ML Bootcamp Report

~Mahima Sunil Bachhav  
IIT(ISM)Dhanbad

Problem Statement:  
According to problem statement we have to implement Linear (and Polynomial) Regression, Logistic Regression, KNN, K-Means Clustering and an n-layer Neural Networks from scratch with the training dataset provided to us and to test it on hidden test dataset to evaluate our implementation.

# Linear Regression:

## Introduction:

The goal of this project was to implement linear regression from scratch using a dataset with dimensions 50,000 x 20. Linear regression is a fundamental algorithm in machine learning used for predicting a continuous target variable based on one or more independent features.

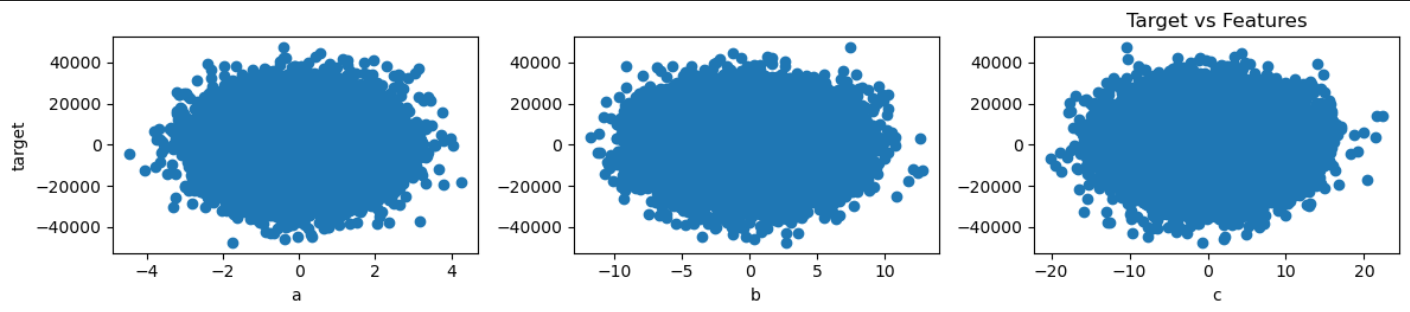
## Dataset Description:

The training dataset provided consists of 50,000 samples, each containing 20 features while the testing dataset consists of 10,000 samples, each containing 20 features. These features represent various characteristics or attributes of the data points. The target variable is assumed to be continuous, and the objective is to develop a linear regression model that can accurately predict this target variable based on the input features.

## Implementation Details:

### 1. Data Preprocessing

* **Loading Data:** The dataset was loaded into the program using pandas, and basic exploratory data analysis was performed to understand the structure and distribution of the features and target variable. Graphs were plotted to understand the relation between target values and features



* **Train-Test Split:** The dataset was split into training and testing sets to evaluate the model's performance on unseen data.

## 2. Linear Regression Model

A linear regression model was implemented from scratch using the following steps:

* **Initialization:** The model parameters (coefficients and bias) were initialized.
* **Cost Function:** The mean squared error (MSE) cost function was used to measure the difference between predicted and actual values.
* **Gradient Descent:** The model parameters were updated iteratively using gradient descent to minimize the cost function.
* **Regularization:** Regularization was used to prevent overfitting.

### 3. Training and Evaluation

The model was trained on the training dataset using the implemented linear regression algorithm. The performance was evaluated on the testing set using appropriate evaluation metrics such as mean squared error.

