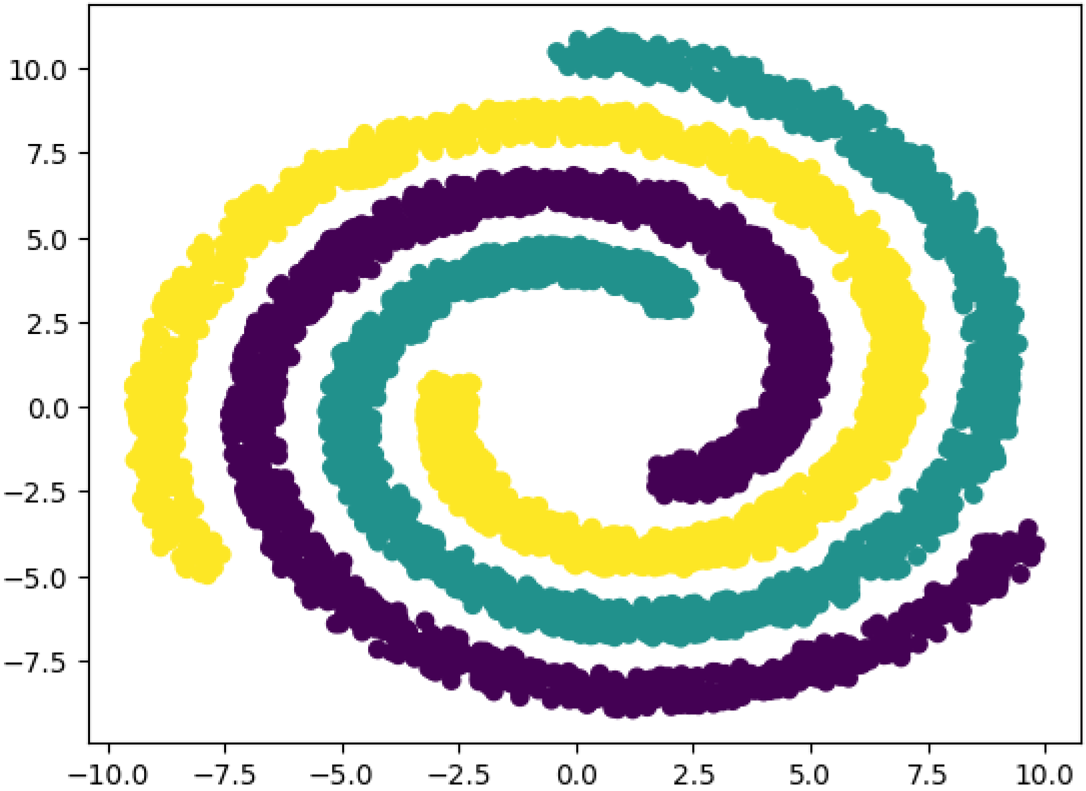
REPORT

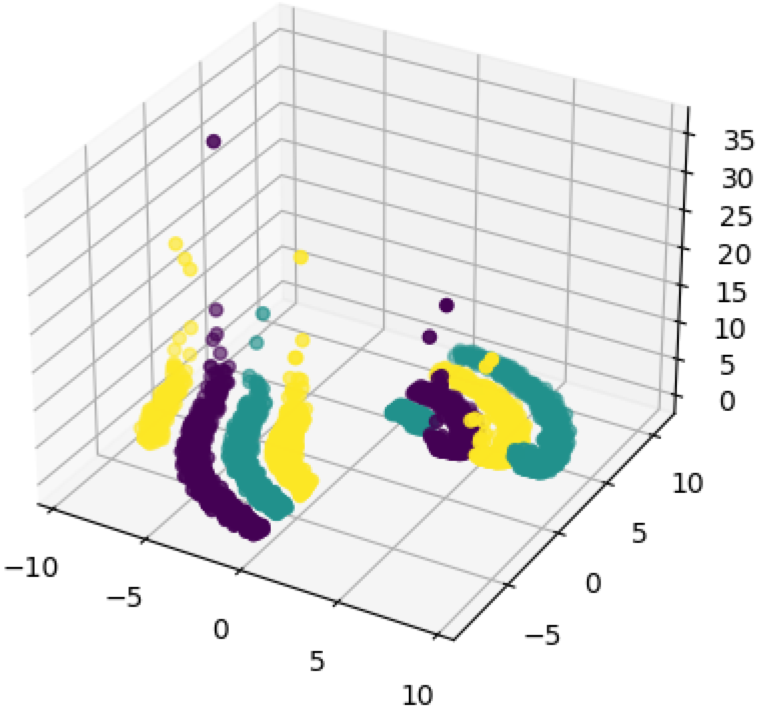
Kashi Vishwanath Bondugula
kbondugu@stevens.edu

1. Data
    a. There were no null or Nan values in the dataset
    b. The given training data set is not linear, and circular / spiral in shape
    c. The labels are more than two and for logistic regression it is not recommended for data apart from binary data.
    d. In order to convert the data into binary, I am using "**One versus Rest**" approach, I am setting class-2 and class-3 as class-2 and class-1 as 1
    e. I have tried to transform the data using multiple math equations but in the end I couldn't convert the data into linear fashion.
    f. 

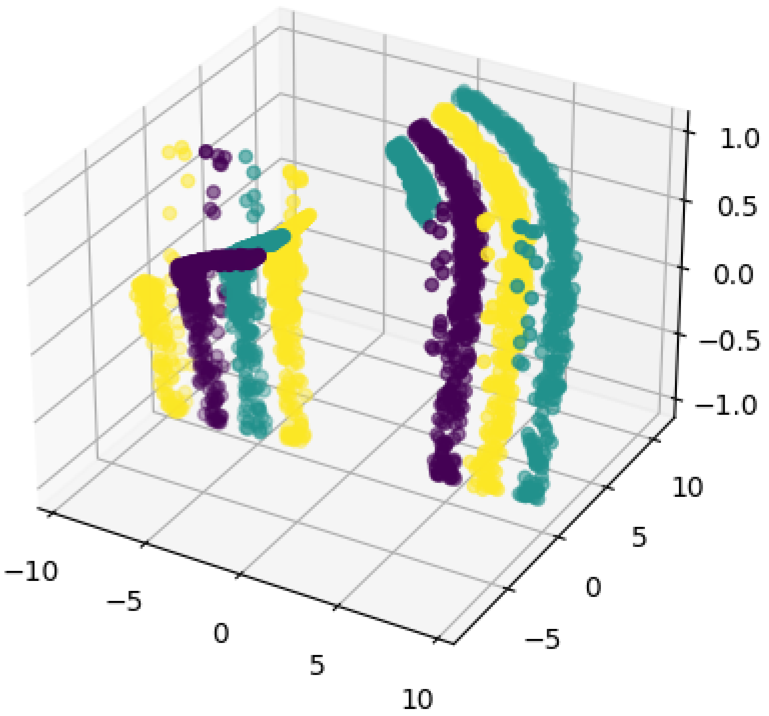
2. Naïve Logistic Regression
    a. I have implemented a Naive Logistic Regression using numpy, sigmoid function as activation function with threshold as 0.5.

   b. I have adjusted the weights according to the learning rate and calculated dw
   c. By default I am considering the learning rate as 0.001 and n_iterations as 1000
   d. Maximum accuracy I have received is 0.6570666666666667

