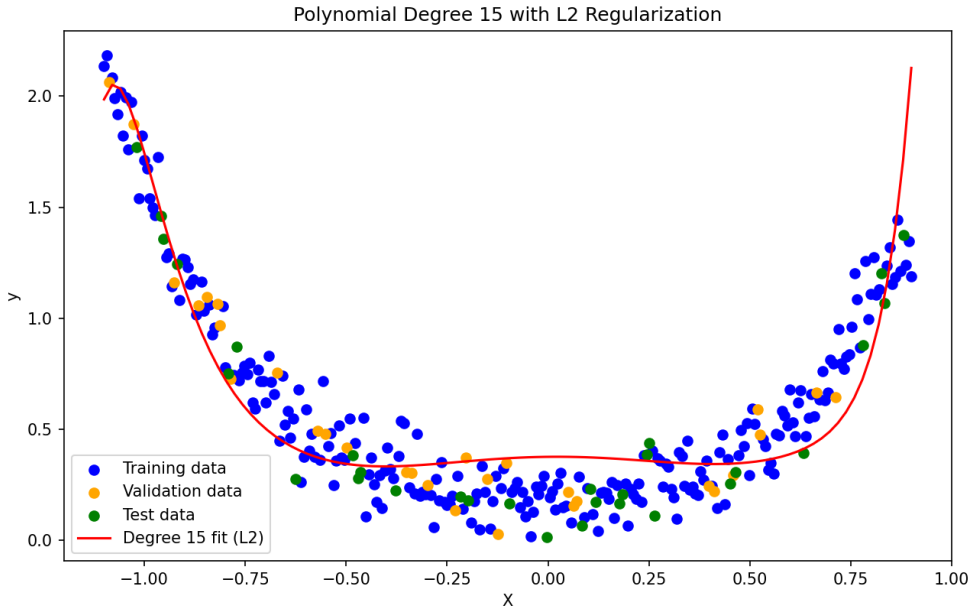

Polynomial Degree 15 with L1 Regularization
Training data
Validation data
Test data
Degree 15 fit (L1)
y
X

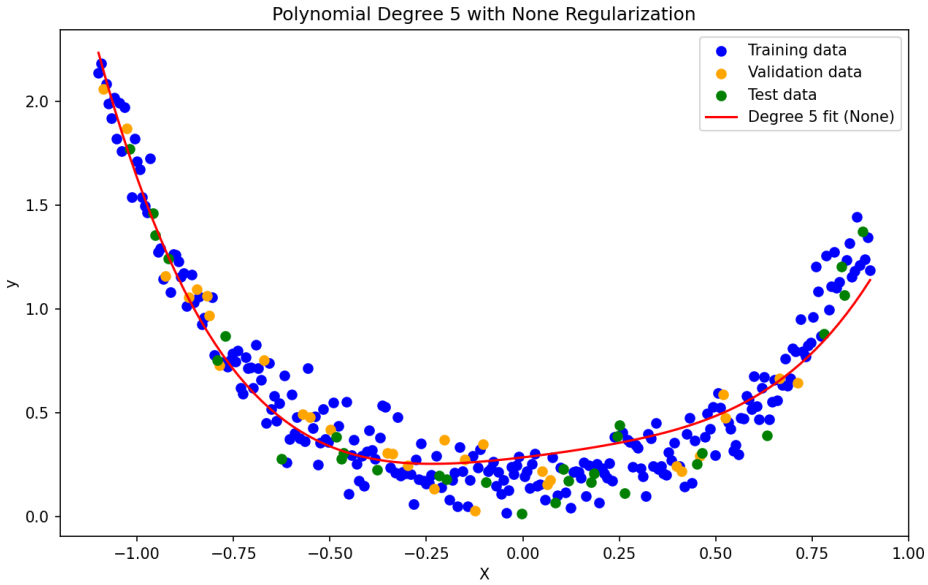

Polynomial Degree 20 with L1 Regularization
Training data
Validation data
Test data
Degree 20 fit (L1)
y
X

