CS6364 MACHINE LEARNING HW 2

Sagar Sheth

Question 1: Myocardial infarction complications

Objective:

The objective of this Question is to implement a develop a binary classification model using the Naive Bayes algorithm. Based on the medical information for a patient, the objective is to predict the target variable 'ZSN' i.e. will the patient have Chronic heart failure.

Data Preprocessing & Univariate Analysis

Data Extraction:

- Importing the data from the online repository.
- The Data is already split into features X and target variables y.
- From y we only have to extract the column 'ZSN'

Data Cleaning:

- Using pd.isnull().sum(), the total no of Null values for each column was calculated.
- Based upon the count of the null values, initially columns containing null values greater than ~10% of the size of the data set were dropped.

Univariate Analysis

- Using the pd.dtypes function to understand what data type has been assigned to columns.

```
print('Column Names:')
print(X.dtypes)
```

```
Column Names:
AGE            float64
SEX              int64
INF_ANAM       float64
STENOK_AN      float64
FK_STENOK      float64
                ...
ANT_CA_S_n     float64
GEPAR_S_n      float64
ASP_S_n        float64
TIKL_S_n       float64
TREN_S_n       float64
Length: 111, dtype: object
```

Figure 1 Output of pd.dtypes

- As this is a classification problem, first I performed univariate analysis on the target variable to analyze the distribution.

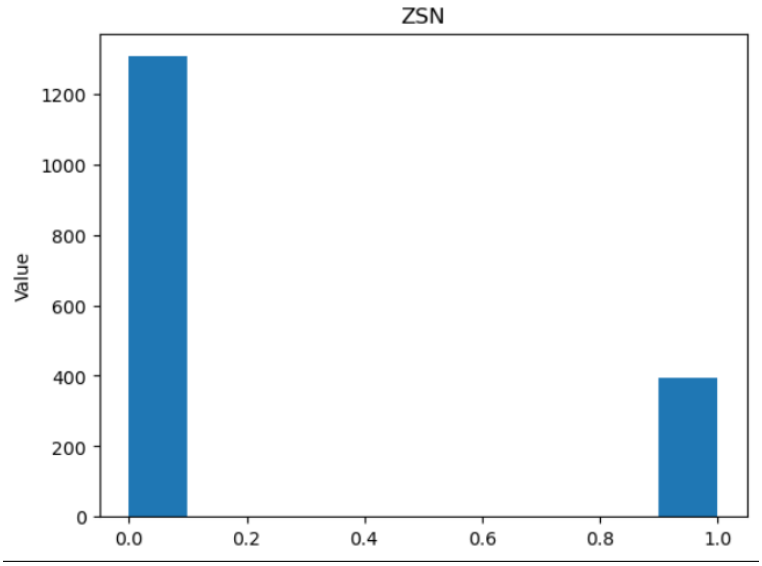


Figure 2 Histogram of ZSN

- For the features using the below code I analyzed the following information.
    - Distribution of Data.
    - Checking if the data is Continuous/Discrete.
    - Getting the mode, median and mean.
    - Getting the Maximum and Minimum value
    - Getting the no of null records in that column.

Data Cleaning Continued:

- Based on the above analysis features that had values either 1 or 0 and were highly skewed to one value and had significant amounts of null rows were dropped.
- As these features will have insignificant correlation with the target variable and have 100+ rows with null values.

Univariate Analysis Continued/Data Visualization:

- For the remaining Features Histograms and Boxplots were generated to further analyze the distribution of data.

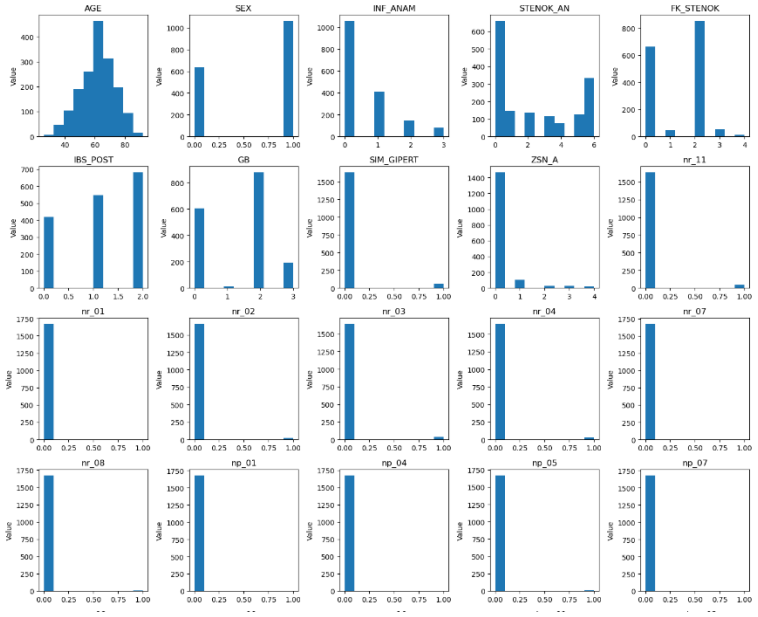


Figure 2 Histograms

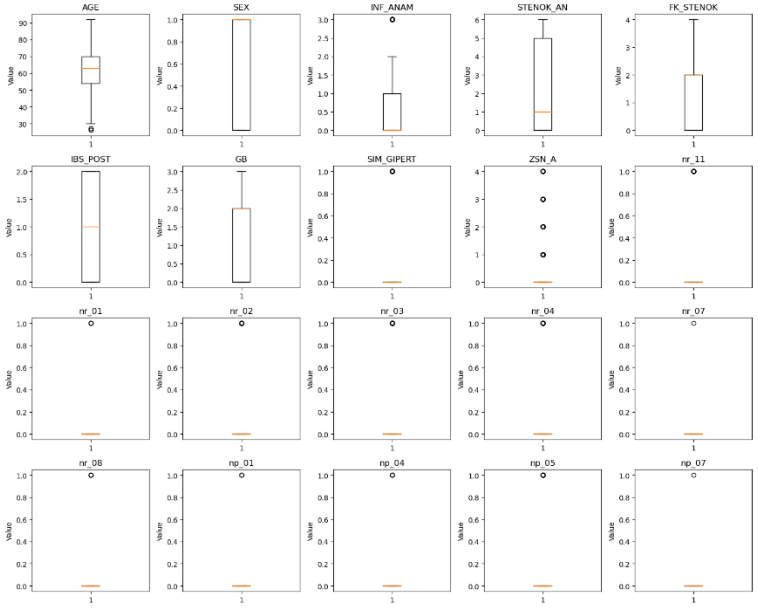


Figure 3 Boxplots

Data Cleaning Continued:

- For the remainder of features, rows with null values were deleted.
- The resultant data set contained 1107 rows × 70 columns.
- Approximately 600 rows of data were lost, i.e. 35% of original dataset.
- Therefore from original dataset now columns contained more that 100 rows of null values were dropped.
- The resultant data set contained 1436 rows × 59 columns.

Data Normalization:

- Using the below formula the features were normalized.

$$\frac{\boldsymbol{curr}_{\boldsymbol{value}} - \boldsymbol{min}_{\boldsymbol{value}}}{\boldsymbol{max}_{\boldsymbol{value}} - \boldsymbol{min}_{\boldsymbol{value}}}$$

Feature Selection:

- For initial approach feature having less than 100 null rows were selected.
- In second approach Feature Selection was used, discussed in the document below.

Data Splitting:

- The data was split into 2 parts, 70% for train and 30% for test and holdout.
- The second part was split into 2 parts, 67% for test and 33% for holdout.

Methodology:

Naive Bayes is a probabilistic machine learning algorithm rooted in Bayes' theorem. It's particularly adept at classification tasks. The "naive" assumption is that features are conditionally independent given the class, simplifying computations. The algorithm calculates the probability of a class given a set of features using Bayes' theorem: P(Y|X) = P(X|Y) * P(Y) / P(X). Here, P(Y|X) is the probability of class Y given features X, P(X|Y) is the likelihood of observing X given Y, P(Y) is the prior probability of Y, and P(X) is the evidence probability. Despite its simplicity, Naive Bayes often performs surprisingly well in various applications.

We learn a generative model $\mathrm{P}_\theta(\mathrm{x}, \mathrm{y})$ by maximizing the maximum likelihood:

$$max_\theta \frac{1}{n} \sum_{i=1}^{n} \quad = \log \mathrm{P}_\theta(\mathrm{x}_{(\mathrm{i})}, \mathrm{y}_{(\mathrm{i})}).$$

Upon getting the derivative we get a closed form solution as follows.

$$\frac{\phi_k}{\sum \phi_l} = \frac{n_k}{n}$$

$$\psi_{jk} = \frac{n_{jk}}{n_k}$$

When making predictions, Naive Bayes calculates the probability of each class given the observed features using Bayes' Theorem. The class with the highest probability is then assigned as the predicted class.

Algorithm: Bernoulli Naive Bayes
Type: Supervised learning (multiclass classification)
Model Family: Linear Models
Objective Function: Log likelihood
Optimizer: Closed Form Solution

Approach

- In the initial approach of using 59 Features we get the below results for train set.
- Train Accuracy: 0.7592039800995025
- Confusion matrix:

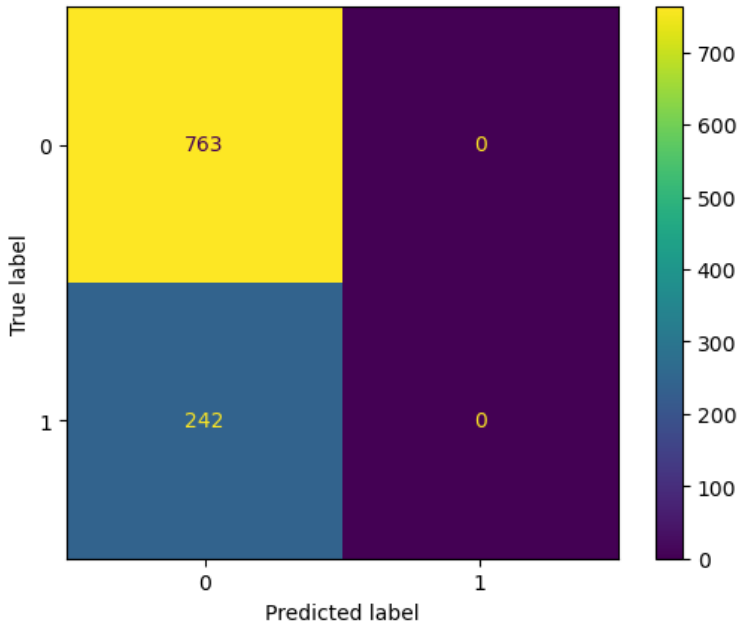


Figure 4 Initial Model Confusion Matrix

- Upon Analyzing we observe that the model is predicting everything as Class 0.
- Therefore this approach is scrapped.

Feature Selection Continued:

- A Correlation matrix is calculated with the features and the target Variable.
- All features who's absolute correlation with the target variable is >0.02 are chosen the rest are scrapped
- Features list:
    - AGE
    - SEX
    - IBS_POST
    - GB
    - ZSN_A
    - nr_01
    - nr_03
    - nr_04
    - nr_08
    - np_01

- np_04
- np_08
- np_09
- np_10
- endocr_01
- endocr_02
- zab_leg_01
- K_SH_POST
- MP_TP_POST
- FIB_G_POST
- ant_im
- lat_im
- IM_PG_P
- fibr_ter_06
- fibr_ter_07
- R_AB_1_n
- NITR_S
- NA_R_1_n
- NOT_NA_1_n
- LID_S_n
- B_BLOK_S_n
- ANT_CA_S_n
- GEPAR_S_n
- TIKL_S_n
- TRENT_S_n

Model Evaluation:

Train

- Accuracy: 0.7676669893514037

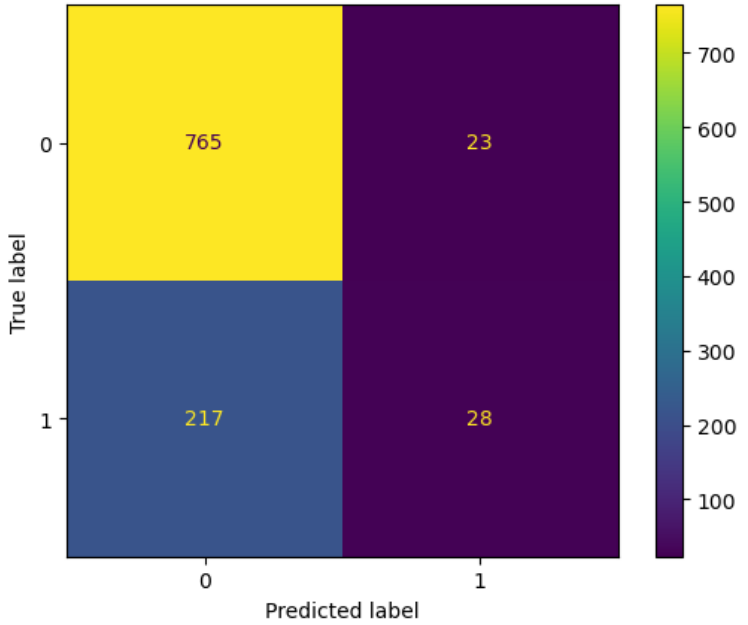


Figure 5 Final Model Confusion Matrix Train

	precision	recall	f1-score	support
0	0.78	0.97	0.86	788
1	0.55	0.11	0.19	245
accuracy			0.77	1033
macro avg	0.66	0.54	0.53	1033
weighted avg	0.72	0.77	0.70	1033

Figure 6 Classification Report Train

Test

- Accuracy: 0.7804054054054054

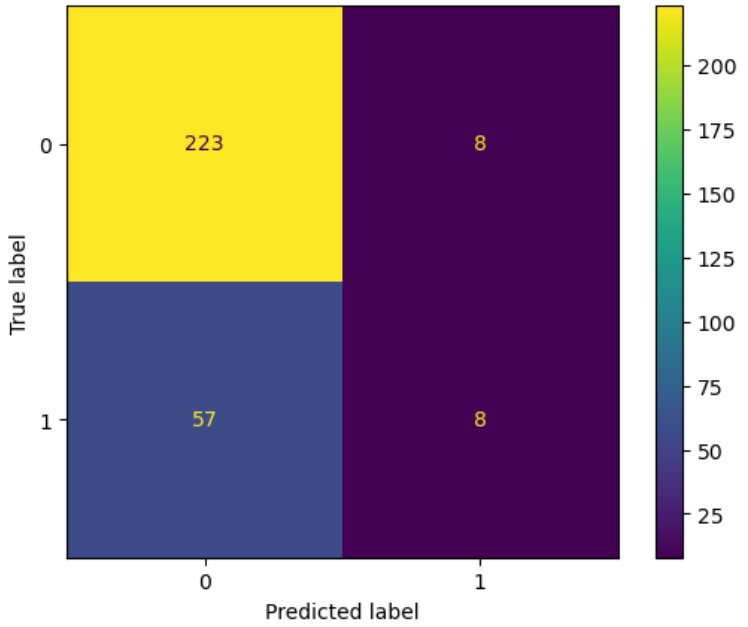


Figure 7 Final Model Confusion Matrix Test

	precision	recall	f1-score	support
0	0.80	0.97	0.87	231
1	0.50	0.12	0.20	65
accuracy			0.78	296
macro avg	0.65	0.54	0.54	296
weighted avg	0.73	0.78	0.72	296

Figure 8 Classification Report test

Holdout:

- Accuracy: 0.7619047619047619

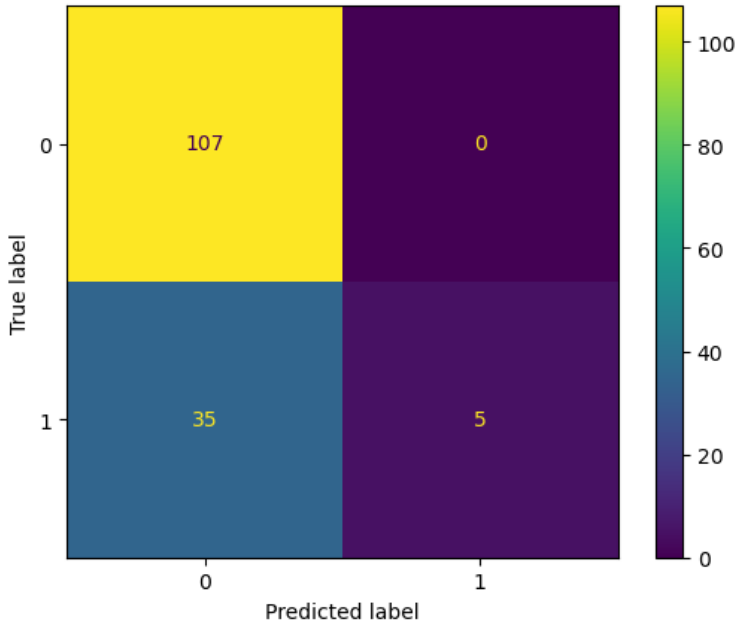


Figure 9 Final Model Confusion Matrix Holdout

	precision	recall	f1-score	support
0	0.75	1.00	0.86	107
1	1.00	0.12	0.22	40
accuracy			0.76	147
macro avg	0.88	0.56	0.54	147
weighted avg	0.82	0.76	0.69	147