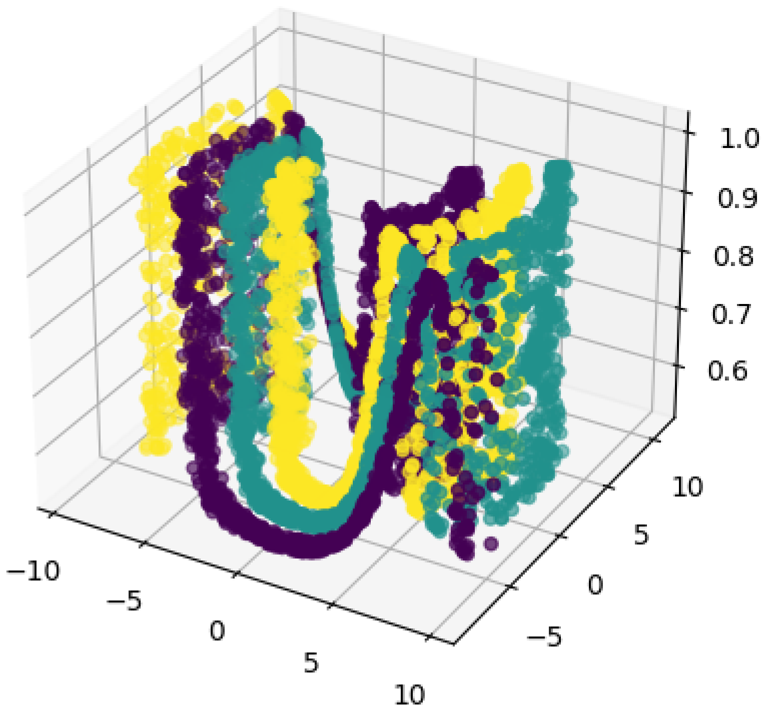
3. Data Transformations
   a. z = np.sqrt(np.log10(np.exp(X_train[:, 0])) / np.log10(np.exp(X_train[:, 1])))



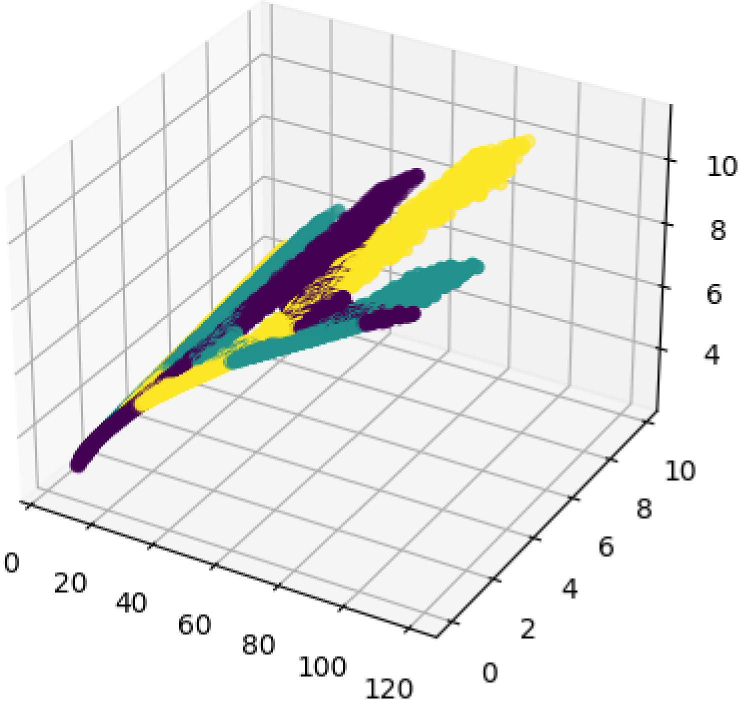
b. z = np.cos(np.sqrt(np.log10(np.exp(X_train[:, 0])) / np.log10(np.exp(X_train[:, 1]))))



c. z = np.cos(np.cos(np.square(np.log10(np.exp(X_train[:, 0])) / np.log10(np.exp(X_train[:, 1])))))



d. Finally I ended up on a transformation using few examples from textbook, where I have ended up converting cartesian plots to polar values and then taking took the square root of the sum of x1 and x2

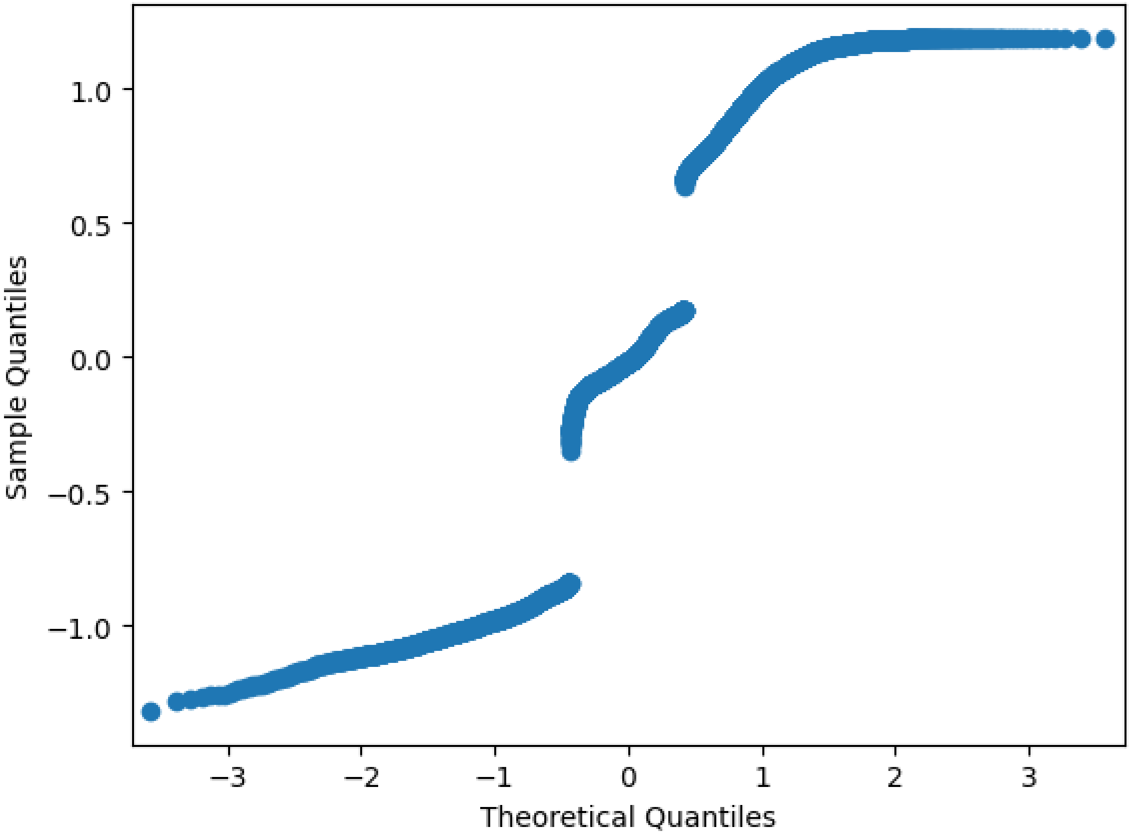


e. I have constructed a new datapoint or dimension x3 with which I tried to convert the non-linear data into linear data.

f. For the transformed data we need to check for **linearity, homoscedasticity, absence of multicollinearity, independence and normality of errors**