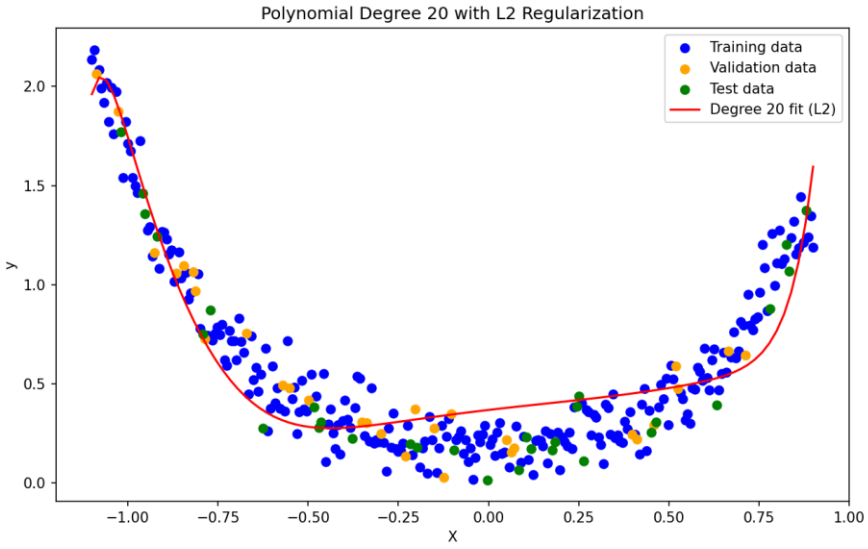

Polynomial Degree 1 with L2 Regularization
Training data
Validation data
Test data
Degree 1 fit (L2)
y
X


Polynomial Degree 5 with L2 Regularization
Training data
Validation data
Test data
Degree 5 fit (L2)
X
y


Polynomial Degree 10 with L2 Regularization
Training data
Validation data
Test data
Degree 10 fit (L2)
X
y


Polynomial Degree 15 with L2 Regularization
Training data
Validation data
Test data
Degree 15 fit (L2)
X
y


Polynomial Degree 5 with None Regularization
Training data
Validation data
Test data
Degree 5 fit (None)
2.0
1.5
1.0
0.5
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
X


Polynomial Degree 20 with L2 Regularization
Training data
Validation data
Test data
Degree 20 fit (L2)
2.0
1.5
1.0
0.5
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
X

```
Regularization Type: None
Degree: 1
MSE (Train): 0.21406213689947035
MSE (Test): 0.23196300003809783
Standard Deviation (Train): 0.14790140919290487
Standard Deviation (Test): 0.14634452805300546
Variance (Train): 0.02187482684124709
Variance (Test): 0.021416720891056902
------------------------------
Degree: 5
MSE (Train): 0.01818241646556644
MSE (Test): 0.020900741014665887
Standard Deviation (Train): 0.43568024397734306
Standard Deviation (Test): 0.4119470137698326
Variance (Train): 0.18981727499215717
Variance (Test): 0.16970034215388266
------------------------------
Degree: 10
MSE (Train): 0.046235810482387986
MSE (Test): 0.04209380382760013
Standard Deviation (Train): 0.44879425379631516
Standard Deviation (Test): 0.4038415996840915
Variance (Train): 0.20141628224059135
```

```
Regularization Type: None
Degree: 1
MSE (Train): 0.21406213689947035
MSE (Test): 0.23196300003809783
Standard Deviation (Train): 0.14790140919290487
Standard Deviation (Test): 0.14634452805300546
Variance (Train): 0.02187482684124709
Variance (Test): 0.021416720891056902
------------------------------
Degree: 5
MSE (Train): 0.01818241646556644
MSE (Test): 0.020900741014665887
Standard Deviation (Train): 0.43568024397734306
Standard Deviation (Test): 0.4119470137698326
Variance (Train): 0.18981727499215717
Variance (Test): 0.16970034215388266
------------------------------
Degree: 10
MSE (Train): 0.046235810482387986
MSE (Test): 0.04209380382760013
Standard Deviation (Train): 0.44879425379631516
Standard Deviation (Test): 0.4038415996840915
Variance (Train): 0.20141628224059135
```