Figure 10 Classification Report Holdout

ROC Curve

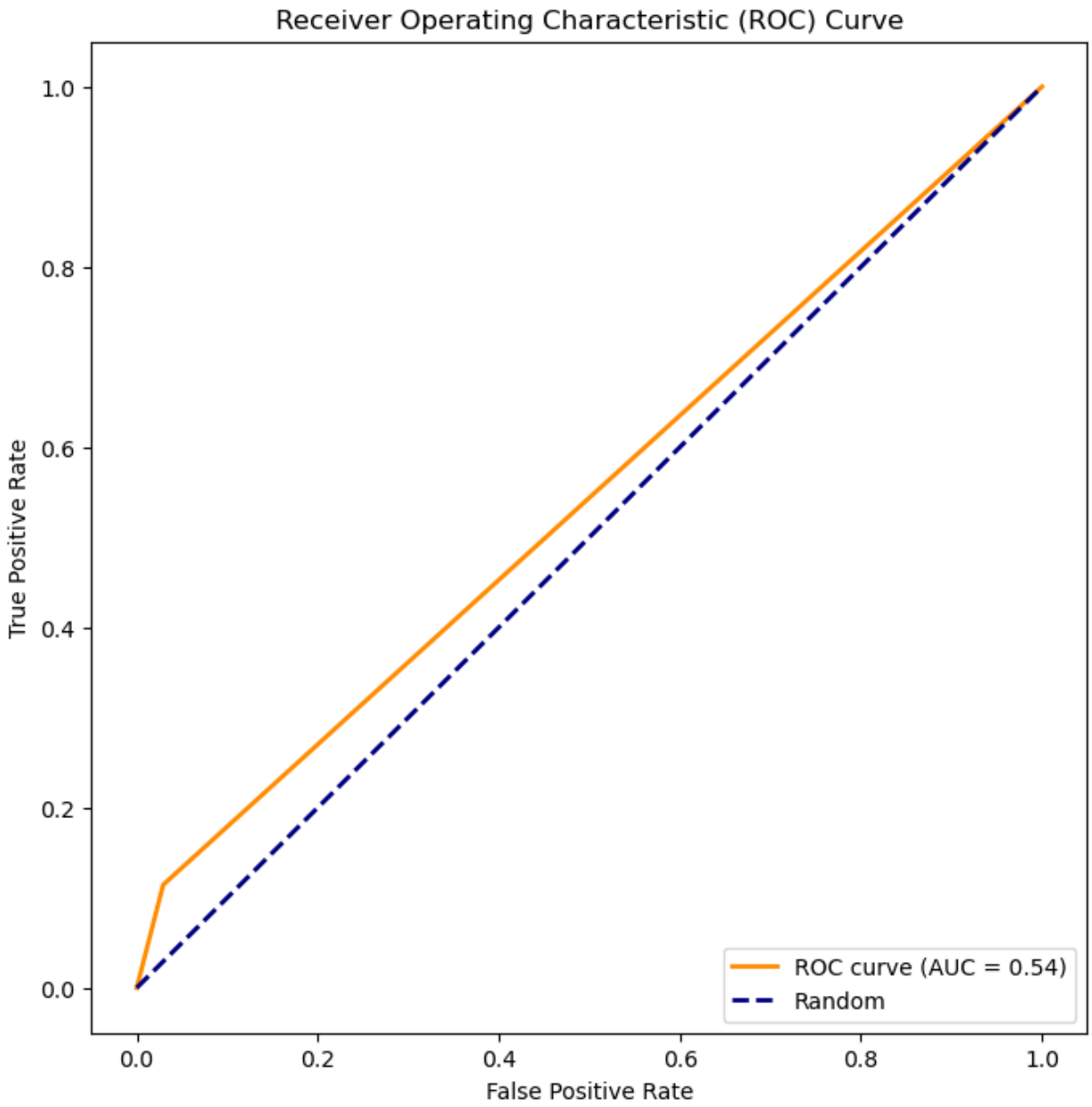


Figure 11 ROC Curve

Learning Curve

After taking different Sizes of training data from 50 to 1000 with step size 50, the performance of the test and holdout set is plotted along with baseline.

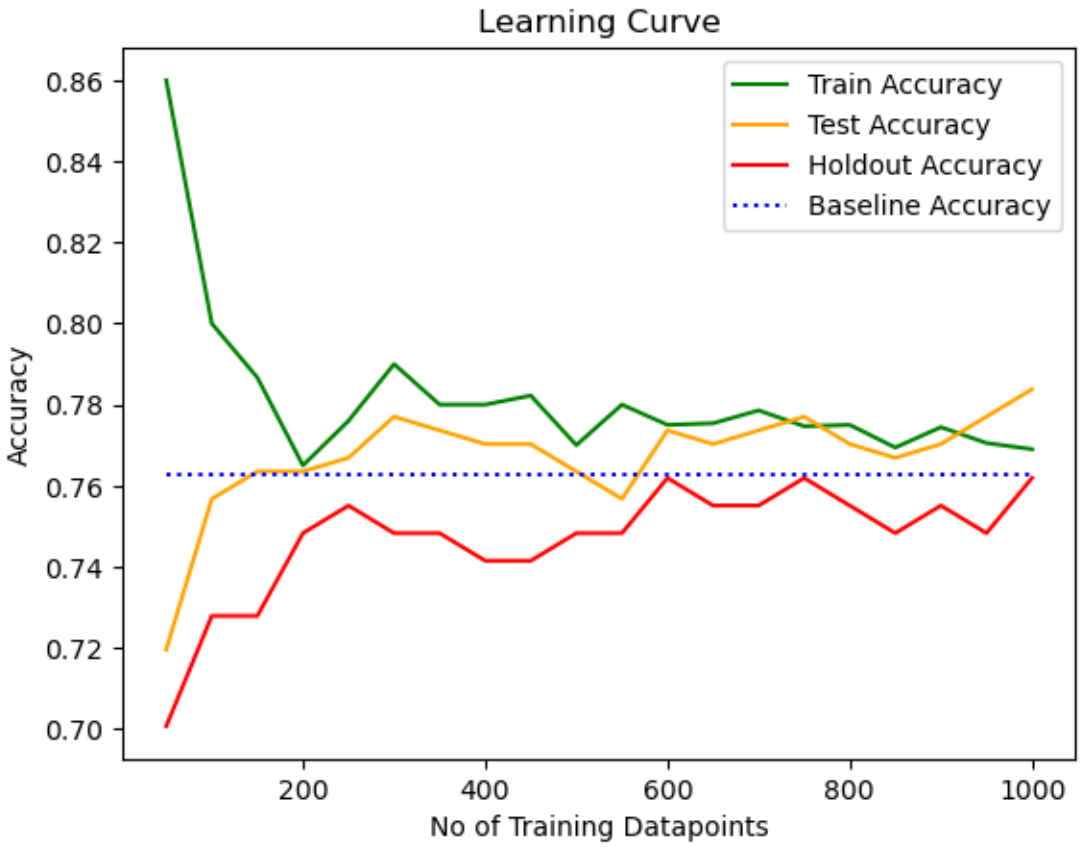


Figure 12 Learning Curve

Conclusion and Insights:

- An AUC of 0.54 suggests that the classifier has only a slight ability to discriminate between the two classes. It is better than random chance but may not be very effective in distinguishing between positive and negative instances. Generally, an AUC between 0.5 and 0.7 is considered to have low discriminatory power, while an AUC between 0.7 and 0.8 is moderate, and an AUC above 0.8 is considered good.
- But as the Classes are highly Skewed to one class AUC is not a good metric to judge the Model on.
- On Comparing the Model performance on baseline, the model just barely outperforms baseline.
- Naive Bayes can be less effective on imbalanced datasets where one class is significantly more prevalent than the other. Since it estimates class probabilities independently for each feature, it may not handle imbalances well.
- Boosting Methodologies maybe used to further enhance this model.

Question 2: Emotion Analysis using SVM and K-Means Clustering

Objective:

Develop an understanding of emotion recognition in text using Support Vector Machines (SVM) for classification and K-Means clustering for pattern discovery. Based on the dialogues from the popular SitCom F.R.I.E.N.D.S we have to build a model that accurately predicts the emotion out of 8 Categories.

Data Preprocessing & Univariate Analysis

Data Extraction:

- Using the below python code, the .json files were read and converted into Data frames.

```
file_path = 'friends_train.json'
with open(file_path,'r') as file:
  data = json.load(file)
flat_data = [utterance for conversation in data for utterance in conversation]
df = pd.DataFrame(flat_data)
```

Data Preprocessing

- After the data is extracted from the .json, we perform tokenization on the utterance field to perform EDA on the Data.
- Additionally we create an additional columns, where we perform the following text operations in sequential Order.
    - Tokenize the data.
    - Remove non-Alpha Numeric words.
    - Remove stop words.
    - Make the words in Lower Case.
    - Lemmatize the words.
    - Concatenate the list of words delimited by Space to then generate TFIDF on the utterance field.

EDA

- Perform Univariate Analysis on the emotion field to see the distribution of the target Variable.

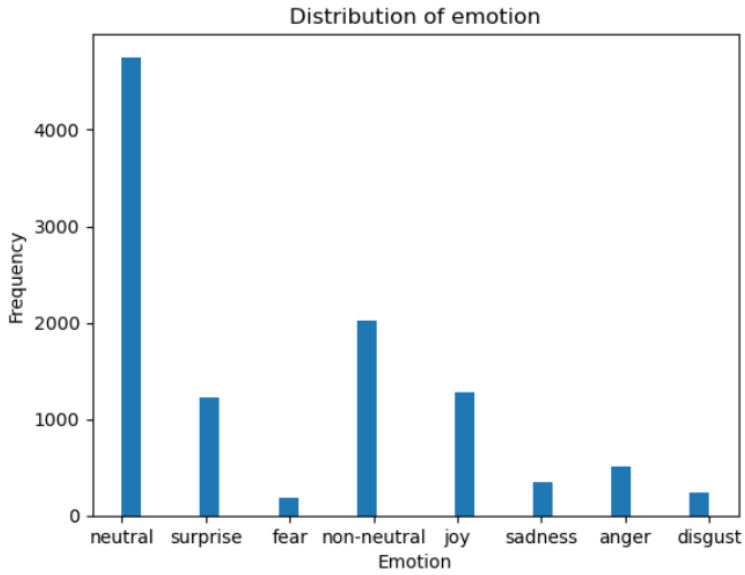


Figure 13 Distribution of emotion.

- Generated a Histogram plotting the top 20 most common words after performing the text operations

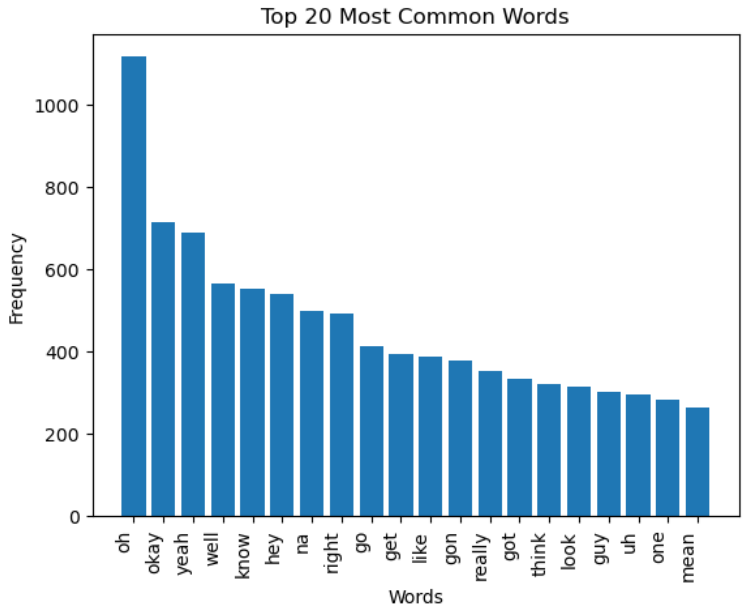


Figure 14 Top 20 Most Common Words.

- Generated A histogram for the most common words for Each Emotion.