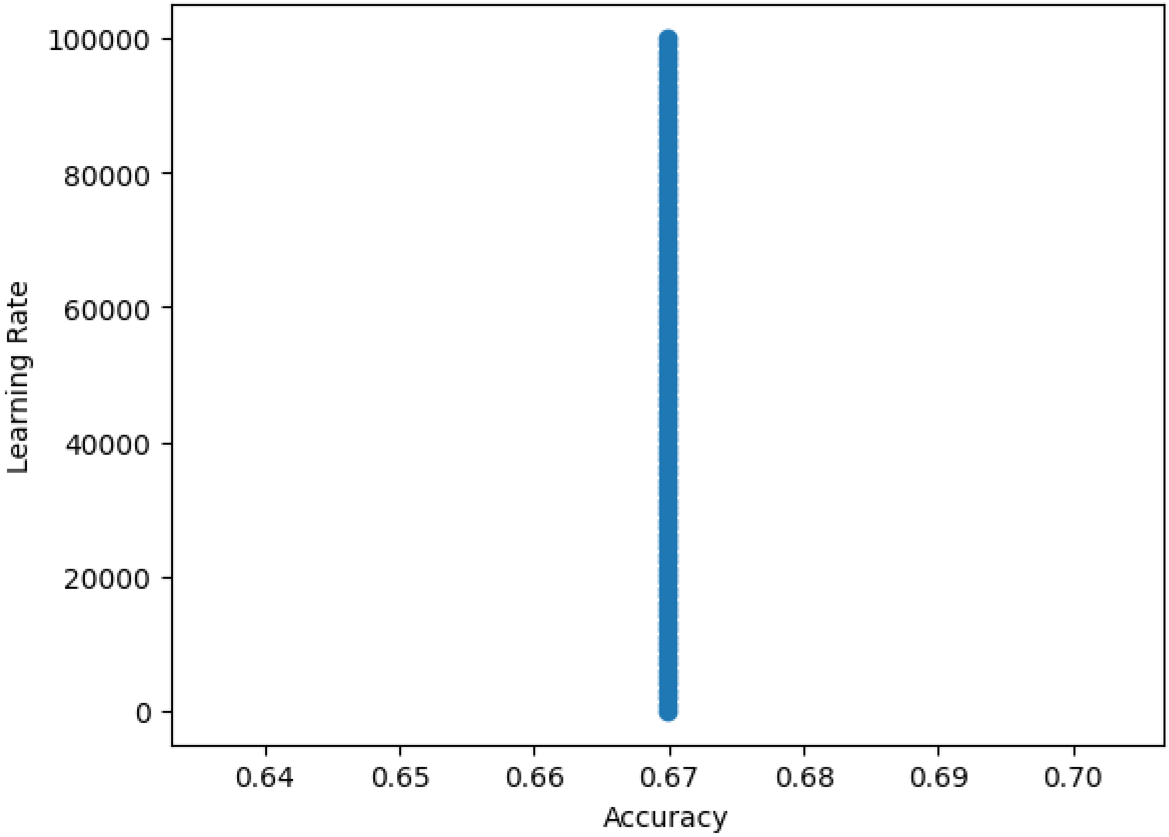
    i. For the linearity I have looked at residual vs fitted plot

ii. For the normality I have considered qqplot from statsmodels library



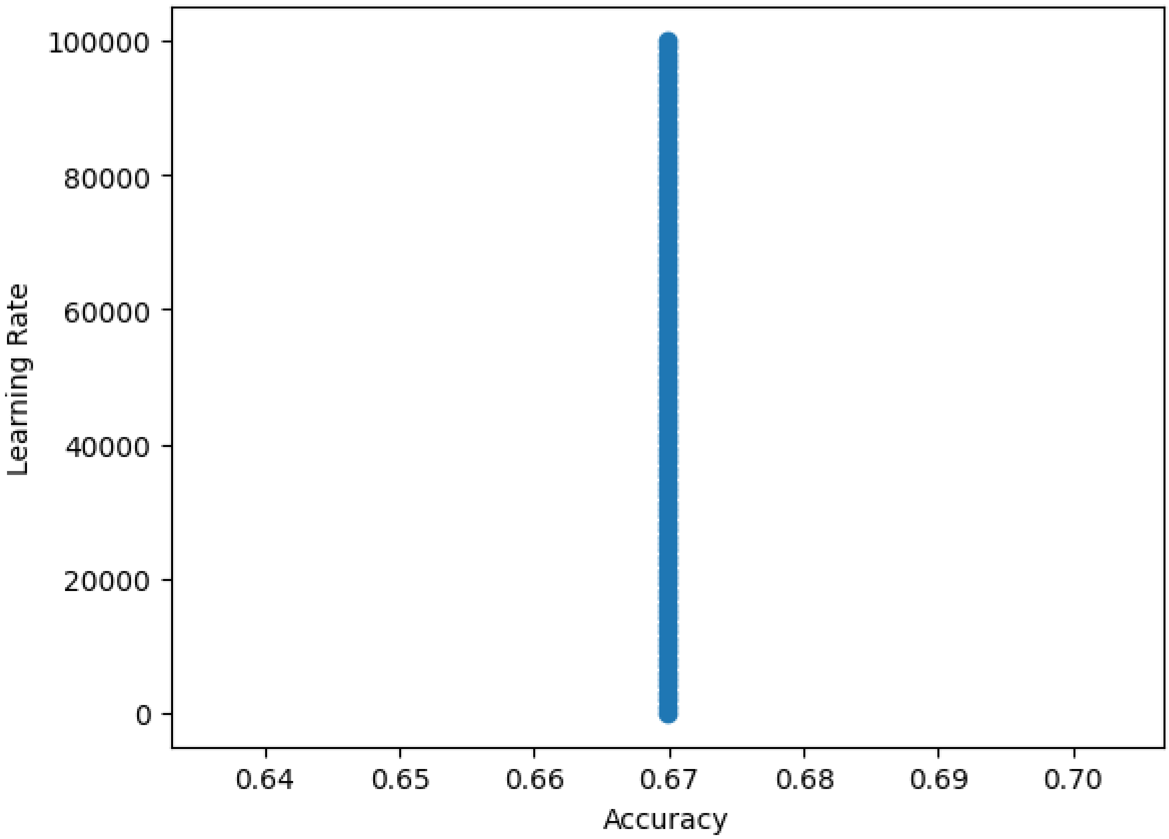
4. For improving the accuracy of the logistic regression model, I have used **GridSearchCV** at various learning rates.
    a. The best accuracy I have achieved using GridSearchCV is 0.6698666666666667

b. The best model is found for learning rate = 1.0



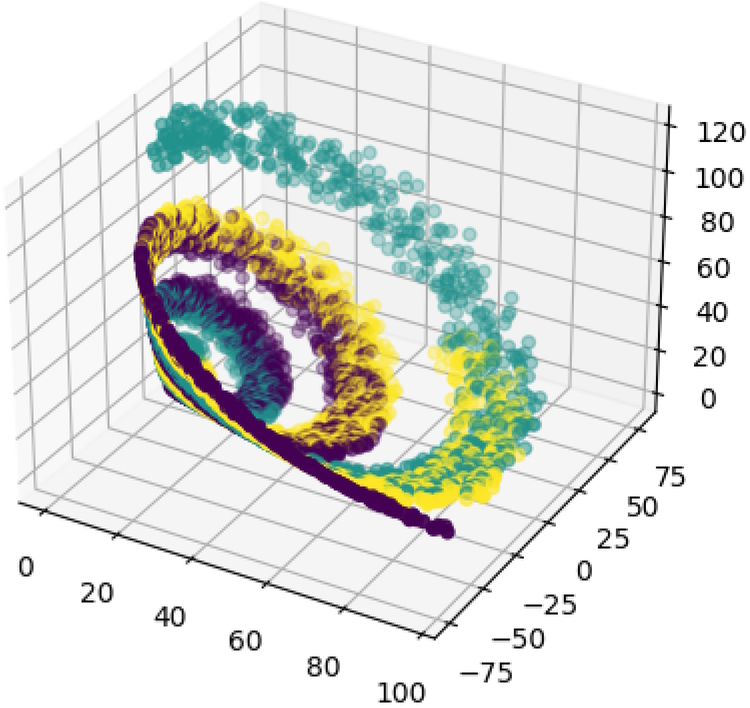
c. The major thing I have noticed is irrespective of the learning rate the accuracy of the model didn’t really change that much, it was almost a constant

d. Things were the same with one of my best transformations of data as well, accuracy didn’t really change irrespective of the learning rate.