### 4. Hyperparameters:

Alpha:  
I tried various different values for alpha. Here’s what I observed

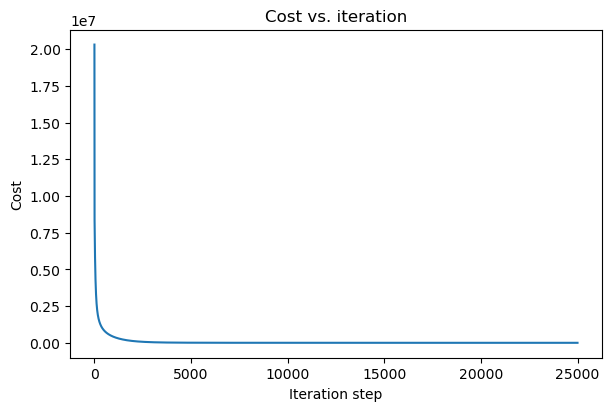
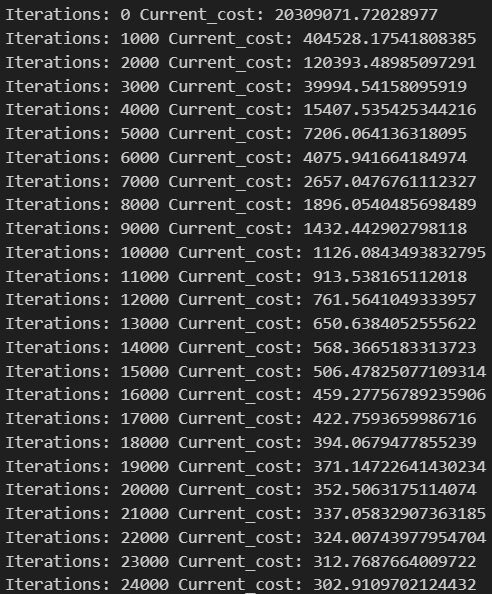
Alpha = 0.0000000001: The value of cost was converging slowly and number of iterations required was huge number, so I changed the value

Alpha = 0.01: When I used this value of alpha and the cost overshoot, so I decreased alpha value

Alpha = 0.00001: While using this alpha value the cost decreased constantly and saturated much faster.

Number of Iterations: 25000

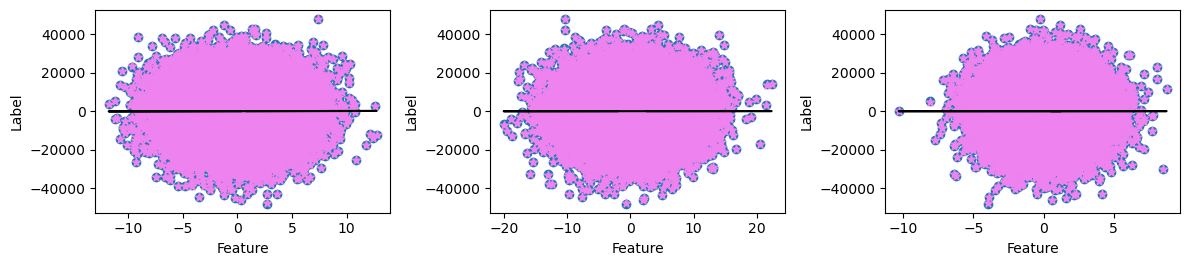
Lambda: 0.01   
I selected this value after trying various others lambda values and computed the cost of the test data and at this lambda value the cost was least

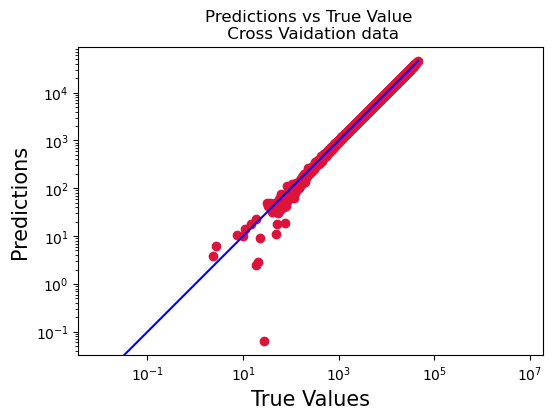
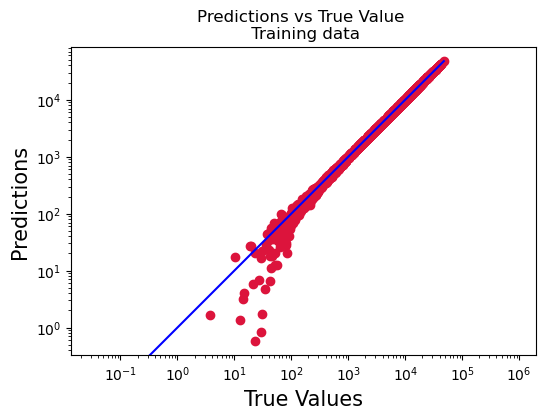


## Results and Analysis

The model's performance was analysed based on the evaluation metrics. Visualization techniques, such as learning curves and predicted vs. actual plots, were used to gain insights into the model's behaviour.

