# Report for Assignment 2 of COL 774

**Ayushi Agarwal**
2018ANZ8503
ayushi.agarwal@cse.iitd.ac.in

## 1 Introduction

The goal of this assignment is to implement two very popular Machine learning algorithms for text classification and classification of FASHION-MNIST dataset. The algorithms used are:

- Naive Bayes - This algorithm is used to classify tweets into two classes (positive and negative). The implementation of this model will be discussed in Section 2.1
- Support Vector Machines - This algorithm will be used to implement the classification of FASHION MNIST dataset. The implementation of this model will be discussed in Section 2.2

In the next section of this report discusses the algorithm in each question, the method to implement the algorithm, report the results and present an analysis of the results obtained.

## 2 Method, Results and Analysis

In this section, we would discuss our algorithm implementation and the results obtained. We would present an analysis of the results as well[I]

### 2.1 Question 1 - Naive-Bayes

In this Question, we have used Naive Bayes algorithm for the classification of tweets by different twitter users. Given a user's tweet, the task is to predict the sentiment of the tweet whether it is positive or negative.
Number of training samples are 1.6 million.
Number of test samples are 359.

We have implemented the Multinomial Event Model for Naive Bayes with Laplace Smoothing.

#### 2.1.1 Part a - Naive Bayes on Raw Data

In the first part, we have implemented the model on raw data. The tweets are first pre-processed only to remove separators of the words and punctuation's. The dictionary is created by adding all the words that appear at least once in the data into a bag called bag of words. This is also called "Bag of Words" model. We also keep a count of the frequency of the words in the data in the dictionary. Laplace Smoothing is done by using $C = 1$ to avoid any zero probabilities for the words that didn't occur in the train set.

The features $x_j$ of the model denote the identity of the $j^{th}$ word in the tweet of size $n_i$ where $i$ is the $i^{th}$ training sample amongst $m$ training samples. A feature $x_j$ can take any value between $(1, ...., |V|)$, where $|V|$ is the size of the dictionary that we have built. There will be a parameter $\theta_k$ for all the words in the dictionary. The maximum likelihood estimates of the model parameters $\theta$ are as follows:

$$\theta_{k|y=1} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} \mathbb{1}\{x_j^{(i)} = k \wedge y^{(i)} = 1\} + C}{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 1\} n_i + |V|\}} \quad (1)$$

$$\theta_{k|y=0} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} \mathbb{1}\{x_j^{(i)} = k \wedge y^{(i)} = 0\} + C}{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 0\} n_i + |V|\}} \quad (2)$$

$$\phi_y = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 1\}}{m} \quad (3)$$

Equations 1 and 2, are the parameters of the model. Equation 3 is used to calculate the class priors. We have used log in our implementations to avoid underflow issues.

This same model is used across the complete implementation, however we will employ several feature engineering techniques in the later subsections which changes the dictionary created hence the features of the model.

**Results a:** The **train accuracy** achieved by training the model on raw data is **84.21%** and the **test accuracy** on raw data is **80.77%**.

**Results b:** The test accuracy obtained by randomly guessing one of the outcomes (**random prediction**) is **50%**. We implemented this by randomly generating a value between $(0, 1)$ for each test tweet and then guessing the category based on the random value.

[I] https://github.com/agarwal-ayushi/Machine-Learning-Assignments/tree/master/Assignment2

The Test Accuracy obtained by predicting the majority occurring class in the training data (**Majority Prediction**) is **50%**.

The amount of improvement of improvement obtained in part(a) over these models are 1.667 and 1.616 times respectively.

| **Model** | **Train Acc(%)** | **Test Acc(%)** | **AUC** |
|---|---|---|---|
| Raw Data NB | 84.21 | 80.78 | 0.888 |
| Raw Data Random Pred | - | 48.48 | 0.5 |
| Raw Data Majority Pred | - | 50 | 0.5 |
| Lemma Data NB | 79.28 | 80.51 | 0.883 |
| Feature 1 Cutoff-40 | 76.78 | 82.72 | 0.895 |
| Feature 2 Uni+Bigrams NB | 77.83 | 82.45 | 0.902 |
| Feature 3 Uni+Trigrams NB | 77.50 | 83.28 | 0.895 |

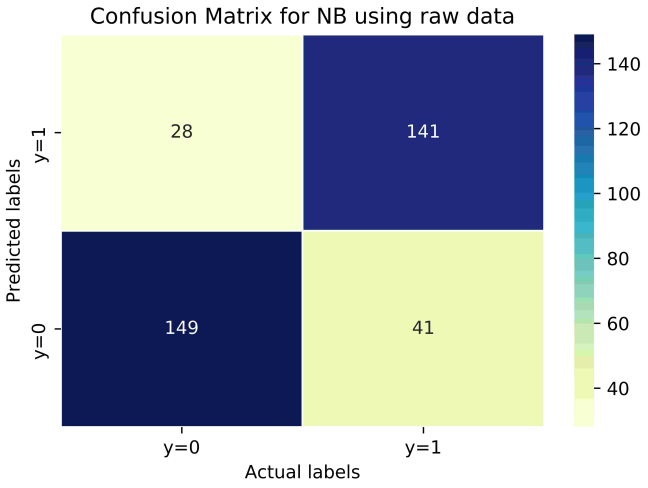
**Table 1:** Accuracy obtained by training NB models using different features like Raw Data, stemmed and lemmatized data, Cutoff of words over frequency=40, Bigrams and Trigrams

Table 1 shows the accuracy number obtained for different models trained on different features. It also shows the AUC score of the model depicting how well the model predicts true positives.