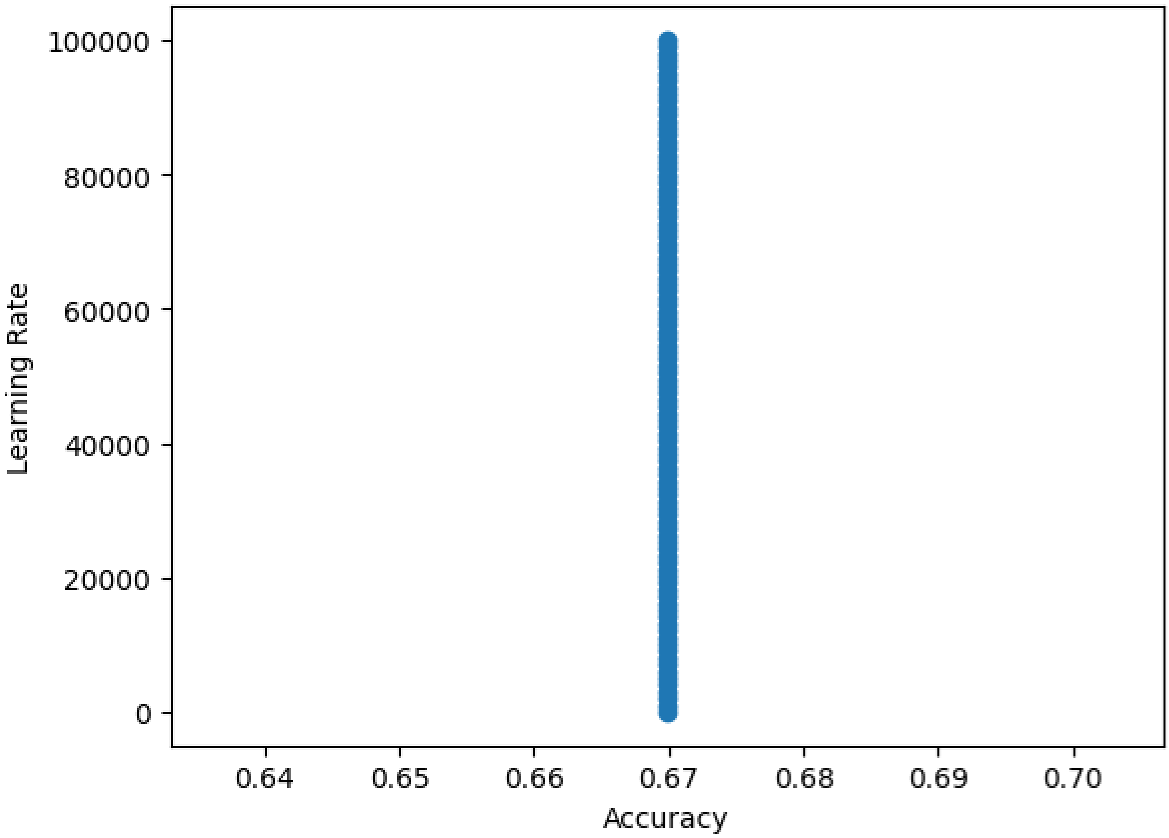


5. Kernelization: a technique for designing efficient algorithms that achieve their efficiency by a preprocessing stage in which inputs are mapped to higher dimensions without the calculation overhead
    a. I have implemented five different kernel functions to transform the data to achieve better results
    b. **Gaussian Kernel:** For gaussian kernel I have mapped both training data and test data into the new gaussian kernel space and achieved a 3-D data from the original 2-D data

c. The transformed data looks something like this

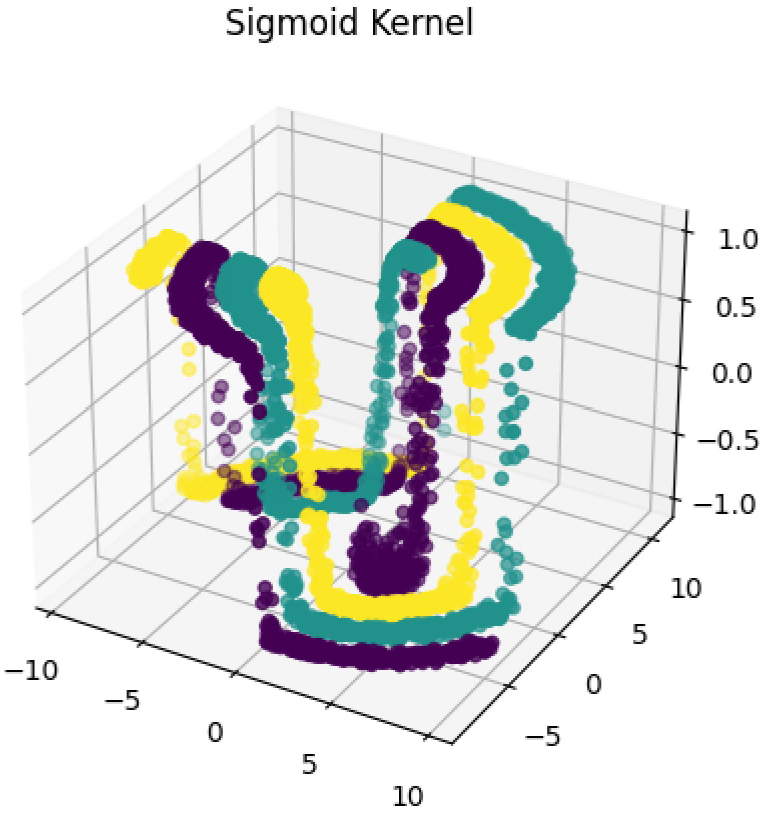


d. Accuracy didn't change by much as well


100000
80000
60000
40000
20000
0
Learning Rate
0.64
0.65
0.66
0.67
0.68
0.69
0.70
Accuracy

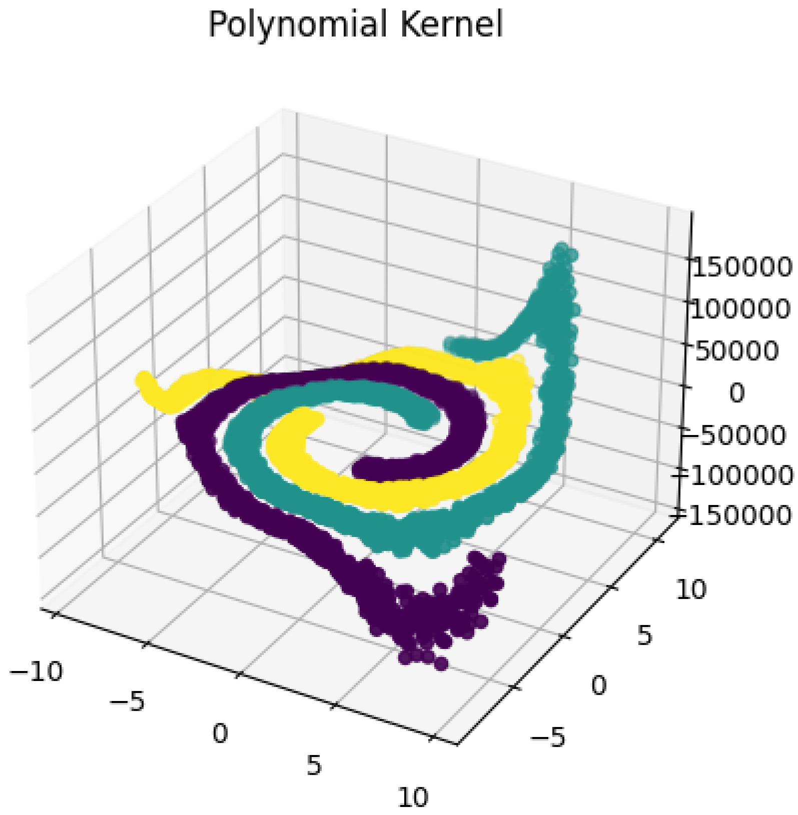
e.

f. **Sigmoid Kernel**: I have implemented the sigmoid kernel using tanh, post transformation the shape of the data has drastically changed



g. But there wasn’t any change in the accuracy because of the non-linearity in the data.