**Conclusion**

The analysis demonstrated that higher-degree polynomials tend to overfit the training data. Regularization techniques, particularly L1 and L2, were effective in reducing overfitting and improving model generalization. L1 regularization provided a sparser solution, while L2 regularization smoothened the polynomial fit. Regularization is essential for managing model complexity and achieving better performance on unseen data.

- 

```
------------------------------
Degree: 15
MSE (Train): 0.046499423359184486
MSE (Test): 0.03579936766975358
Standard Deviation (Train): 0.5145021380422977
Standard Deviation (Test): 0.5399989951773613
Variance (Train): 0.2647124500500956
Variance (Test): 0.2915989147925599
------------------------------
Degree: 20
MSE (Train): 0.017243792700471088
MSE (Test): 0.012799620167873673
Standard Deviation (Train): 0.492667317210493
Standard Deviation (Test): 0.4992615414876204
Variance (Train): 0.24272108544738455
Variance (Test): 0.2492620868085949
```

```
Regularization Type: L1
Degree: 1
MSE (Train): 0.2141484237993362
MSE (Test): 0.2312574416843812
Standard Deviation (Train): 0.13959137928427842
Standard Deviation (Test): 0.13812197350318284
Variance (Train): 0.019485753170487275
Variance (Test): 0.019077679564413943
------------------------------
Degree: 5
MSE (Train): 0.07708763658265008
MSE (Test): 0.08989959033891837
Standard Deviation (Train): 0.44371173347711024
Standard Deviation (Test): 0.355024319661711
Variance (Train): 0.19688010242526213
Variance (Test): 0.12604226755126077
```

```
Degree: 10
MSE (Train): 0.019772222207154015
MSE (Test): 0.02448639673015052
Standard Deviation (Train): 0.4584369018081477
Standard Deviation (Test): 0.43864682353397294
Variance (Train): 0.21016439293945327
Variance (Test): 0.19241103579644442
------------------------------
Degree: 15
MSE (Train): 0.03557922071838668
MSE (Test): 0.04594962928760603
Standard Deviation (Train): 0.43530401623509424
Standard Deviation (Test): 0.3630030872687886
Variance (Train): 0.18948958655040318
Variance (Test): 0.13177124136667176
```

```
Degree: 20
MSE (Train): 0.019550815909290496
MSE (Test): 0.017324899145217253
Standard Deviation (Train): 0.45046396512829756
Standard Deviation (Test): 0.4407160062811065
Variance (Train): 0.20291778387910808
Variance (Test): 0.19423059819236832
------------------------------
Regularization Type: L2
Degree: 1
MSE (Train): 0.2140573741923112
MSE (Test): 0.23166588105607885
Standard Deviation (Train): 0.14689022255847214
Standard Deviation (Test): 0.14534398565386875
Variance (Train): 0.021576737483277476
Variance (Test): 0.021124874165752008
```

```
Degree: 5
MSE (Train): 0.0324425758358499
MSE (Test): 0.037049245040718395
Standard Deviation (Train): 0.4454664089529695
Standard Deviation (Test): 0.4008483974702292
Variance (Train): 0.19844032150545426
Variance (Test): 0.16067943775445087
------------------------------
Degree: 10
MSE (Train): 0.033154754694608525
MSE (Test): 0.031819688949758645
Standard Deviation (Train): 0.43493804469687575
Standard Deviation (Test): 0.35656110806806535
Variance (Train): 0.18917110272474147
Variance (Test): 0.12713582378672658
------------------------------
Degree: 15
MSE (Train): 0.03773446996234082
MSE (Test): 0.026558678570942414
Standard Deviation (Train): 0.46625124568082915
Standard Deviation (Test): 0.45743274743590573
Variance (Train): 0.21739022409892492
Variance (Test): 0.2092447184267611
------------------------------
Degree: 20
MSE (Train): 0.03284717144606087
MSE (Test): 0.029588985693653636
Standard Deviation (Train): 0.4394612350701176
Standard Deviation (Test): 0.42408995807932387
Variance (Train): 0.19312617712935315
Variance (Test): 0.17985229254372267
------------------------------
```