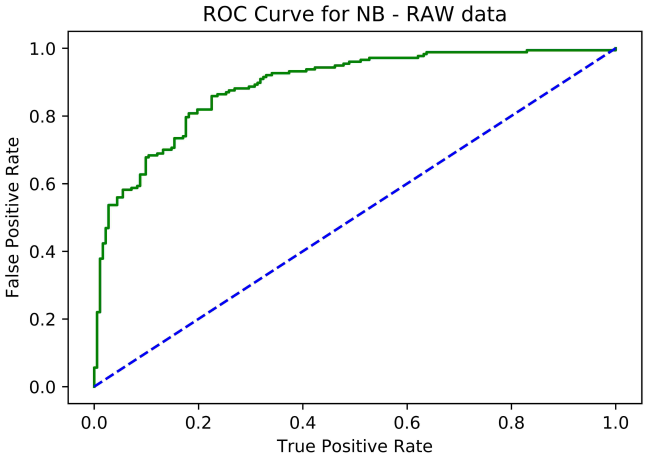
**Results c:** Figure 1 shows the confusion matrix of the model tested on test data and the ROC curve of the model trained on Raw Data. The confusion matrix is as follows:
Confusion Matrix= $\begin{pmatrix} tn & fn \\ fp & tp \end{pmatrix}$=$\begin{pmatrix} 149 & 41 \\ 28 & 141 \end{pmatrix}$

The highest value of the diagonal entry is the negative class with value 149. This means that the model is able to predict negative class more accurately since the number of false negatives are lesser than false positives. This also shows that the model is not very accurate to make correct predictions between the two classes. Its AUC score is 0.888.

**Results d:** We will now process the data by removing the features that do not contribute in the classification of the data. Some such words are stopwords like 'to', 'the', 'and', etc. We will also do stemming and lemmatization of the data to have common features for words that have the same meaning. We also remove twitter handles by using tokenization and learn a new model using the new features.
The test accuracy obtained for this data is 80.51%. The




**(a)** Confusion Matrix for Model




**(b)** ROC Curve for the model

**Figure 1:** Confusion Matrix and ROC curve for NB Model Trained on RAW DATA

AUC score of the model is 0.883. The confusion matrix for this model is :
Confusion Matrix= $\begin{pmatrix} tn & fn \\ fp & tp \end{pmatrix}$=$\begin{pmatrix} 141 & 34 \\ 36 & 148 \end{pmatrix}$

So we see that this model is actually able to predict the positive category more accurately in contrast to the previous model that was trained on Raw Data. However, we do see a dip in the accuracy by 0.2%.

**Results e:** Feature Engineering is an important aspect in Machine Learning which is used to manipulate existing features or to create new features to improve the overall accuracy of the model. In the previous models, we have used every word in the tweet to create the dictionary hence every word is used as a feature.

**Feature 1: Cut-off Frequency on Existing dictionary**
There are some words that appear in only some tweets and have very less frequency, are not very important in determining the class of the tweet or of being a tie-breaker. Such words are the words that have very less frequency. So the dictionary can be reduced in size, decreasing the number of model parameters, to increase

the accuracy of the model by reducing noise.
The test accuracy obtained by the model trained after clipping the words that have $frequency < 40$ is **82.72%**. However, there is a dip in the training accuracy by 5% since we have removed more words from the dictionary. But this increased the generalization capability of the model. The AUC score of the model is 0.895.
The confusion matrix of this model is :
Confusion Matrix= $\begin{pmatrix} tn & fn \\ fp & tp \end{pmatrix}$=$\begin{pmatrix} 151 & 26 \\ 36 & 146 \end{pmatrix}$

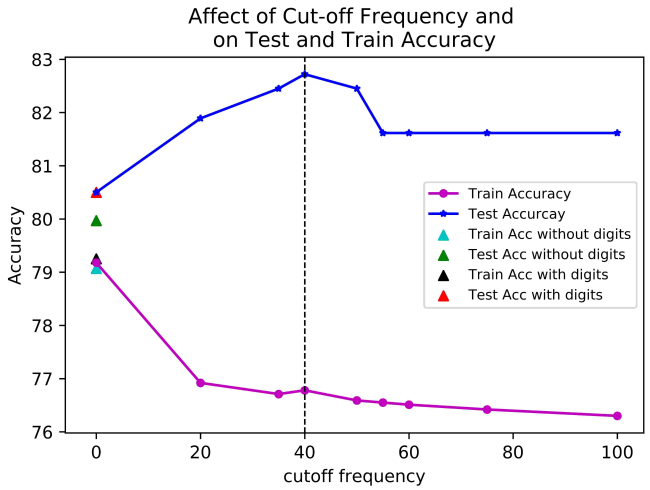



**Figure 2:** Effect on the Accuracy of the NB Model with different Cut-off Frequencies

Figure 2 shows the affect of the cutoff frequency on the accuracy obtained by the model. The maximum test accuracy is obtained at $f = 40$. This figure also shows that the digits in the tweets are of significant importance since the accuracy is higher when the model is trained with data without removing the digits.