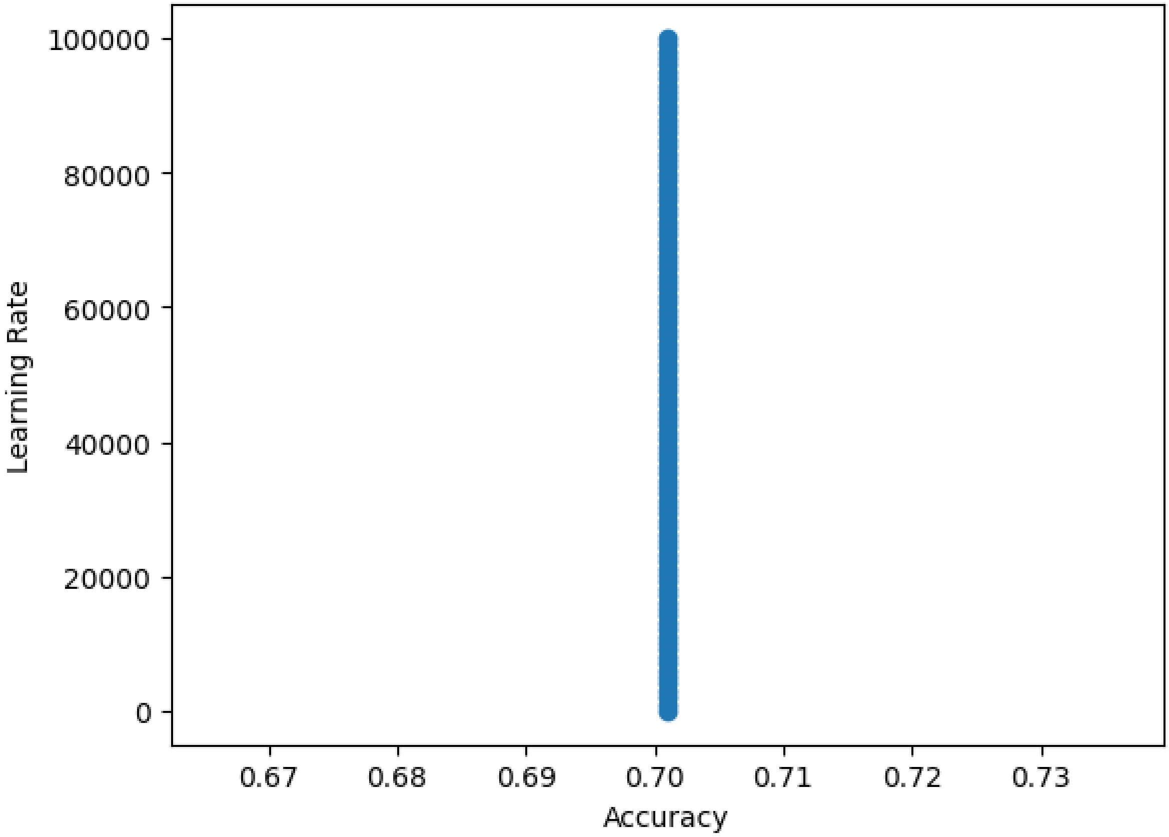
h. **Polynomial Kernel:** I have taken a polynomial of degree 3 and using dot product I have generated the third dimension x3 with which the data is being transformed into polynomial kernel



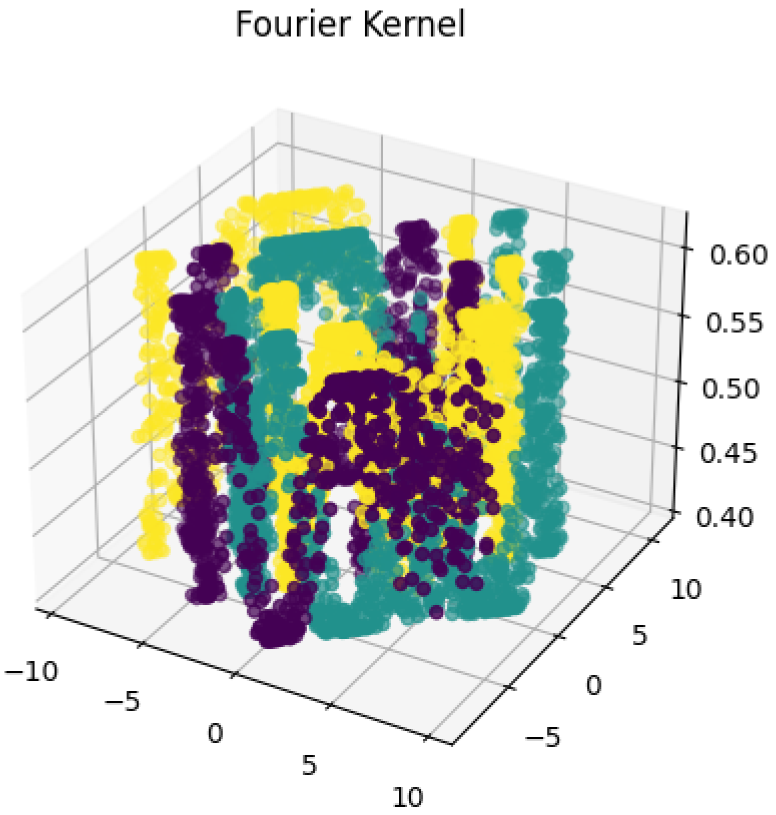
Polynomial Kernel

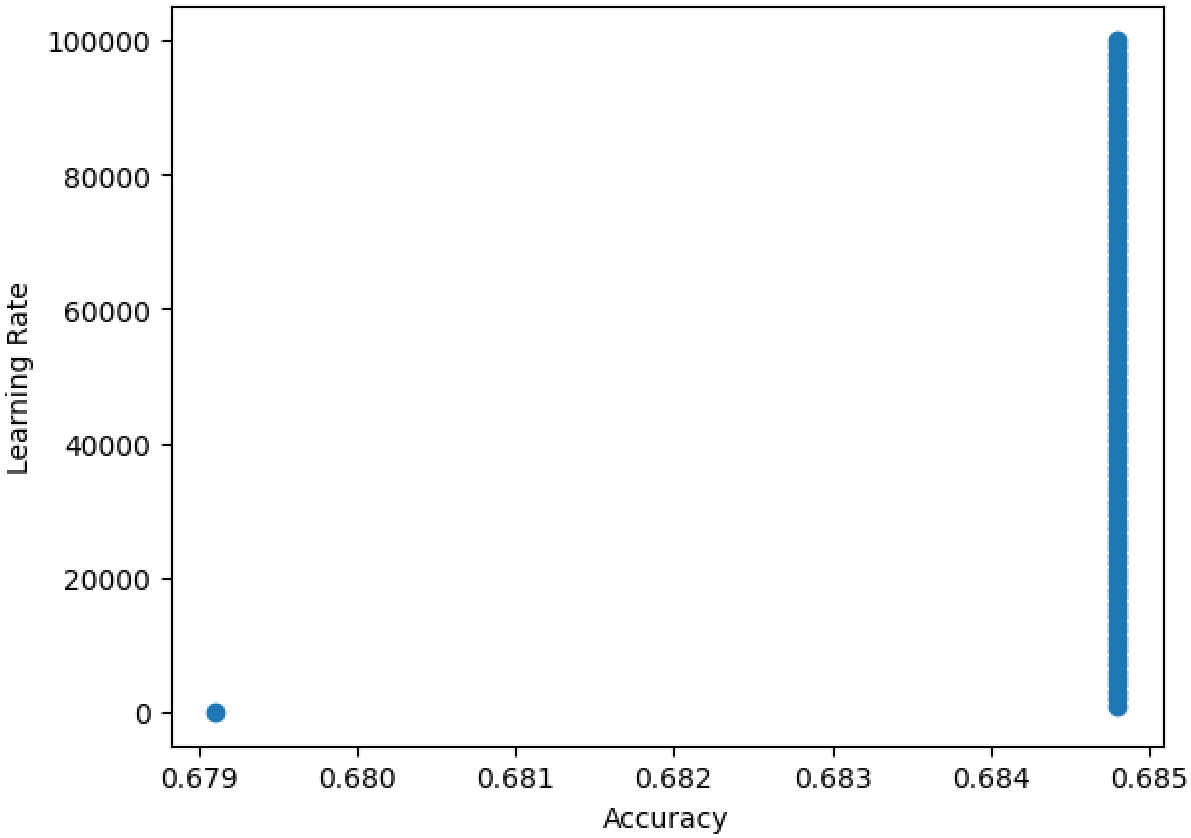
space

i. However the separation of data is not much but it did affect the behavior of the data and the accuracy of the model has increase by quite a bit to almost 71%



