# Project Report

## Machine Learning Division


NAME:

**DIVYANSH GUPTA**

Adm. No.:

**21JE0319**

**CONTACT**

Phone:
+91-8416872935

Institute e-mail:
21je0319@iitism.ac.in

Personal e-mail:
divyanshgupta351@gmail.com

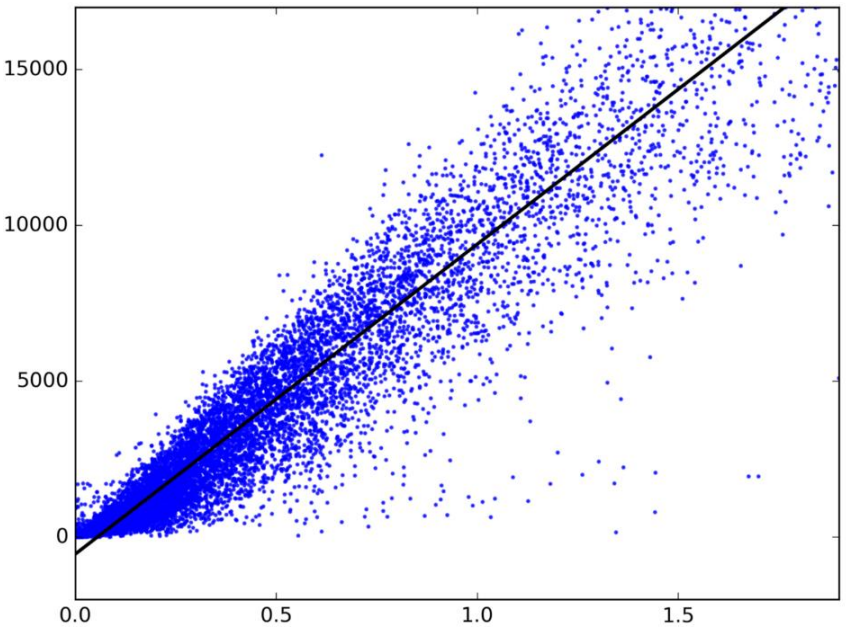

## Personal Background:

College: **IIT (ISM) Dhanbad**
Major: **Computer Science and Engineering**
Year of Study: **1st Year**
Place: **Gorakhpur, Uttar Pradesh, India**
Time zone: **GMT +5:30**
Linkedin: Redirect to Profile
GitHub: Redirect to Profile

**Project**: ML Bootcamp
**Mentor**: Kushagra Bhushan sir and Aryamaan Thakur sir

# Report

## Linear Regression:



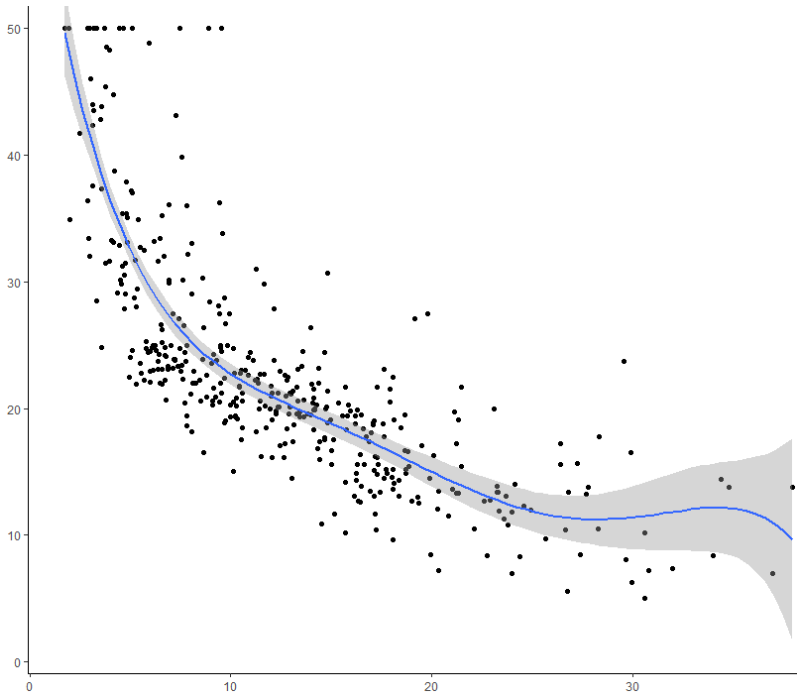

It is the ML model in which the aim is to fit a line (2D data) or plane (3D data) or some linear function to a given dataset and make predictions on the test set with this linear function. I coded two models for the same, the gradient descent model and the normal model. The gradient descent model had an R2 score of 0.99999977 while the normal model had an R2 score of 0.99999986.