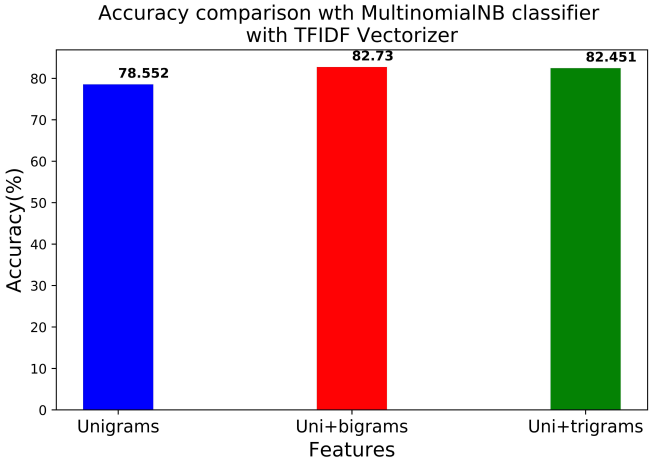
**Feature 2: Using Uni-grams and Bi-grams**

The next feature is to include pair of consecutive words in the dictionary so that the model can learn some context of the data. The new model now is trained on a dictionary that contains both uni-grams and bi-grams. By performing a similar experiment to use cut-off frequency on the bi-grams dictionary, it was found that the optimum cutoff is at $f = 10$. This model is able to achieve a test accuracy of **82.45%** with an AUC score of 0.902. Table 1 shows that this model is the best model in terms of AUC score. The confusion matrix of the model is :
Confusion Matrix= $\begin{pmatrix} tn & fn \\ fp & tp \end{pmatrix}$=$\begin{pmatrix} 148 & 29 \\ 34 & 148 \end{pmatrix}$

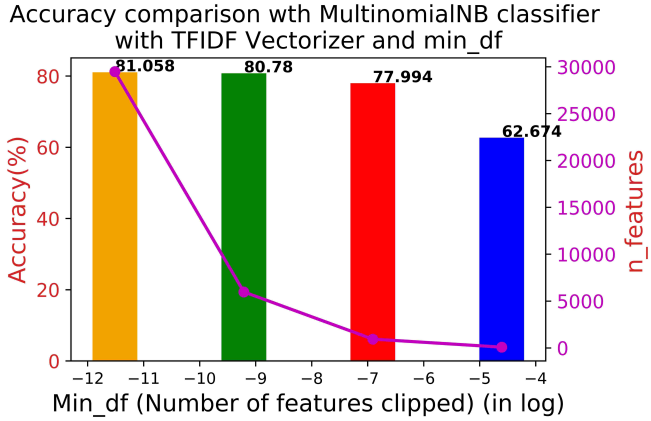
**Feature 3: Using Uni-grams and Tri-grams**

The next feature is to include three consecutive words together in the dictionary so that model can learn the context of the data. The new model is now trained on a dictionary with uni-grams as well as tri-grams. The optimum cut-off frequency is $f = 10$. This model is able to achieve a test accuracy of **83.28%** with an AUC score of 0.895. Table 1 shows that this model is the best in terms of achieving the best test accuracy. The confusion matrix of the model is:
Confusion Matrix= $\begin{pmatrix} tn & fn \\ fp & tp \end{pmatrix}$=$\begin{pmatrix} 149 & 28 \\ 32 & 150 \end{pmatrix}$

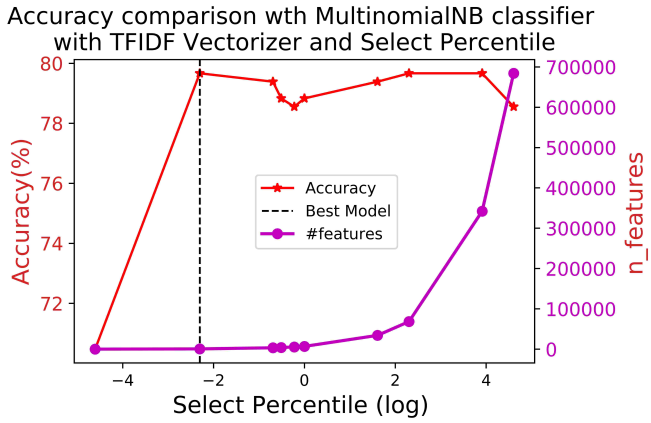
**This model trained with uni-grams and tri-grams is able to better predict the positive and the negative classes as compared to all the other models and also achieves the highest test accuracy.**




**(a)** Accuracy achieved by MultinomialNB model using features




**(b)** Accuracy achieved by MultinomialNB with different number of features




**(c)** Affect of Select Percentile on MultinomialNB Accuracy

**Figure 3:** Results obtained on SKLEARN MultinomialNB classifier model

**Results f and g:** This section will use the SKLEARN libraries to train MultinomialNB and Gaussian NB classi-

fiers. TFIDF or Term Frequency-Inverse Document Frequency reflects how important is a word to a particular document or data point in a collection of corpus. The following results are for Multinomial and Gaussian Naive Bayes Model trained on TFIDF features.