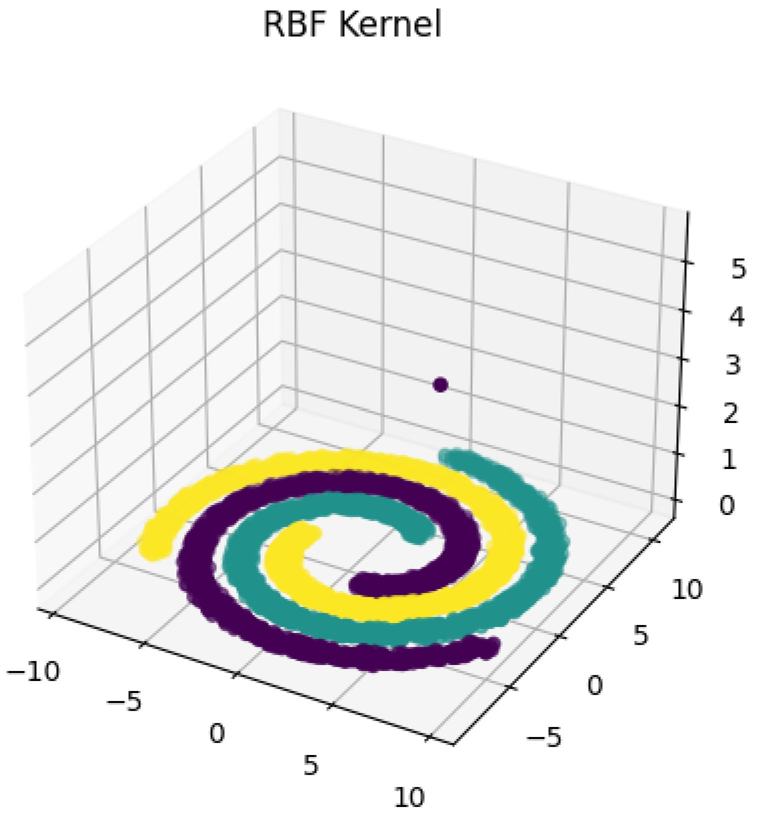
j. **Fourier Kernel**: I have implemented the fourier kernel with the help of cos function and _q as 0.1, this was an interesting transformation which reduced the accuracy instead of increasing it





k.

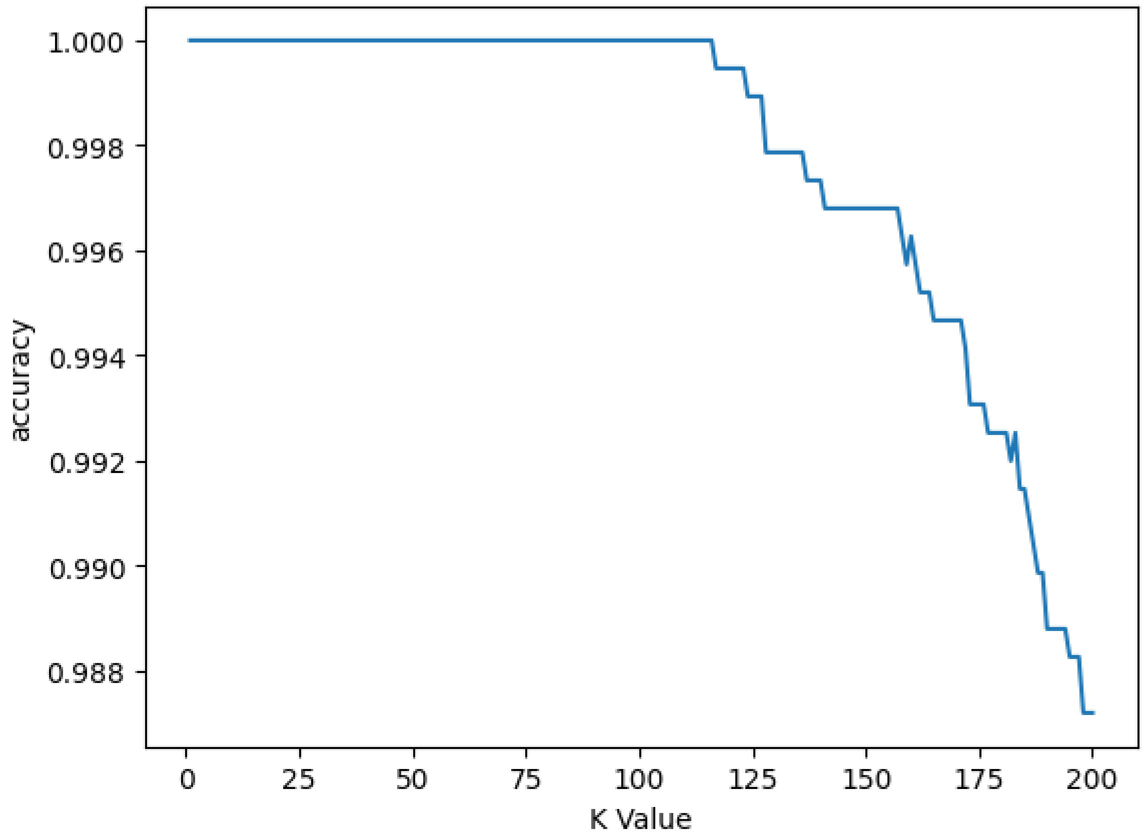
l. **RBF Kernel:** I have implemented the RBF kernel using the exponent of the square of difference original inputs multiplied by a gamma value.



m. This transformation failed to transform the data even though we converted it into high dimensionality

6. **Non-parametric KNN Classification**
    a. Right out of the box k-nearest neighbor classification worked flawlessly on the data with accuracy of 100% without any transformation of data.
    b. However since the accuracy was 100% which indicates the model is overfitted I have increased the K-value which resulted