Comparison of different metrics over both models:

| Metric/Model | **Normal Model** | **Gradient Descent Model** |
|---|---|---|
| **Mean Absolute Error:** | 1.4716 | 1.9154 |
| **Mean Squared Error:** | 3.3441 | 5.5927 |
| **Root Mean Squared Error:** | 1.8287 | 2.3649 |
| **r2 score:** | 0.99999986 | 0.99999977 |

We can see that the normal model outperforms the gradient descent in every metric, this is because gradient descend cannot always converge to the absolute minima for the function. It goes to the approximate minima and keeps on fluctuating in a very small range near the absolute minima while normal model calculates the absolute minima in one go.

## Polynomial Regression:



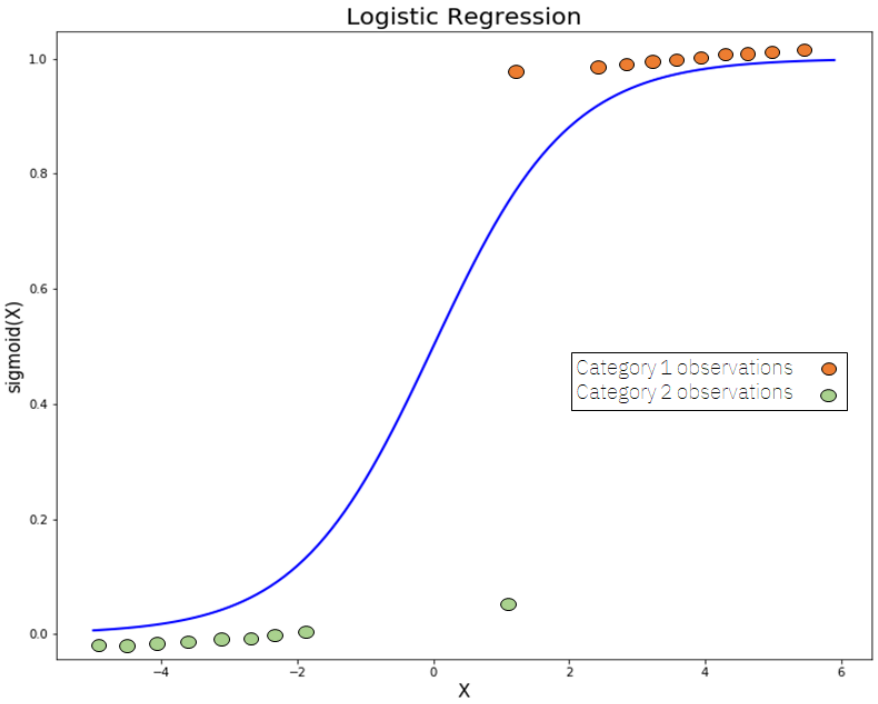

In polynomial regression the aim is to fit a polynomial instead of linear equation to the given dataset. Here also we can use both gradient descent and normal model but I used just the normal model as the dataset was small and normal model works best on small dataset. In polynomial regression I used linear features, quadratic features and cubic features as suggested by mentors. Comparison of the three is listed on the next page.

| Metric/Degree | Linear | Quadratic | Cubic |
|---|---|---|---|
| **Mean Absolute Error:** | 876.1379 | 174.9409 | 175.0421 |
| **Mean Squared Error:** | 1454181.6214 | 54378.0816 | 54396.7662 |
| **Root Mean Squared Error:** | 1205.8945 | 233.1911 | 233.2311 |
| **r2 score:** | -0.00032883 | 0.96259342 | 0.96258057 |

Here we can see that the quadratic and cubic features are almost performing same and outperforming the linear by a great margin in all aspects.