R2 Score was used to find the accuracy of the model.  
R2 score for validation data = 0.9999955126671569  
R2 score for training data = 0.9999955159335259





## Challenges and Future Work

* **Hyperparameter Tuning:** Experimenting with different hyperparameter values and regularization strengths could further improve the model's performance.

# Polynomial Regression:

## Introduction

The training dataset provided consists of 50,000 samples, each containing 3 features while the testing dataset consists of 10,000 samples, each containing 3 features. Polynomial regression is an extension of linear regression that allows for capturing non-linear relationships between features and the target variable.

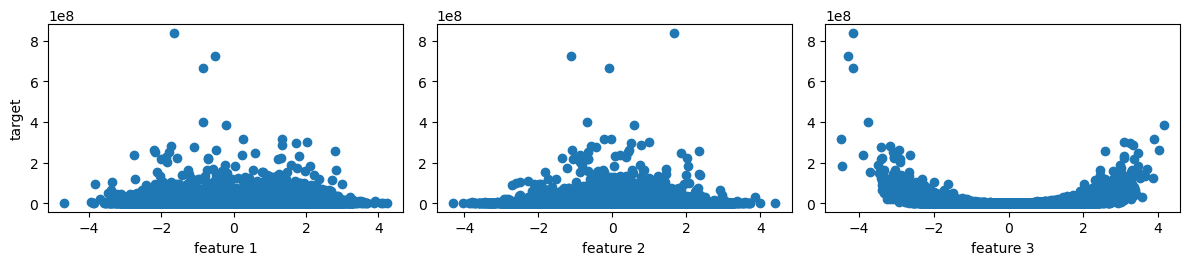
## Dataset Description

The dataset provided consists of 50,000 samples, each containing three features. These features represent various characteristics or attributes of the data points. The target variable is continuous, and the aim is to develop a polynomial regression model that can accurately predict this target variable based on the input features.

## Implementation Details

### 1. Data Preprocessing

* **Loading Data:** The dataset was loaded using pandas, and preliminary exploratory data analysis was conducted to understand the distribution of features and the target variable.



* **Feature Scaling:** Feature scaling was applied to normalize the features, ensuring consistency in their impact on the model.
* **Train-Test Split:** The dataset was split into training and testing sets for model evaluation.

### 2. Polynomial Regression Model

A polynomial regression model was implemented with the following steps:

* **Initialization:** Model parameters (coefficients and bias) were initialized.
* **Cost Function:** The mean squared error (MSE) cost function was used to quantify the difference between predicted and actual values.
* **Gradient Descent:** Gradient descent was employed to iteratively update model parameters and minimize the cost function.
* **Multinomial features:** The code was run for degrees 2,3,4,5,6 and 7. The cost for cross validation set was minimum for degree 6. Hence polynomial of degree 6 was chosen.
* **Regularization:** Regularization was used to prevent overfitting.

### 3. Training and Evaluation

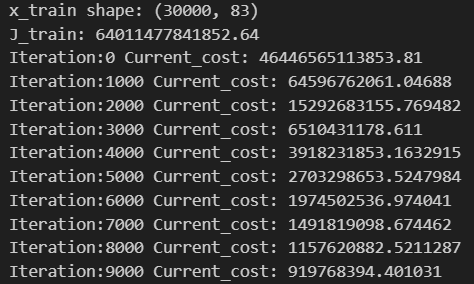
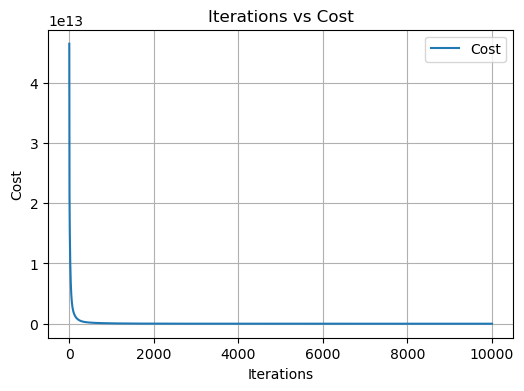
The model was trained on the training dataset, and performance was evaluated on the testing set using appropriate evaluation metrics such as mean squared error.