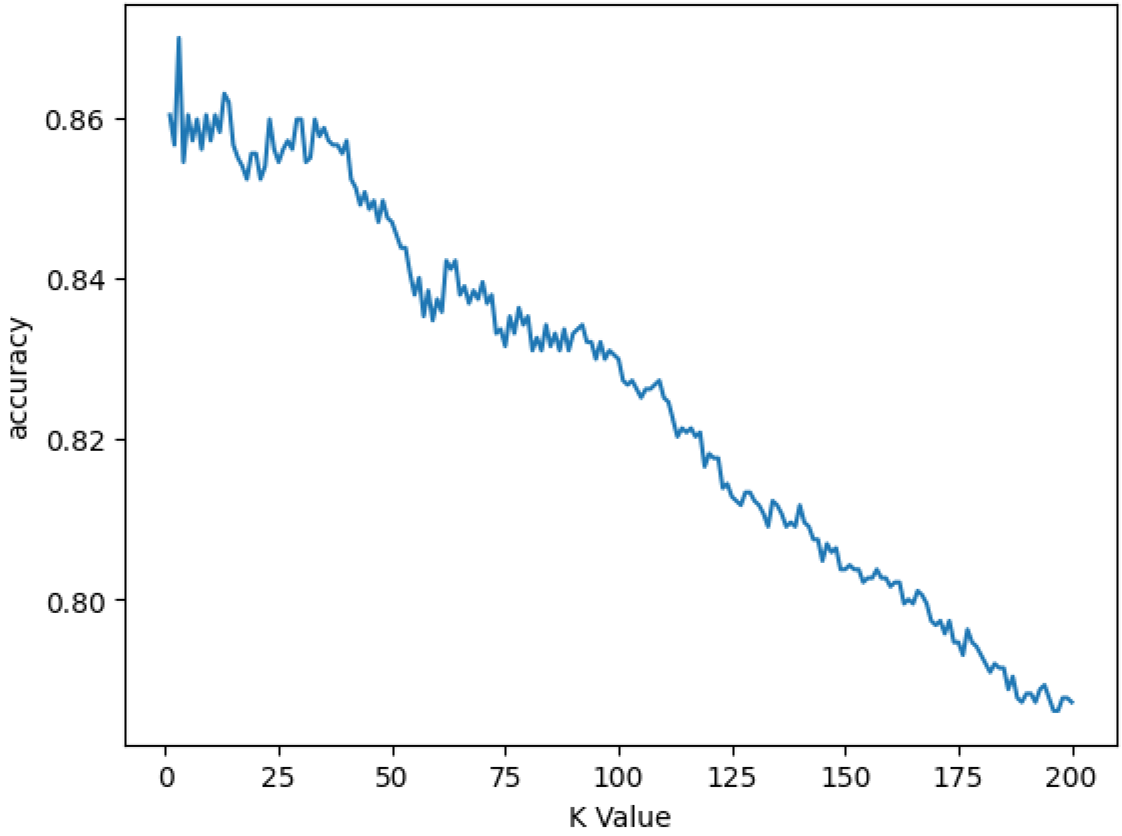
in an accuracy of 98%



c. There was a steady decline of the accuracy with the increase in k-value

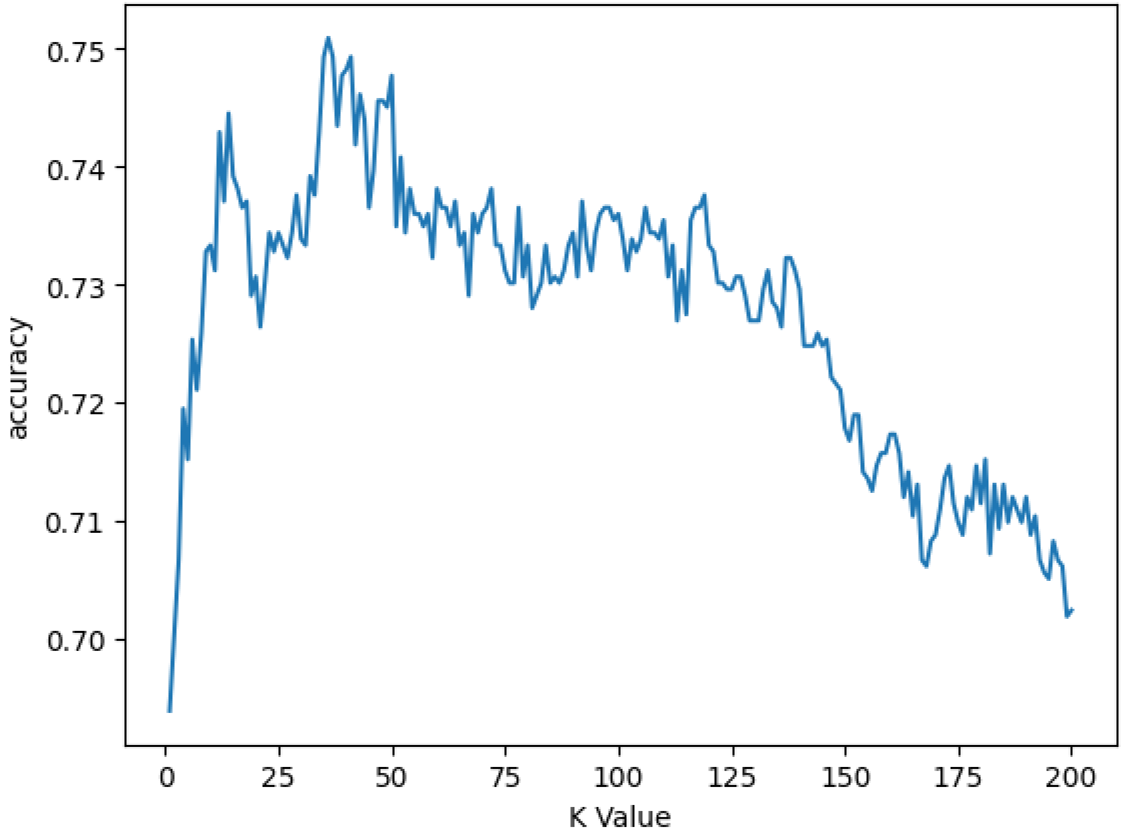
7. I have combined the KNN with the transformed data and re-ran the results to see if anything changes. In the transformed data accuracy

is strongly affected by the increase in k - value

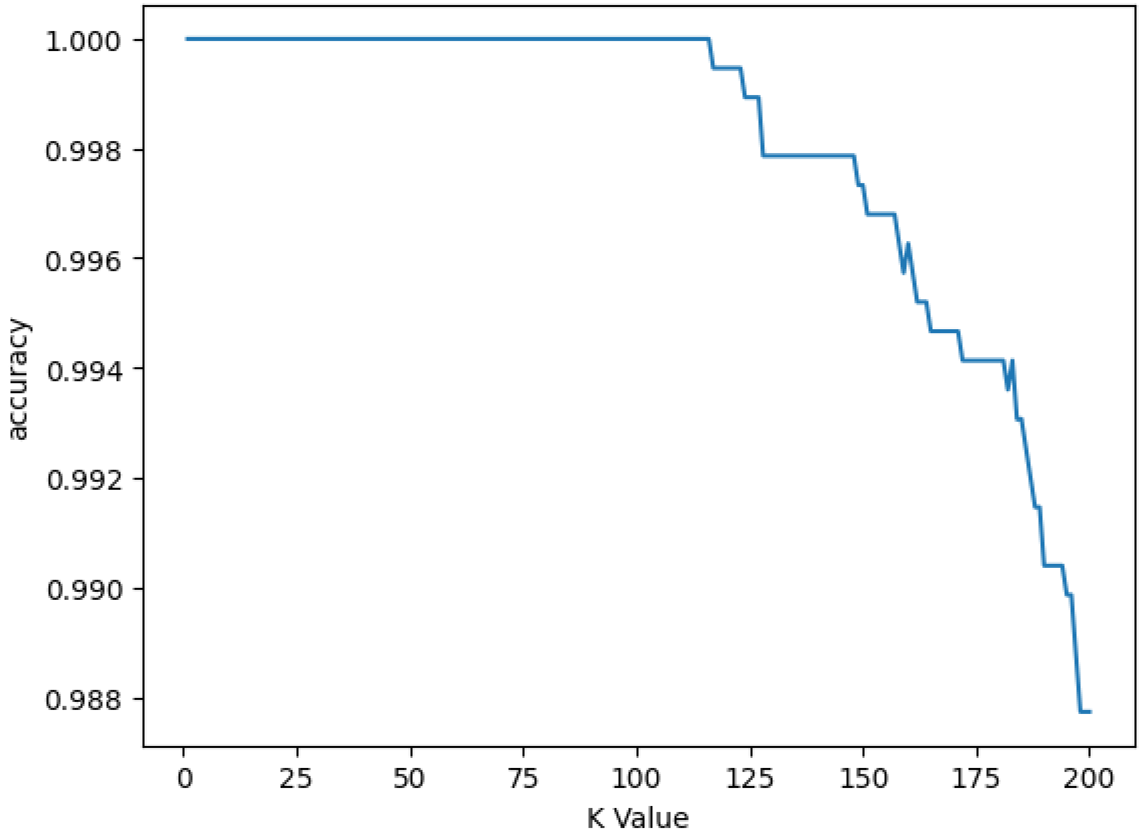


8. For the Gaussian kernel with KNN there was a steady increase in the accuracy with increase in k - value but gradually dropped after a