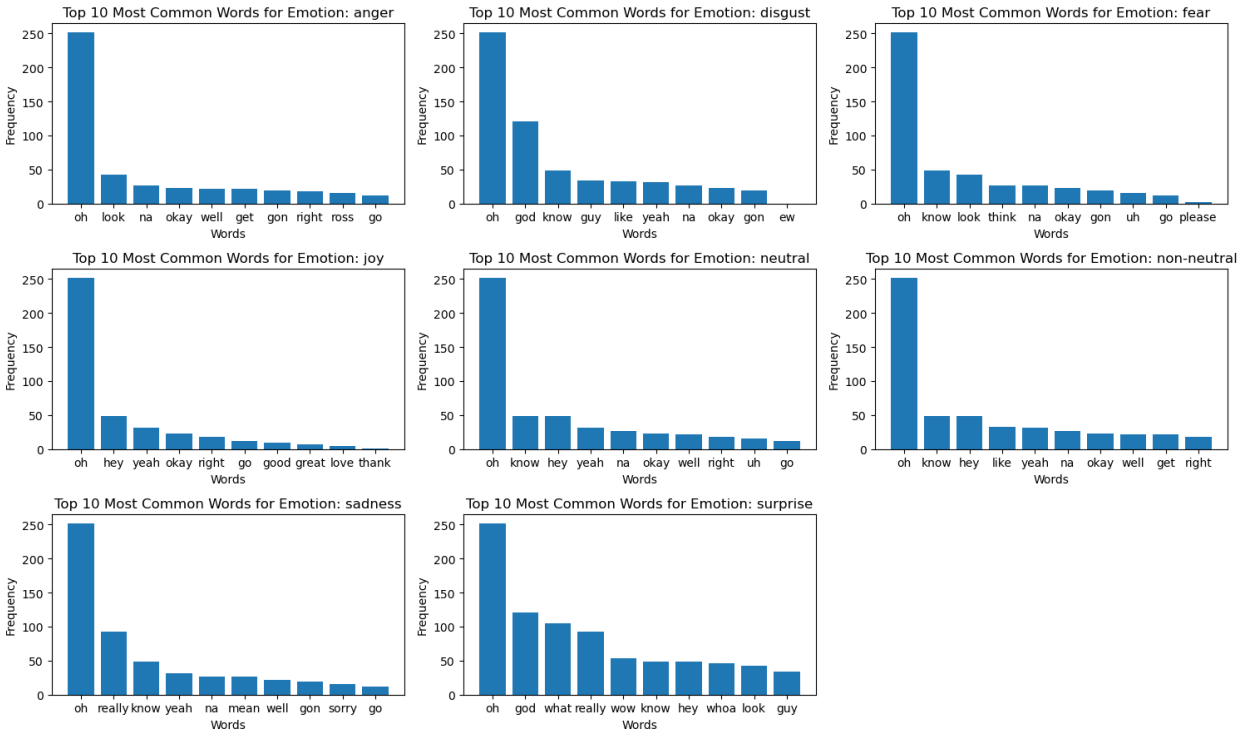


Figure 15 Top 10 Common Words for Each Emotion.

Data Preprocessing Continued

- Using the TfidfVectorizer we create a vectorized matrix for the processed_text and then apply it to the train dataset.
- Creating Target Variable from Emotion, we assign a numerical value to each emotion as follows.
    - anger: 0
    - disgust: 1
    - fear: 2
    - joy: 3
    - neutral: 4
    - non-neutral: 5
    - sadness: 6
    - surprise: 7
- As the Data is already split in 3 JSON files, train, test & Holdout the data need not be splitted.
- We Drop the non-required Columns and convert the features and target variable into numpy arrays.

Methodology

Support Vector Machine (SVM) is a powerful machine learning algorithm used for classification and regression tasks. It works by finding a hyperplane that best separates data into different classes, maximizing the margin between them. SVM is effective in high-dimensional spaces and widely employed in various domains for its robust performance.

Since there are 8 classes of the Target Variable, we will use the one v/s rest classifier from scikit learn to model and then predict the data. This Model Assigns individual probability to each class and then the class with the high probability is classified as the Target Variable.

K-Means is a popular unsupervised machine learning algorithm used for clustering data into distinct groups. It operates by iteratively assigning data points to clusters based on their proximity to cluster centroids and recalculating centroids until convergence.

The algorithm aims to minimize the within-cluster sum of squares, optimizing cluster homogeneity. K-Means requires users to specify the desired number of clusters (k) beforehand. Widely applied in various fields, such as image segmentation, customer segmentation, and anomaly detection, K-Means is efficient for large datasets but sensitive to initial centroid selection. Its simplicity and effectiveness make it a foundational tool for pattern recognition and data analysis.

Evaluation

Train

- Accuracy: 0.8457532430641038

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	513
1	0.99	0.82	0.90	240
2	0.96	0.43	0.60	185
3	0.85	0.66	0.75	1283
4	0.78	0.95	0.86	4752
5	0.89	0.64	0.74	2017
6	0.90	0.77	0.83	351
7	1.00	1.00	1.00	1220
accuracy			0.85	10561
macro avg	0.92	0.78	0.83	10561
weighted avg	0.86	0.85	0.84	10561

Figure 16 Classification Report Train.

Test

- Accuracy: 0.6544862518089725

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	161
1	0.79	0.22	0.34	68
2	0.14	0.03	0.05	32
3	0.50	0.34	0.40	304
4	0.63	0.84	0.72	1287
5	0.43	0.27	0.33	541
6	0.49	0.24	0.32	85
7	1.00	0.99	1.00	286
accuracy			0.65	2764
macro avg	0.62	0.49	0.52	2764
weighted avg	0.63	0.65	0.63	2764

Figure 17 Classification Report Test.