### 4. Hyperparameters

Alpha, Number of Iterations:

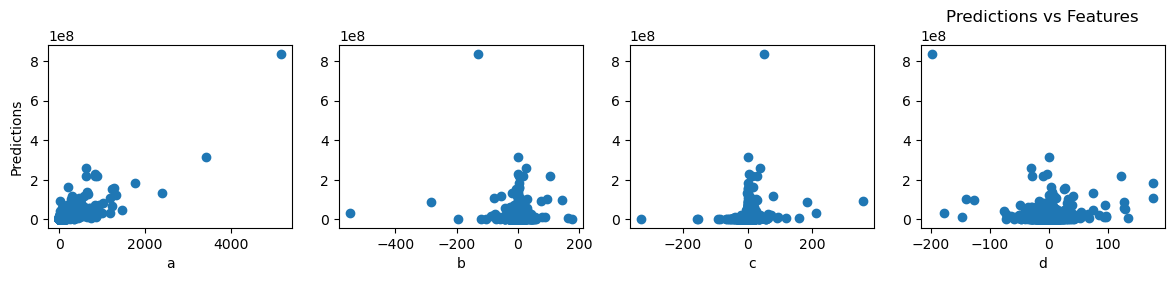
I tried various different values for alpha. Here’s what I observed

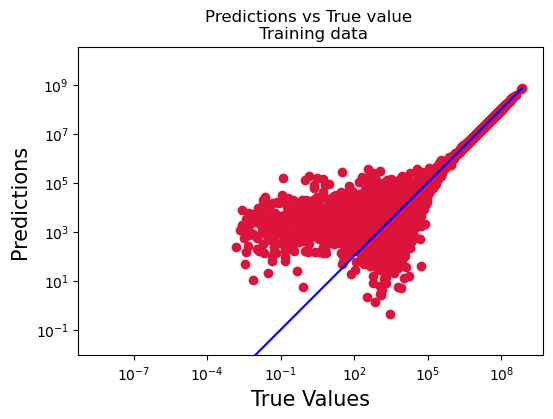
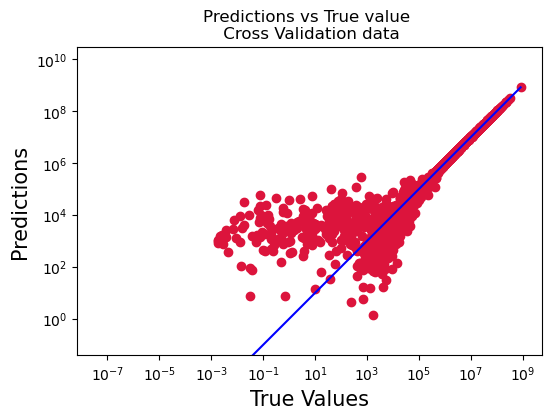
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Degree2 | Degree3 | Degree4 | Degree5 | Degree6 | Degree7 |
| Alpha | 0.01 | 0.01 | 0.001 | 0.0001 | 0.00001 | 0.000001 |
| Number of Iterations | 4000 | 9000 | 10000 | 10000 | 10000 | 10000 |

## Results and Analysis

The model's performance was analysed based on evaluation metrics. Visualizations, including polynomial regression curves and predicted vs. actual plots, were used to interpret the model's behaviour.



## Challenges and Future Work

* **Overfitting:** Polynomial regression models are prone to overfitting, especially with high-degree polynomials. Regularization and hyperparameter tuning can be explored to address this challenge.

# Logistic Regression with Multiclass Classification (One-vs-All)

## Introduction

The goal of this project was to implement logistic regression for multiclass classification using the one-vs-all (OvA) method. Logistic regression is a fundamental algorithm for binary classification, and the OvA approach extends it to handle multiple classes.

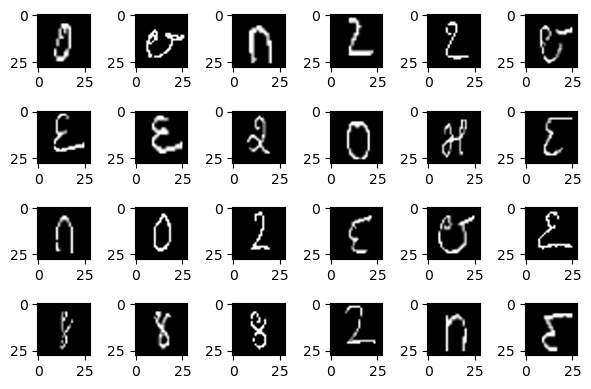
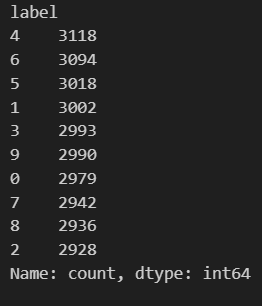
## Dataset Description

The provided dataset consists of 30,000 samples, each containing 784 features. These features represent pixel values of images. The target variable is categorical, representing different classes. The objective is to develop a logistic regression model capable of classifying the input images into multiple classes.