MultinomialNB Model:

The data is first transformed into a TFIDF vectorized matrix on which the model is fitted. We fit multinomial naive bayes model on different forms of TFIDF Vectors as shown in Table 2 and Figure 3a. The **time taken to train the model** with Uni-grams is 5x less that the time taken to train the model with Uni-grams+Tri-grams as the total number of features are higher. Figure 3b shows the experiment done to see the accuracy obtained with different number of features using MIN_DF. MIN_DF is a property of the TFIDF Vectorizer to retain only the most important features. MIN_DF prunes the words/features that appear too infrequently. The figures shows the degradation in accuracy with the decrease in the number of features used to fit the model.

| **Model** | **Test Acc(%)** |
|---|---|
| Multinomial Unigrams | 78.552 |
| Multinomial Uni+Bigrams | 82.73 |
| Multinomial Uni+Trigrams | 82.451 |
| Gaussian NB Unigrams | 77.16 |
| Gaussian NB Uni+Bigrams | 76.88 |
| Gaussian NB Uni+Trigrams | 76.6 |

**Table 2:** Accuracy obtained by training models using SKLEARN library with Unigrams, Bigrams and Trigrams

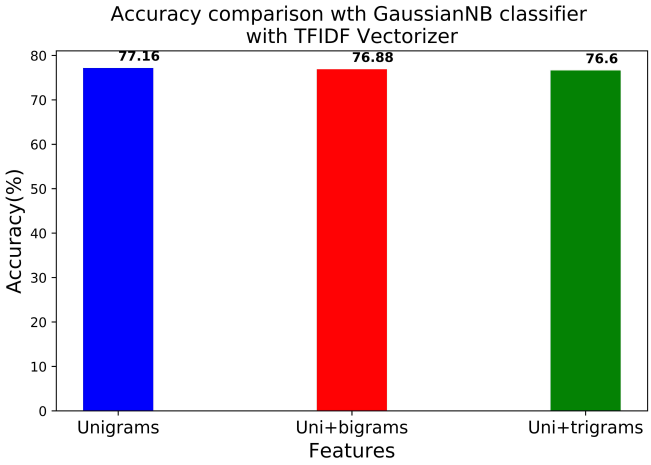
GaussianNB Model:

The Gaussian NB model of SKLEARN takes only dense arrays in contrast to MultinomialNB model which accepts the sparse matrix generated from TFIDF vectorization. On fitting the default features generated from TFIDF Vectorizer with Gaussian NB, memory error is generated because the default number of features are close to 700k. Hence, by using MIN_DF of TFIDF Vectorizer, an accuracy of 77.16%, as shown in Table 2, is obtained by using **min_df = 0.001** which means that the vectorizer ignores all the words that appear in less than 0.1% of the tweets. This retains around **937** most important features of the model. On using more than 2000 features, we get memory issue. So, maximum accuracy obtained for Gaussian NB model was 77.16%.

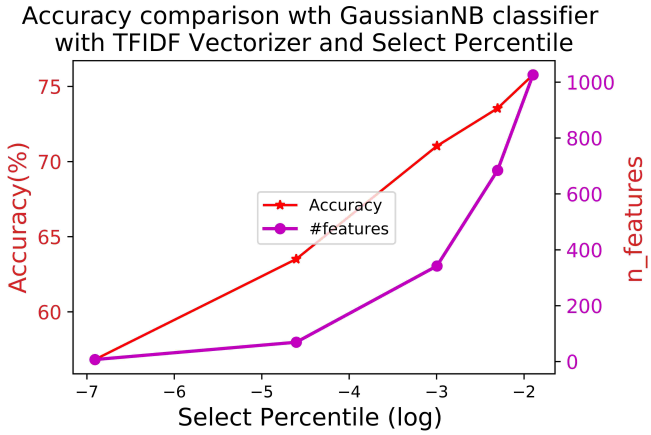
**Some experiments with Mini-Batch Training:** We also explored mini-batch training on Gaussian NB model. With a batch size of 1000, the maximum achieved accuracy was only 64%. Hence, we have not reported that in our results and report. Similar results of close to 52% accuracy were obtained on trying Truncated SVD and PCA. Further exploration is pending in this case.

Using SELECT PERCENTILE:

Select percentile is used to select only a percent of the total number of features obtained by TFIDF Vectorizer. The results shown in Figure 3c shows the affect of accuracy with different Select Percentiles ranging from 1% of the total features to 100% (where all features remain). We observe that the Multinomial NB model is able to achieve equivalent accuracy with 10% of the total features as well. For Gaussian, the Select Percentile had to be chosen in the range of 15% to 0.1% of the total features so as to tackle memory issues. Figure 4b shows the dip in accuracy with reducing number of features.
The time taken to train the model with different percentiles also reduces as the number of features reduce.