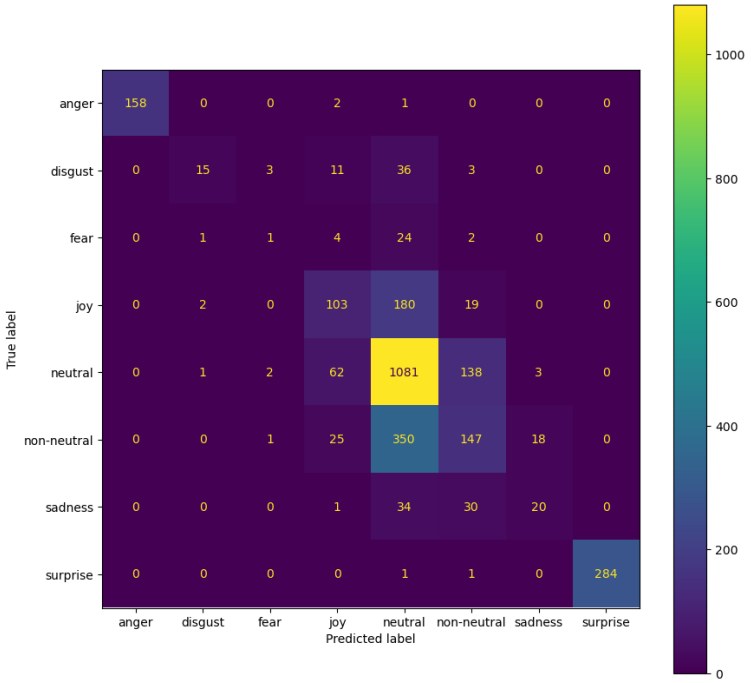


Figure 18 Confusion Matrix Test.

Holdout Set

- Accuracy: 0.6426146010186757

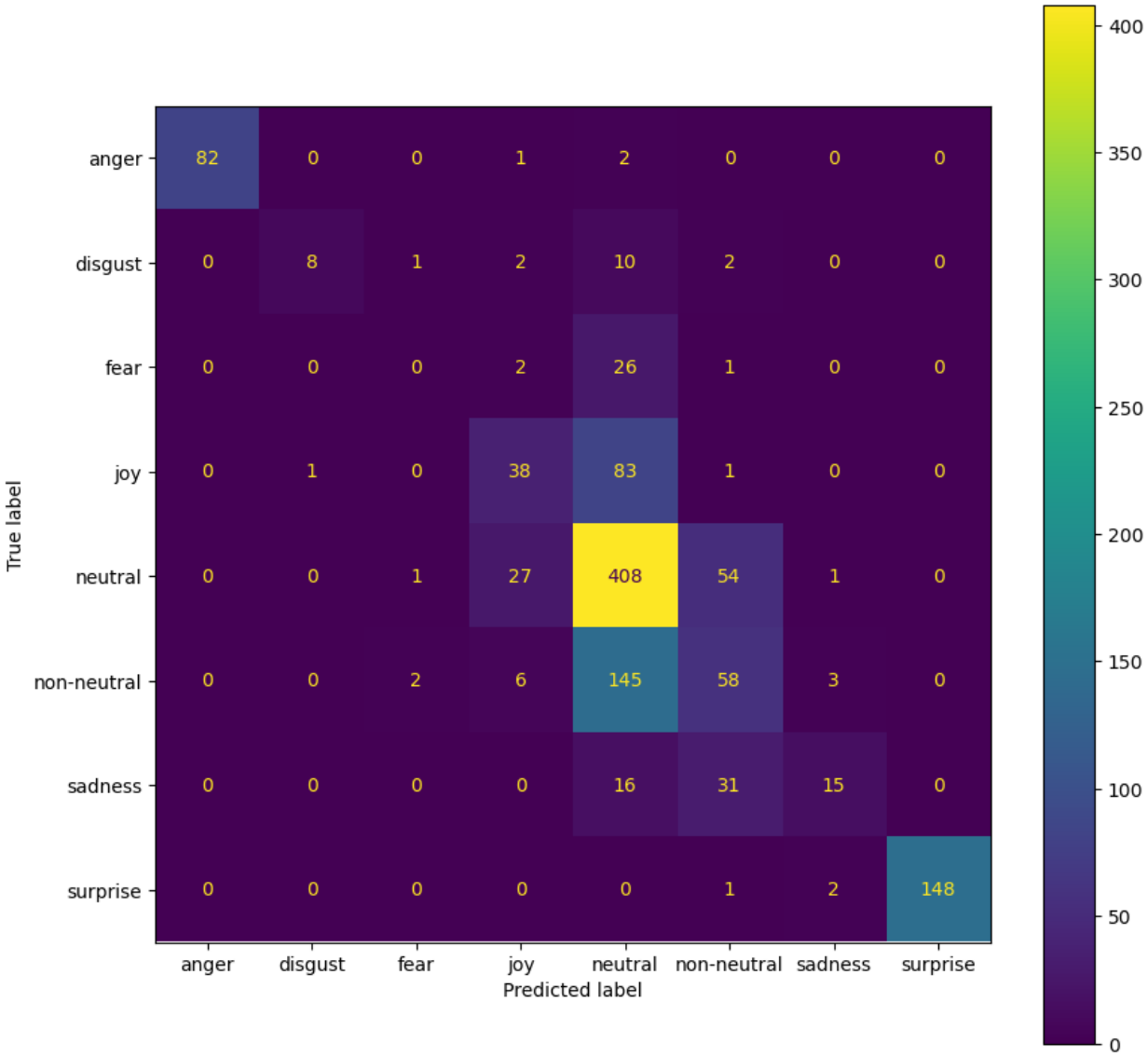


Figure 19 Confusion Matrix Holdout.