## Implementation Details

### 1. Data Preprocessing

* **Loading Data:** The dataset was loaded using pandas, and reshaped into a suitable format for logistic regression.

* **Train-Test Split:** The dataset was split into training and testing sets to evaluate the model's performance on unseen data.

### 2. Logistic Regression Model (One-vs-All)

A logistic regression model for multiclass classification was implemented using the one-vs-all approach:

* In the start I created 10 categories in each category one label is labelled as 0 and rest other labels are given label 1
* **Initialization:** Model parameters (weights and bias) were initialized for each class.
* **Cost Function:** The cross-entropy loss function was used to quantify the difference between predicted probabilities and actual class labels.
* **Gradient Descent:** Gradient descent was employed to iteratively update model parameters and minimize the cost function for each class.
* Lastly I run the gradient descent for each category then computed cost respectively, and then the category whose cost was maximum was selected in this case that is category 7

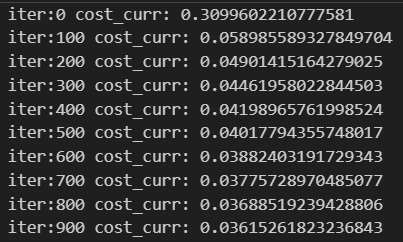
### 3. Training and Evaluation

The model was trained on the training dataset using the OvA logistic regression approach. Performance was evaluated on the testing set using metrics such as accuracy, precision, recall, and F1 score.

### 4. Hyperparameters

Alpha:  
Alpha values for each category is same i.e 0.00001.

Number of Iterations:  
1000 for all the categories.

****This is iteration vs cost for category 6

## Results and Analysis

The model's performance was analysed based on evaluation metrics. Confusion matrices and classification reports were generated to provide a detailed understanding of how well the model performed for each class.

Accuracy = 98.022%

## Challenges and Future Work

* **Hyperparameter Tuning:** Experimentation with learning rates and regularization strengths may improve the model's generalization capabilities.

# K-Nearest Neighbours

## Introduction

The aim of this project was to implement the k-Nearest Neighbours (KNN) algorithm for multiclass classification using a dataset with dimensions 30,000 x 784. KNN is a simple and effective non-parametric algorithm that makes predictions based on the majority class among its k-nearest neighbours.

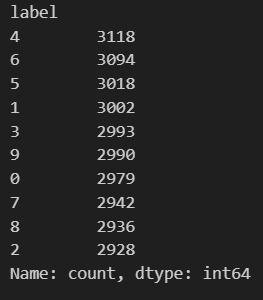
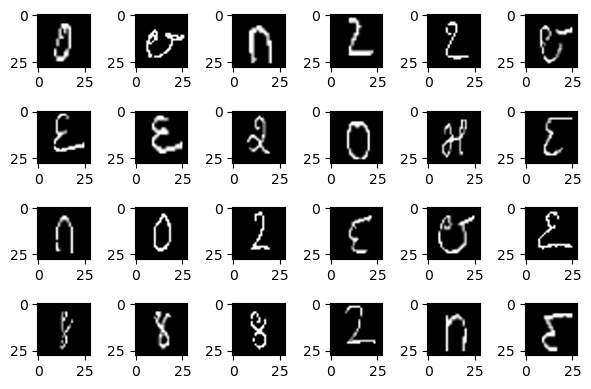
## Dataset Description

The dataset provided consists of 30,000 samples, each containing 784 features. These features represent pixel values from a 28x28 image, and the target variable is a class label indicating the digit represented by the image. The objective is to implement and evaluate the KNN algorithm for accurate digit classification.

## Implementation Details

### 1. Data Preprocessing

* **Loading Data:** The dataset was loaded into the program, and basic exploratory data analysis was conducted to understand the structure of the features and labels.



* **Train-Test Split:** The dataset was divided into training and testing sets for model evaluation.