**(a)** Accuracy achieved by GaussianNB model using features




**(b)** Affect of Select Percentile on Gaussian Accuracy

**Figure 4:** Results obtained on SKLEARN GaussianNB classifier model

## 2.2 Question 2: Support Vector Machines

Support Vector Machines is a supervised machine learning algorithm that has been used to classify the different classes present in the FASHION MNIST dataset. The SVM optimization problem will be solved by two methods: by using a general purpose convex optimizer (CVXOPT) and by using a customized SVM Classifier library from SKLEARN. Fashion MNIST dataset is used for this problem. It has a total of 10 classes and around 22500 training samples, 2500 validation samples and 5000 test samples. The features taken a value in the range of

[0 255] which has been scaled down to [0 1].

### 2.2.1 Question 1. Binary Classification

This section documents the results obtained for binary classification problem. The entry number is $2018ANZ8503$, hence in this section, the results are for the classification of class 3 from class 4 i.e. dress vs. coat. The training, validation and test set for this problem have been sampled from the original test set according to the classes 3 4. We will used the soft-margin form of SVM given by Equation 4 by allowing data points to be on the wrong side of the margin or inside the margin. The parameter $C = 1.0$ in this formulation which sets the weight of the penalty ($\xi_i$)of the outliers in the data to the decision boundary.

The soft margin formulation of the SVM is :

$$\begin{aligned} \min_{w,b} \frac{1}{2} w^T w + C * \sum_i \xi_i \\ s.t. \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \\ \xi \geq 0 \end{aligned} \tag{4}$$

The Dual Formulation of the soft-margin SVM is:

$$\begin{aligned} \max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)} x^{(j)} > \\ s.t.\ 0 \leq \alpha_i \leq C, i = 1, 2, ..., m \\ \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \\ w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \\ b = Y - w^T.X \end{aligned} \tag{5}$$

**CVXOPT Package Formulation**
In this part, the SVM dual problem shown in Equation 5 is expressed in the form that is acceptable by the CVXOPT package given the Equation 6. So Equation 5 has to be expressed in the form of Equation 6.

$$\begin{aligned} \min_{\alpha} \frac{1}{2} \alpha^T P \alpha + q^T \alpha \\ s.t\ G\alpha \leq h \\ A\alpha - b = 0 \end{aligned} \tag{6}$$

The matrices that are input to the cvxopt optimizer for Linear and Gaussian kernel are given in Equation 7.

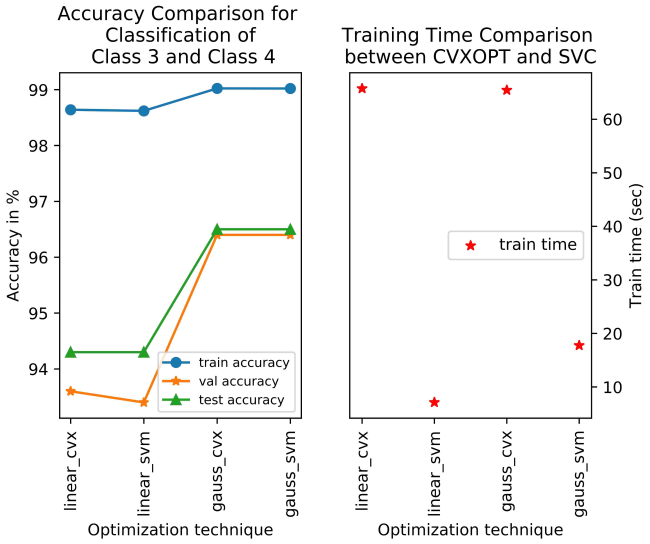
$$\begin{aligned} P = \sum_{i,j} y^{(i)} y^{(j)} < x^{(i)} x^{(j)} > \ Linear \\ P = \sum_{i,j} y^{(i)} y^{(j)} < \phi(x^{(i)}) \phi(x^{(j)}) > \ Gaussian \\ where, K(x,z) = \phi(x)^T \phi(z) \\ K(x,z) = exp(-\gamma * \|x - z\|^2) \\ q = [-1](m \times 1) \\ G = vstack([diag(-1)][diag(1)])(2m \times m) \\ h = hstack([0], C)(2m \times m) \\ A = y^T \\ b = 0 \end{aligned} \tag{7}$$

**Results 1.a: Linear Kernel with CVXOPT**
The SVM model trained by CVXOPT package by using inputs shown in Equation 7 with a Linear Kernel obtains a Train Accuracy of 98.64%, Validation Accuracy of 93.5% and Test Accuracy of 94.3%. The detailed results are shown in Table 3.