Training Curve

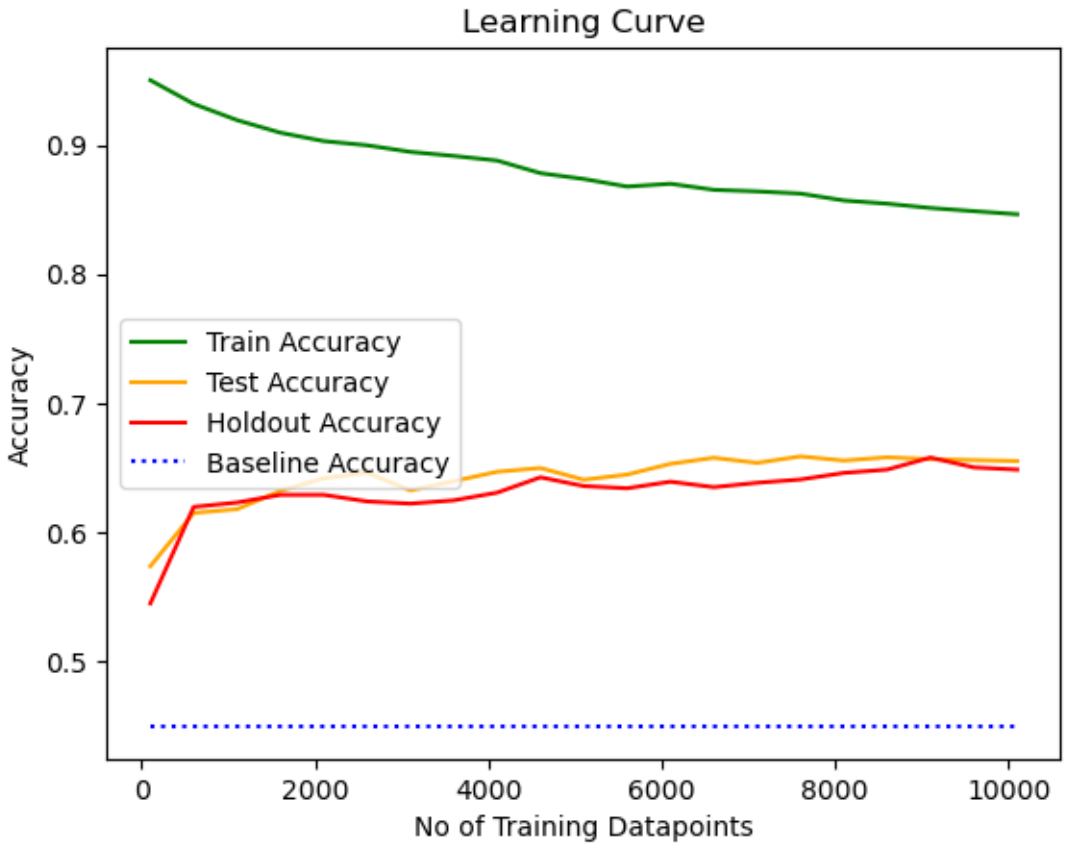
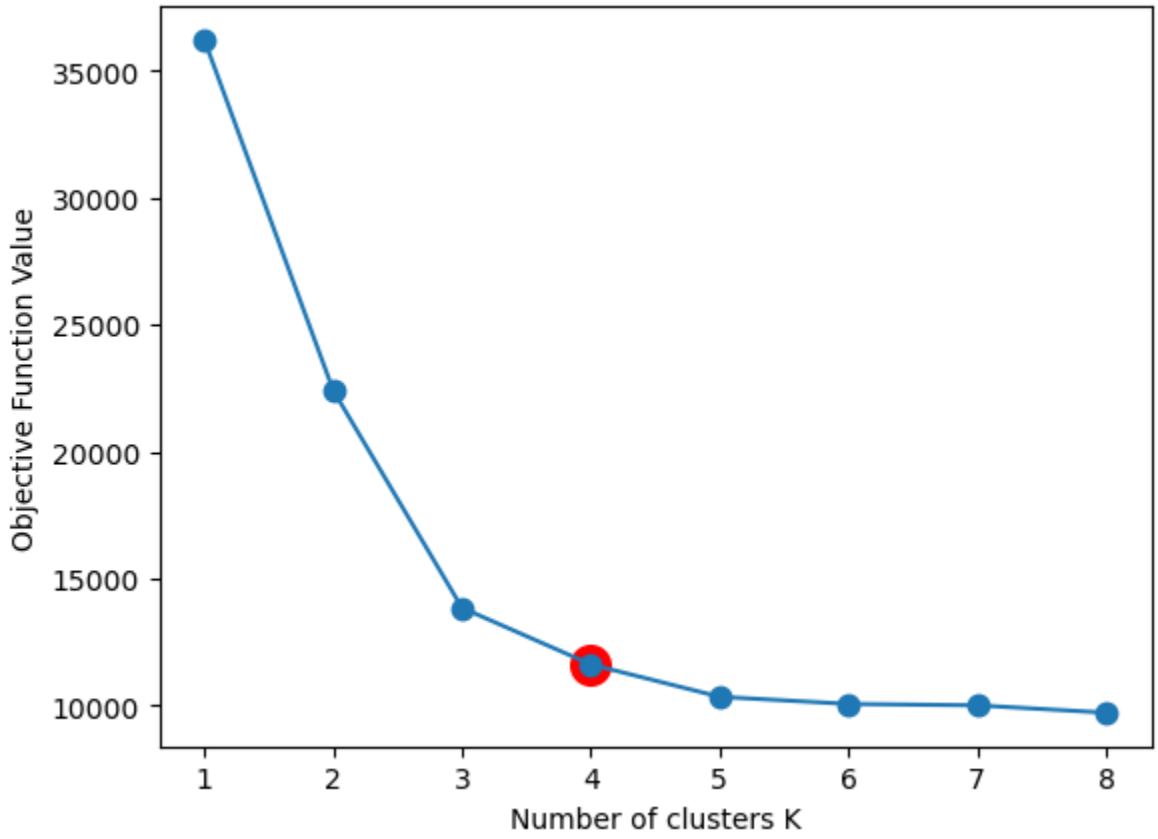


Figure 20 Training Curve.

K Means Clustering Results



As we Can Observe the Appropriate no of clusters using the Elbow Methos is 4 or 5. What We can infer from this information is that the inherent structure in the features doesn't align perfectly with the provided target labels. As Observed during Univariate Analysis on the Emotion Target Variable, there are some classes whose size is very small compared to other classes. What might be occurring during K Means is that some classes are spatially close in the feature space.

This can be concurred from the SVM Model we have created as the Accuracy is 64% on the holdout set suggesting that some of the classes are spatially close and SVM is unable to properly generate a Hyperplane to separate the Classes without overlap.