Assignment 2 Report

Priyadarshi Hitesh

8 March 2020

1 Text Classification

1.1 Naive Bayes Implementation

Implemented Naive Bayes from scratch without using any python module to classify the tweets sentiments as positive and negative. Used Laplace smoothing in order to avoid any case of zero probability.

1.2 Test Set Accuracies

Original Algorithm	**Random Baseline**	**Majority Baseline as Class4**
80.77994428969359%	53.2033426183844%	50.69637883008357%

Original Algorithm	**Random Baseline**	**Majority Baseline as Class0**
80.77994428969359 %	53.2033426183844%	49.30362116991643 %

Here, we had to compare accuracy of our algorithm with random prediction and majority prediction. Since in the given dataset, both class0 and class4 has occurred equal number of times. So, our accuracy is compared with both the cases taking one as majority class at a time.

There is an improvement of approximately 30% in accuracies as compared to random prediction and majority prediction.

1.3 Confusion Matrix

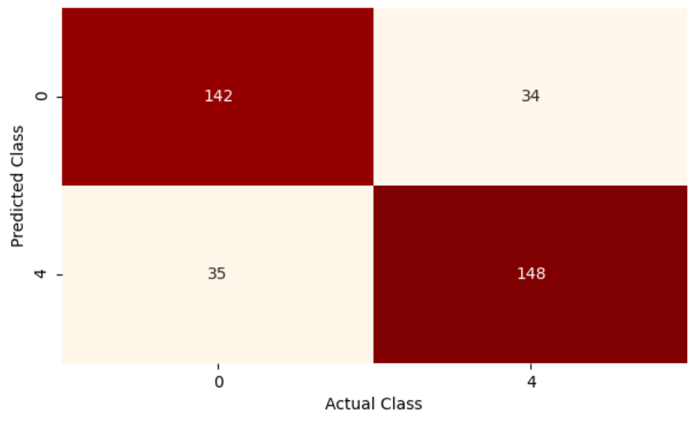


Figure 1: Confusion Matrix For Test Data

1.3.1 Observations

- Class4 has highest value of diagonal entry.
- My Algorithm has incorrectly predicted 34 test examples as class0 which belongs to class4.
- My algorithm has incorrectly predicted 35 test examples as class4 which belongs to class0.
- My algorithm has correctly predicted for 290 test data examples.

1.4 Cleaning Given Dataset

From our original dataset, we have removed stopwords such as 'of', 'the', 'and' etc. Stemming is also performed on the original dataset. The clean data which is obtained after stemming and stopwords removal is used to train the model and better test accuracy is obtained which we can see from table.

Test Accuracy on original Data	**Test Accuracy on cleaned Data**
80.77994428969359%	81.33704735376045%

We can see from the above table how test accuracy is improved after training on cleaned data.

1.5 Feature Engineering

1.5.1 Unigrams with Bigrams

In this, we have added bigrams along with Unigrams in vocabulary on cleaned data which we had obtained after stemming and stopword removal.

Test Accuracy on original Data	**Test Accuracy on Unigram With Bigram**
80.77994428969359 %	83.84401114206128 %

Test Accuracy on cleaned Data	**Test Accuracy on Unigram with Bigram**
81.33704735376045 %	83.84401114206128 %

We can clearly make comparison from the tables above. After adding bigram with unigram as a feature, our test set accuracy is increased by approximately 3% as compared to original data and 2% as compared to cleaned data.

1.5.2 Parts Of Speech Tagging

In this on the data obtained after cleaning, every word of tweets are tagged to their respective POS. Then while implementing naive bayes we have increased weightage of some words according to their POS and trained the model again. We have tried different combination and result is shown below for the combination where we have got some improved accuracy than what we were getting initially.

Test Accuracy on original Data	**Test Accuracy after Parts of Speech Tagging**
80.77994428969359 %	82.79235235432528 %

Test Accuracy on cleaned Data	**Test Accuracy after Parts of Speech Tagging**
81.33704735376045 %	82.79235235432528 %

Accuracy has increased a little bit, we can clearly see from the table above.

Weightage of every adverb is increased to thrice its earlier weightage and weightage of every adjective is increased to twice its earlier weightage.

From the above different feature engineering, bigram with unigram has performed better as compared to parts of speech tagging, though the difference between both is not that significant.