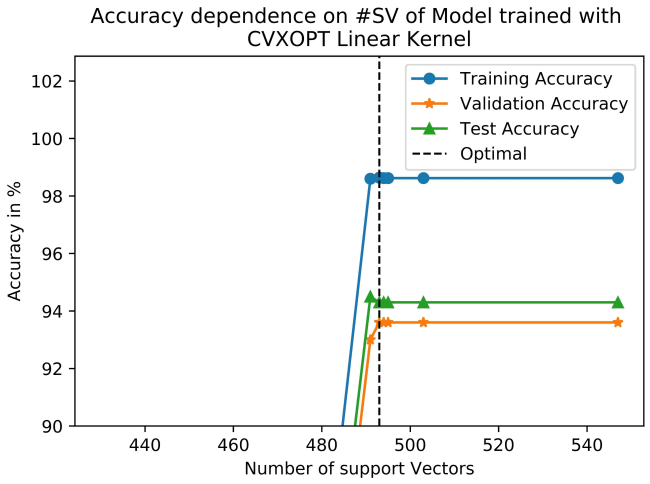
**Results 1.b: Gaussian Kernel with CVXOPT**
Equation 7 shows the form of the matrix P with Gaussian Kernel. The Dual problem is now optimized and solved in the Gaussian Space. So, the testing and the validation is also done in the Gaussian space. Hence, the input $x^{(i)}$ is replaced by $\phi(x^{(i)}$ everywhere. For calculation of the intercept term, w is calculated by transforming the training set data-points in the Gaussian space.
For prediction of a new point, the point is first taken to the Gaussian Space and then the prediction is made using $w^T + b$ where w is as in Equation 5.
The SVM Model trained with Gaussian Kernel achieves a Train Accuracy of 99.07%, Validation Accuracy of 96.4% and Test accuracy of 96.5%. The detailed results are in Table 3. **The model trained with Gaussian Kernel is able to achieve higher accuracy as compared to the linear Kernel. But this model has more number of support vectors as well as the training time is slightly higher than the linear kernel.**

| Model | Time(s) | #SV | b | Val Acc% | Test Acc% |
|---|---|---|---|---|---|
| CVXOPT Linear | 61.56 | [259,234] | -1.336 | 93.6 | 94.3 |
| SVMC Linear | 7.04 | [259,234] | -1.263 | 93.4 | 94.3 |
| CVXOPT Gaussian | 61.63 | [626,632] | -0.027 | 96.5 | 96.4 |
| SVMC Gaussian | 16.83 | [627,632] | -0.016 | 96.4 | 96.5 |

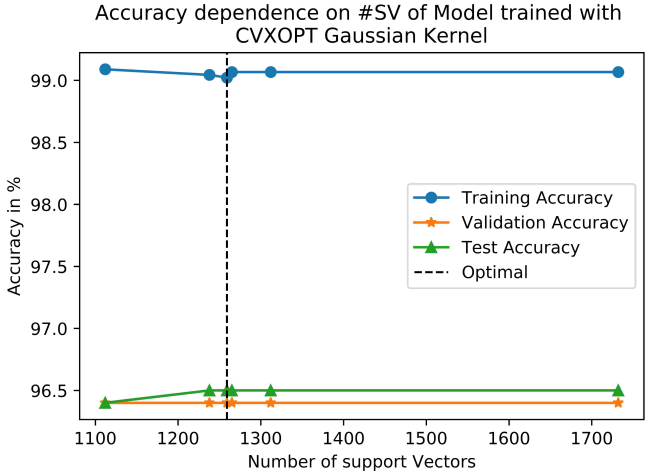
**Table 3:** Train Time, Number of Support Vectors, intercept and Accuracy obtained by different SVM models for Binary Classification




**(a)**




**(b)**



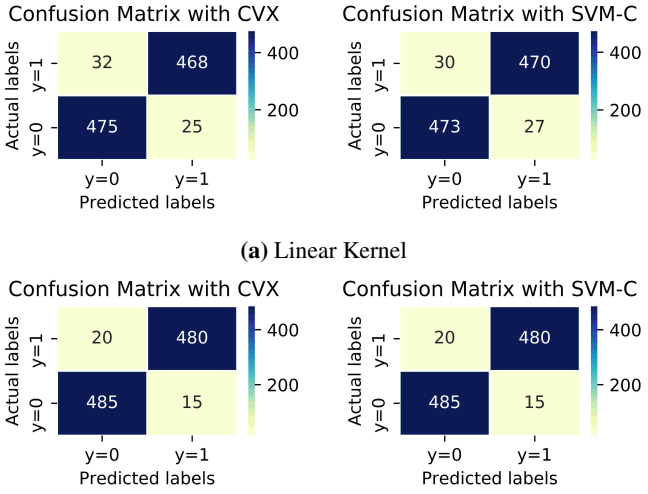

**(c)**

**Figure 5:** Comparison of Accuracy,Training Time, and accuracy dpeendence on the number of support Vectors between SVM classifier and CVXOPT Optimizer

**Results 1.c: SVM classifier from Scikit with Linear and Gaussian Kernel**

In this, a classifier optimized for SVM is used from the Scikit library package and trained with both Linear and Gaussian Kernel. The comparison of the results with CVXOPT implementation is shown in Table 3 and Figure 1. The model trained with SVM package converges 10x times faster than the CVXOPT package. The accuracy and the number of support vectors obtained by both the implementations is almost the same.