### 2. K-Nearest Neighbours Algorithm

The KNN algorithm was implemented with the following steps:

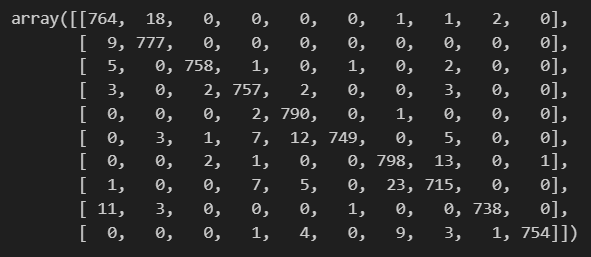
* **Distance Metric:** Euclidean distance was used as the distance metric to measure the similarity between data points.
* **K-Nearest Neighbours:** For each test data point, the k-nearest neighbours were identified based on the smallest Euclidean distances.
* **Majority Voting:** The majority class among the k-nearest neighbours was assigned as the predicted class for the test data point.

### 3. Training and Evaluation

The KNN model was trained on the training dataset, and its performance was evaluated on the testing set using metrics such as accuracy.

## Results and Analysis

The model's performance was analysed based on evaluation metrics and visualized using confusion matrices. The impact of different values of k on model performance was also explored.

  
This is a confusion matrix for k=3

Accuracy = 98%

# Neural Network

## Introduction

The purpose of this project was to implement a neural network for multiclass classification using a dataset with dimensions 30,000 x 784. Multiclass classification involves predicting the class labels of instances when there are more than two possible classes.

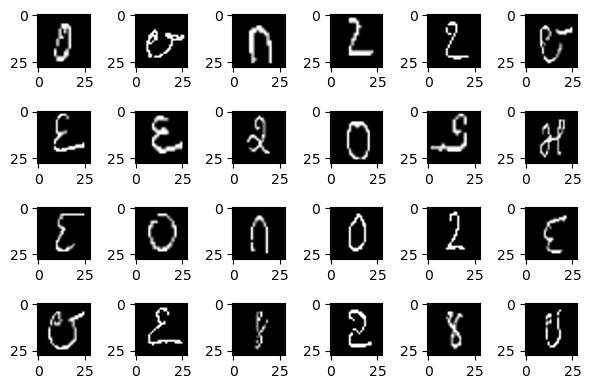
## Dataset Description

The dataset consists of 30,000 samples, each containing 784 features. These features represent pixel values of images. The task is to classify these images into multiple classes based on the provided labels. The dataset was pre processed to ensure compatibility with the neural network architecture.

## Implementation Details

### 1. Data Preprocessing

* **Loading Data:** The dataset was loaded into the program, and initial exploratory data analysis was conducted to understand the characteristics of the images.



* **Label Encoding:** Class labels were encoded using one-hot encoding to handle the multiclass classification problem.
* **Train-Test Split:** The dataset was split into training and testing sets for model evaluation.

## 2. Neural Network Architecture

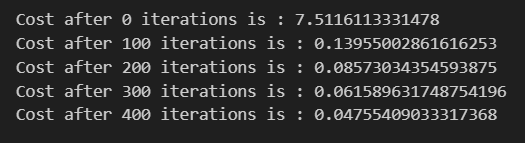
The neural network architecture was designed with the following components:

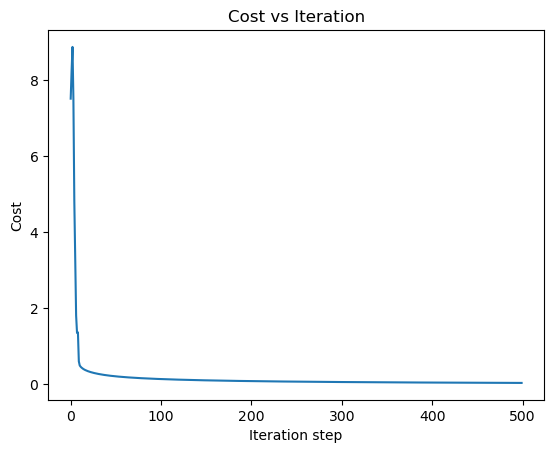
* **Input Layer:** 784 neurons representing the pixel values of each image.
* **Hidden Layers:** One or more hidden layers with a variable number of neurons, activated using ReLU (Rectified Linear Unit) activation function.
* **Output Layer:** The output layer with neurons equal to the number of classes, activated using the softmax function for multiclass classification.
* **Loss Function:** Categorical Cross-Entropy loss was chosen as the appropriate loss function for multiclass classification.
* **Optimization:** Stochastic Gradient Descent (SGD) or other optimization algorithms were employed for model training.