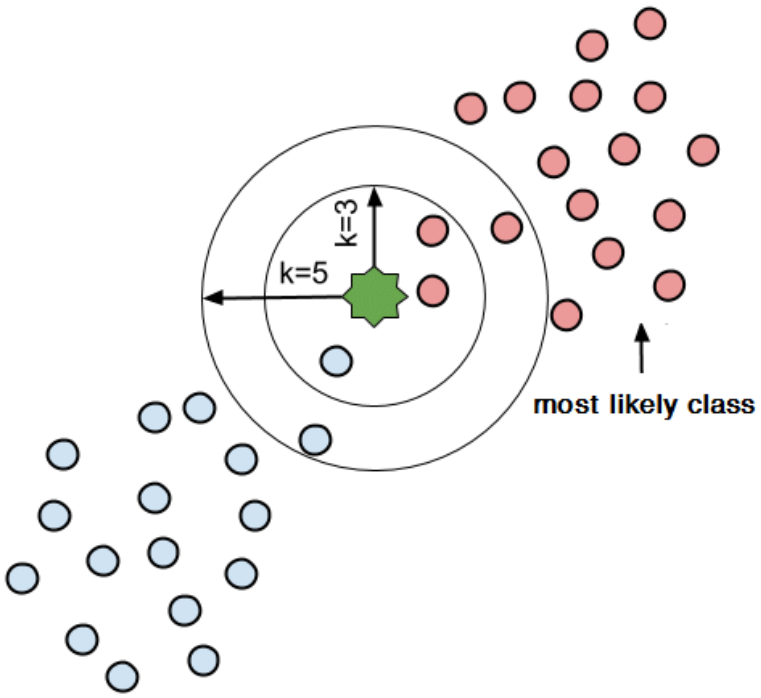
# Logistic Regression:

Logistic regression is used in classification (binary/One vs all). I coded the algorithm of logistic regression using the basic techniques as discussed in the Andrew NG's course and had achieved an accuracy of 65.26% over the test set in 1000 iterations with learning rate 1.2. The accuracy could have been increased if I had increased the number of iterations or used vectorization (suggested by mentors). I could not use vectorization as I didn't have much knowledge about it.



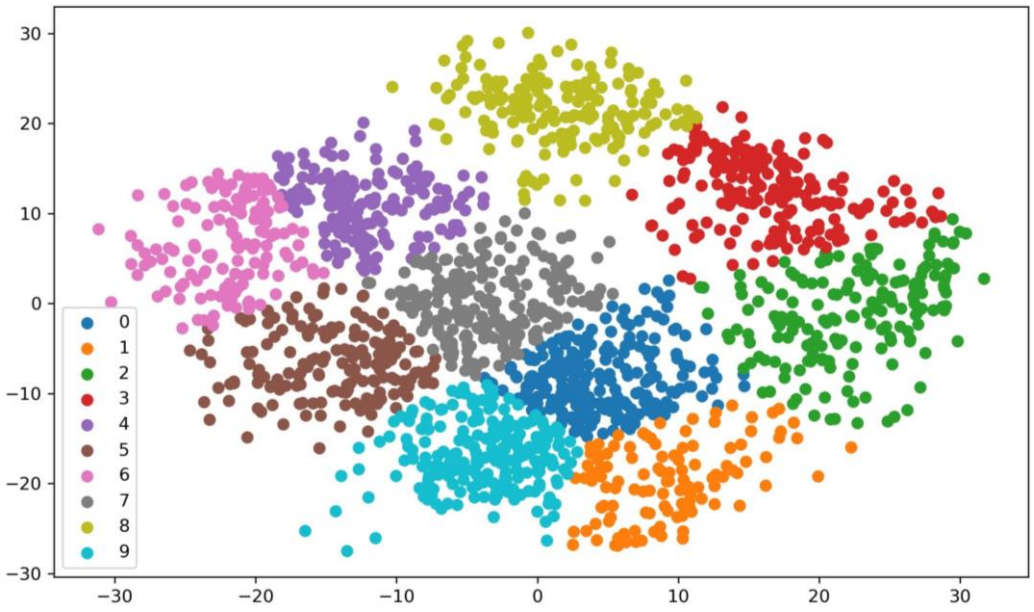

# K-Nearest Neighbours:

KNN is a supervised ML algorithm used for classification. It classifies datapoints based on how vits neighbours are classified. It stores all available cases ands classifies new cases based on similarity measure. Here K denotes the number of neighbours to be checked for classifying the datapoint to any class. I coded its algorithm and got an accuracy of 85.99% over the test set for K = 10. The major flaw in this model is that it is too slow as it stores the complete training set and checks all examples in the training set before classifying a datapoint.



# K-Means Clustering:

K-Means clustering is an unsupervised learning algorithm. In this the datapoints are put into different clusters based on distance from the centroid of the cluster. I coded the algorithm for K-Means Clustering and ran it over the test set only as it had sorted data making visualization easier. I calculated the accuracy manually which came out to be 45.86% but since manual accuracy calculation may have some error it can be safely assumed to be something around 35%.




# Neural Network:

As the name suggests it is an algorithm that tries to mimic the human brain. It can have multiple layers including an input and an output layer. It is used in classification problems. Each layer has an activation function, the activation of a layer is dependent on a layer just before it. I coded the algorithm for two layered neural network. It had an accuracy of 80.85% over the training set in 3000 iterations and learning rate of 0.5. It had 77.68% accuracy over the test set.