Figure 6 shows the confusion matrix obtained for the SVM classifier and the CVXOPT Optimizer for Linear and Gaussian Kernel which shows that with Linear Kernel, the model trained with CVXOPT is able to classify class 3 with more confidence. The SVM classifier performs equivalent for both the classes.




**(a)** Linear Kernel




**(b)** Gaussian Kernel

**Figure 6:** Confusion matrix between SVM classifier and CVXOPT Optimizer

### 2.2.2 Question 2. Multi-Class Classification

This section extends the classification from binary to multi-class classification. Only Gaussian Kernel will be used in this case as in the previous section.

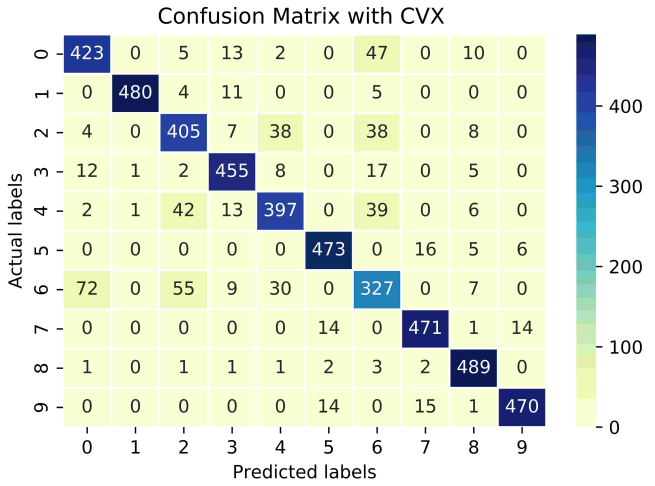
**Results 2.a:** In this, one model per pair of classes is trained so we have $\binom{k}{2}$ number of classifiers that differentiate between two classes. Here, $k = 10$. This is called a one-vs-one or OVO multi-class SVM classifier. The Accuracy obtained using CVXOPT and native Gaussian Kernel implementation is 87.6% on the validation set and 87.8% on the Test set.

| Model | Time(m) | Val Acc% | Test Acc% |
|---|---|---|---|
| CVXOPT Gaussian | 60 | 87.6 | 87.8 |
| SVMC Gaussian | 30 | 88.0 | 88.1 |

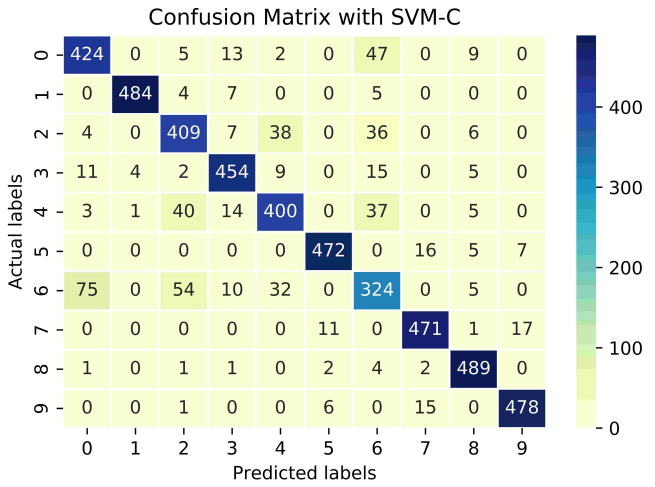
**Table 4:** Train Time, Number of Support Vectors, intercept and Accuracy obtained by different SVM models for Binary Classification

**Results 2.b:** A multi-class SVM trained using Scikit library with a gaussian kernel and OVO decision function achieved a validation accuracy of 88% and test accuracy of 88.1%. The comparison of the results obtained

by Scikit Library and CVXOPT is shown in Table 4. The training time is 2x times more for CVXOPT as compared to SVM optimized Scikit Library.