## 3. Model Training and Evaluation

The neural network was trained on the training dataset with the following steps:

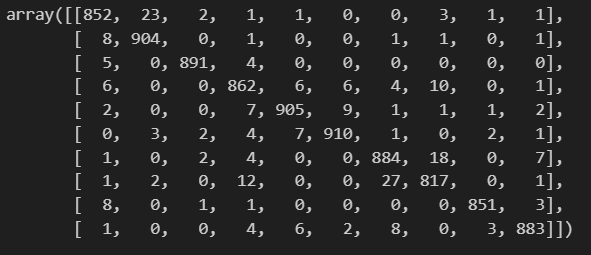
* **Forward Propagation:** Input data was passed through the network to generate predictions.
* **Backward Propagation:** Gradients were computed using backpropagation to update the model parameters.
* **Training Metrics:** Training metrics such as accuracy and loss were monitored during the training process.
* **Model Evaluation:** The trained model was evaluated on the testing set to assess its generalization performance.





## Results and Analysis

The performance of the neural network was analysed based on various metrics, including accuracy and confusion matrix. Visualization techniques, such as learning curves and confusion matrices, were utilized to interpret the model's behaviour.

  
Confusion Matrix

Accuracy of Train dataset = 99.2%  
Accuracy of Validation dataset = 97%

## Challenges and Future Work

* **Hyperparameter Tuning:** Experimentation with different hyperparameter values, including the number of hidden layers, neurons per layer, and learning rates, could further optimize model performance.

# K-means Clustering

## Introduction

The purpose of this project was to implement the K-means clustering algorithm from scratch using a dataset with dimensions 178 x 13. K-means clustering is an unsupervised machine learning algorithm used for partitioning a dataset into K distinct, non-overlapping subsets (clusters).

## Dataset Description

The dataset provided contains 178 samples, each having 13 features. These features represent various attributes related to the characteristics of certain items. The objective is to group these items into K clusters based on the similarity of their feature vectors.

## Implementation Details

### 1. Data Preprocessing

* **Loading Data:** The dataset was loaded into the program, and basic exploratory data analysis was conducted to understand the nature and distribution of the features.
* **Feature Scaling:** Feature scaling was applied to normalize the features, ensuring that all features contribute equally to the clustering process.

### 2. K-means Algorithm

The K-means clustering algorithm was implemented with the following steps:

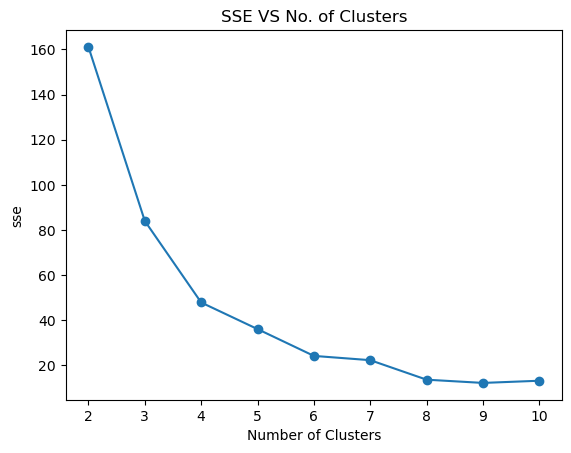
* **Initialization:** K initial centroids were randomly chosen from the dataset.
* **Assignment Step:** Each data point was assigned to the nearest centroid, forming K clusters.
* **Update Step:** The centroids of the clusters were updated based on the mean of the data points belonging to each cluster.
* **Convergence:** Steps 2 and 3 were repeated until convergence, where the centroids no longer change significantly or a predetermined number of iterations is reached.

### 3. Training and Evaluation

The model was trained on the entire dataset, and the resulting clusters were evaluated based on their coherence and separation. Evaluation metrics such as the sum of squared distances from each point to its assigned centroid were considered.

## Results and Analysis

The clustering results were analysed in terms of the formed clusters and their characteristics. Visualization techniques, including scatter plots and silhouette analysis, were employed to interpret the clustering performance.



## Challenges and Future Work

* **Initialization Sensitivity:** K-means is sensitive to the initial choice of centroids. Experimentation with different initialization methods can be conducted to address this issue.