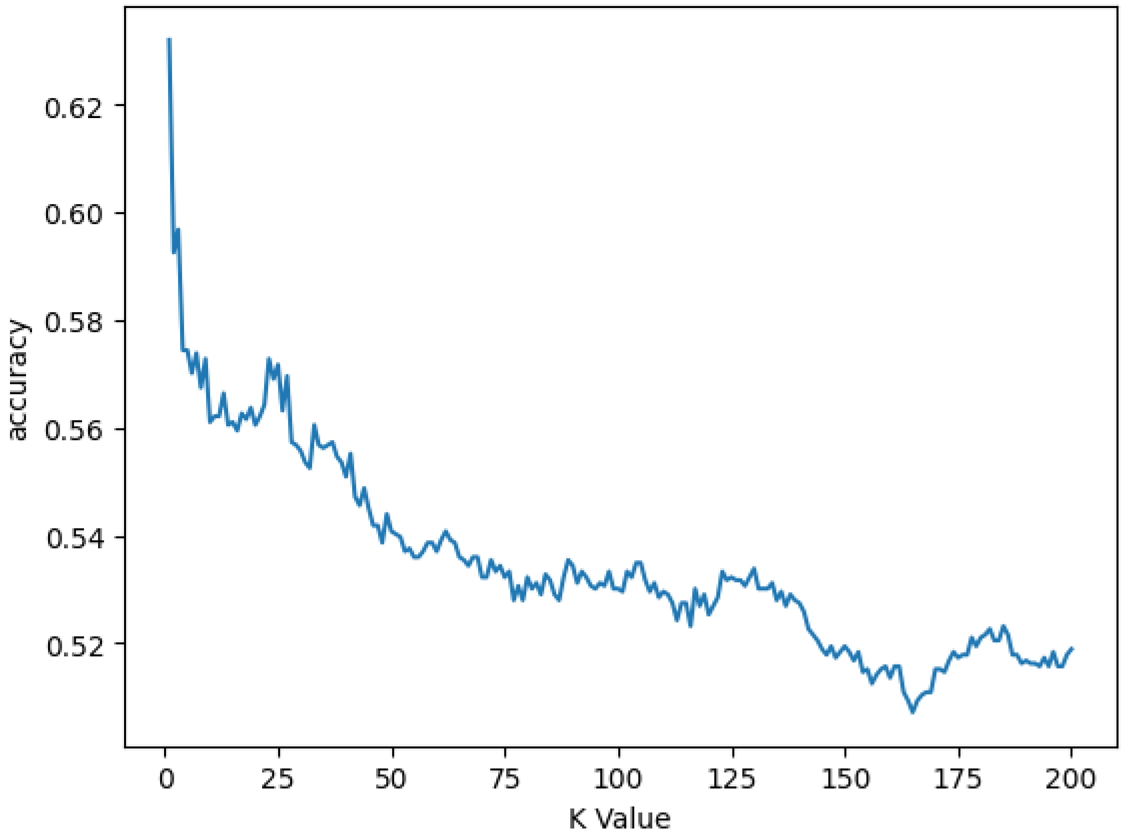
threshold of k = 50



9. For the sigmoid kernel combined with KNN, the model had 100% accuracy for the most k- values until k = 125

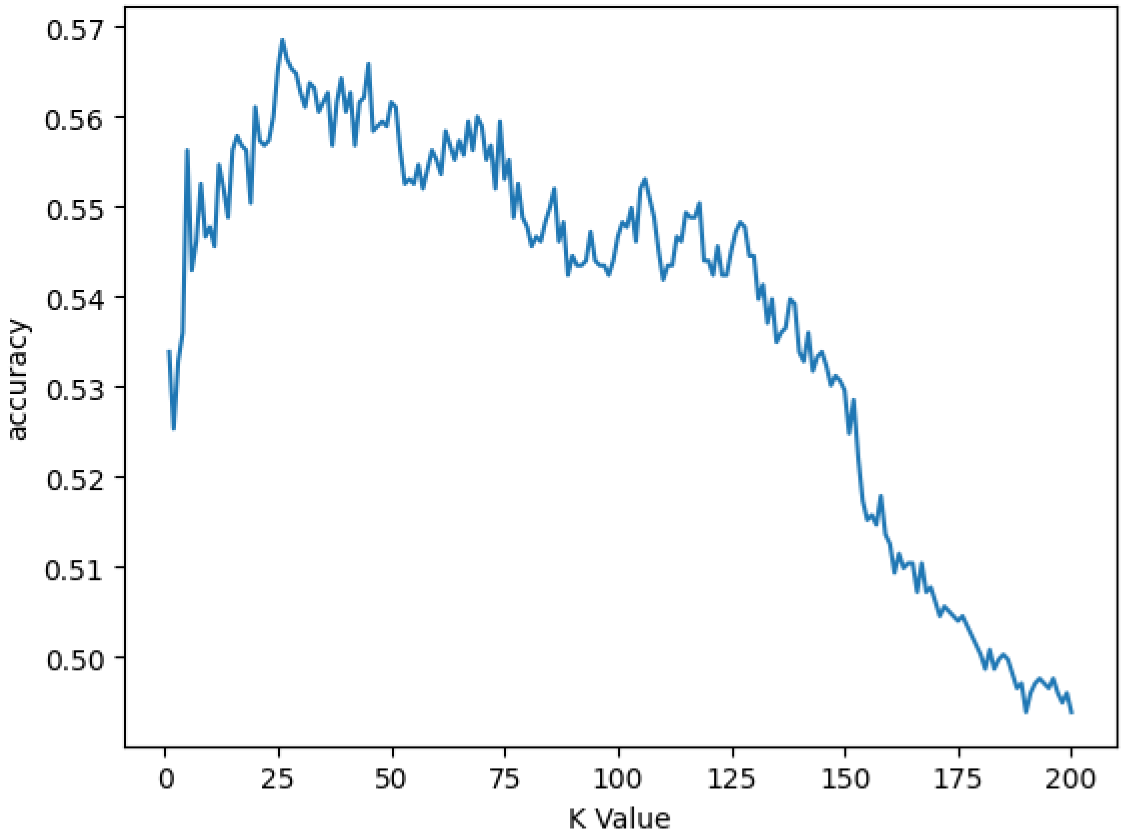


10. Polynomial Kernel didn’t behave properly with KNN classifier where the maximum accuracy it could achieve was only 62% and the drop in accuracy was drastic with increase in K - Values in the first quarter



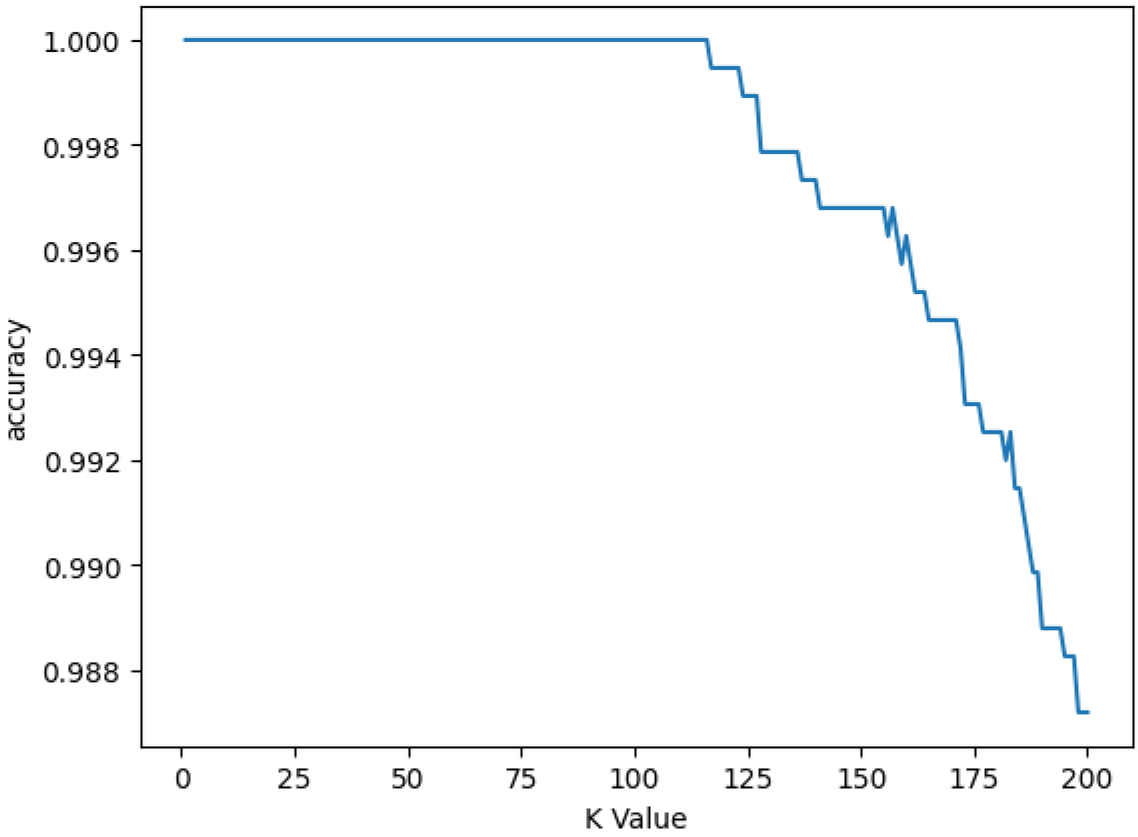
11.

12. RBF kernel was also not that successful with KNN where the maximum accuracy it could reach was 57%



13.

14. However Fourier kernel was successful with KNN where it achieved 100% accuracy for most K values and dropped its accuracy to 98% on continuous increase of K-Value


1.000
0.998
0.996
0.994
0.992
0.990
0.988
accuracy
0
25
50
75
100
125
150
175
200
K Value

15.