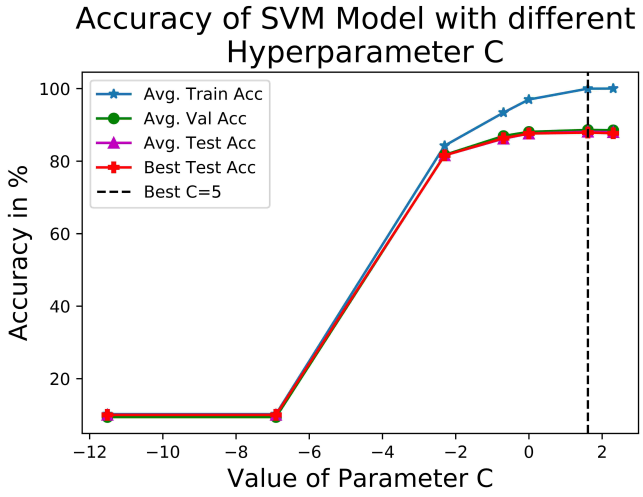



**(a)** Gaussian Kernel with CVXOPT

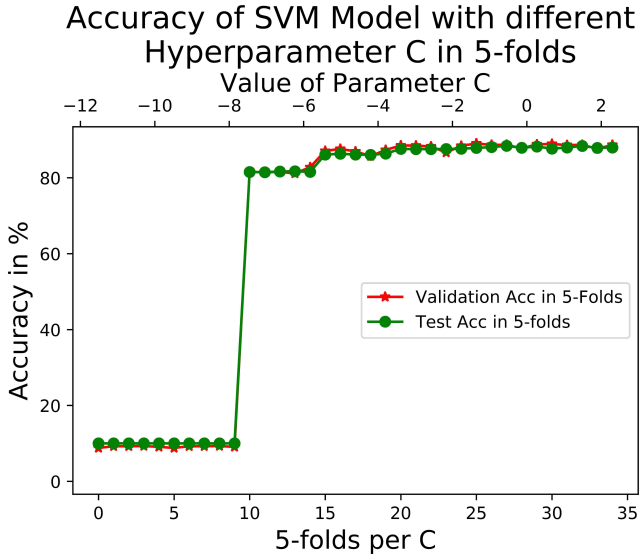



**(b)** Gaussian Kernel with SVM-C

**Figure 7:** Confusion matrix between SVM classifier and CVXOPT Optimizer for Multi-Class Classification




**(a)** Average Accuracy obtained for different C




**(b)** Accuracy achieved by differnt folds in 5-fold CV for different C

**Figure 8:** 5-Fold Cross-Validation for choosing the best hyperparameter C

**Results 2.c:** Figure 7 shows the confusion matrix obtained for multi-class classification for both the above methods on the test set. The most mis-classified articles are as follows:

- Class 0 mis-classified as Class 6 and vice-versa: Class 0 is T-shirt/Top and Class 6 is shirt. So the result does make sense because these classes are classifying similar categories, hence could easily be mis-classified.
- Class 2 mis-classified as Class 4 and Class 6 and vice-versa : Class 2 is Pullover, Class 4 - Coat and Class 6 is shirt. This also makes sense, since these are similar object classes.
- Class 7 mis-classified with Class 5 and Class 9 and vice-versa: Class 7 is Sneaker, Class 5 is Sandle and Class 9 is Ankle-boot which could easily be mis-classified due to similarity in features.

**Results 2.d:** Validation Set is typically used to estimate the best value of the model parameters (for example, C in our model) by randomly selecting small subset of data from the training set and then training the model on the remaining data. K-fold cross validation is done by dividing the data into K-folds and then treat each part as validation set once and training the model on (K-1) parts. The results presented here are to explore the model accuracy with 5-fold cross validation to choose the best hyperparameter C from $\{10^{-5}, 10^{-3}, 0.1, 0.5, 1, 5, 10\}$. classifiers are trained on these 5-folds and the average Validation and Test Accuracy is analyzed. The fold which gives the highest validation accuracy for a particular value of C is chosen as the desired classifier.
Figure **??** shows the trend of Accuracy for different values of C. The optimal model is obtained at **C=5**. The model that achieves the highest validation accuracy also obtains the highest test accuracy. Figure 8b shows the accuracy within the 5-folds for different values of C. This

shows that the model performance varies according to the data and cross-validation can be used to choose the best combination of train and validation set.

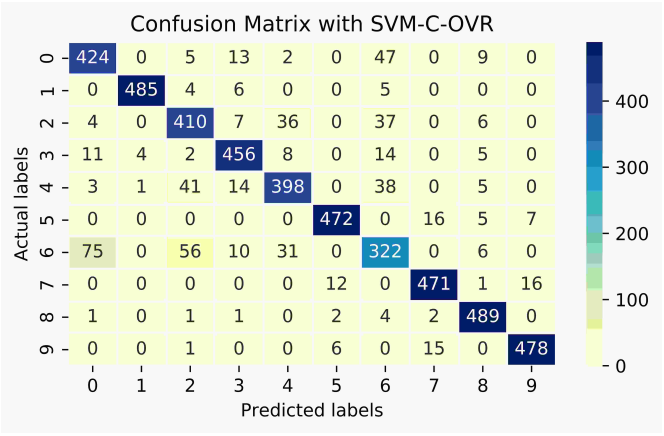



**Figure 9:** 5-Fold Cross-Validation for choosing the best hyperparameter C

### 2.2.3 Question 3:b - One-vs-Rest Multi-Class Classification

The one-vs-rest classification for multi-class trains one model per label/class considering that class as the positive class and all the other classes as negative classes. On training this kind of classifier using CVXOPT, we get memory issues, since the size of the matrix will become very large as we are trying to train on the complete dataset in contrast to training on 2 classes at a time in OVO strategy.
Hence, we present our results obtained by the SVM library. The training time is 30 mins and is similar to OVO strategy as the number of data-points to be looked at per classifier is large. The model is able to achieve a training accuracy of 96.92%, validation and test accuracy of 88.1%. This is similar to the OVO strategy. Figure 9 shows the confusion matrix obtained by training the SVM model by the OVR strategy and is similar to OVO.

## 3 Conclusions

This report is a summary of the second assignment done for Machine Learning course. We obtain a good understanding of how the algorithms that we talked about in this report are implemented and we also analyzed the affect of different hyper-parameters on the learning.