1.6 TF-IDF

1.6.1 TF-IDF with Gaussian NB

In this features are generated through TF-IDF and we have trained the model through Gaussian Naive Bayes.

Test Accuracy on original Data	**Test Accuracy using TF-IDF**
80.77994428969359 %	49.58217270194986 %

We can see after using TF-IDF our test set accuracy has dropped drastically. So, i have tried using TF-IDF with MinDf and we can see the result below.

Test Accuracy on original Data	**Test Accuracy using TF-IDF with MinDf**
80.77994428969359 %	78.55153203342619 %

So, basically best possible accuracy we got from TF-IDF is 78.55153203342619 % which is still less than what we had achieved through our implementation on original data.

1.6.2 TF-IDF Select Precentile with Gaussian NB

In this features are generated through TF-IDF with select percentile. We can get different features by varying select percentile, for smaller values of select percentile we will get less features or basically features having more importance. Below are the results gotten for different value of Select percentile.

Test Set Accuracies

On original Data	**Through TFIDF**	**Through Select 10 Percentile**
80.77994428969359 %	78.55153203342619 %	71.3091922005571 %

On original Data	**Through TFIDF**	**Through Select 5 Percentile**
80.77994428969359 %	78.55153203342619 %	64.62395543175488 %

On original Data	**Through TFIDF**	**Through Select 2 Percentile**
80.77994428969359 %	78.55153203342619 %	60.16713091922006 %

On original Data	**Through TFIDF**	**Through Select 35 Percentile**
80.77994428969359 %	78.55153203342619 %	77.71587743732591 %

As we can see from the above tables, the best accuracy we have got through Select percentile is when select precentile is 35 and the accuracy is almost same as what we had got through TF-IDF without select percentile.

Time Taken For Training Models in (Seconds)

Our algorithm	**TFIDF**	**TFIDF through Select 35 Percentile**
33.3402736	5461.23242351	1753.2434565

Our algorithm	**TFIDF**	**TFIDF through Select 10 Percentile**
33.3402736	5461.23242351	611.35862

From the above tables, we can compare the time taken by different models for training the model.

Our algorithm	**TFIDF**	**TFIDF through Select 35 Percentile with MinDf**
33.3402736	5461.23242351	32.96500

Our algorithm	**TFIDF**	**TFIDF through Select 10 Percentile with MinDf**
33.3402736	5461.23242351	28.96500

The above tables contains time taken when MinDf is used, it takes comparatively less time than what it took without select percentile.

1.7 ROC Curve

1.7.1 For Test Data after trained on training data

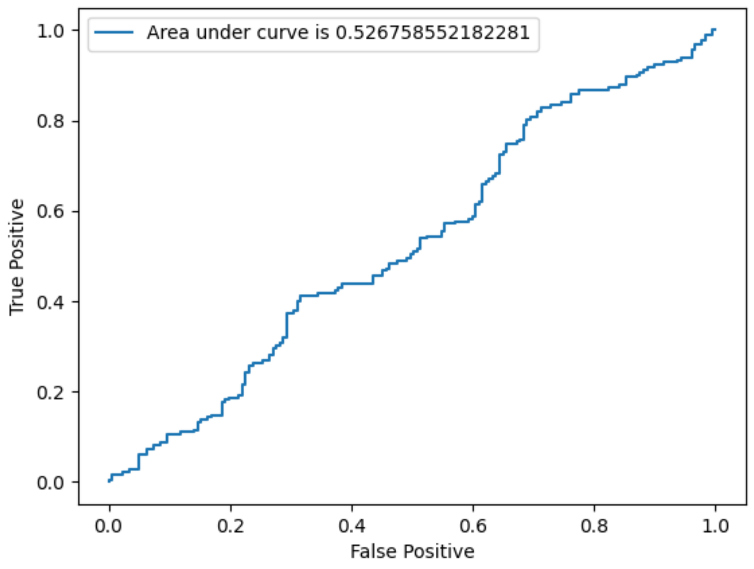


Figure 2: ROC Curve Through Model Trained On Original Data

As we can see Area under curve is near 0.5, which clearly tells that our model is not able to classify the data properly. It is more like some random prediction.

1.7.2 For Cleaned Test Data after training on cleaned Train Data

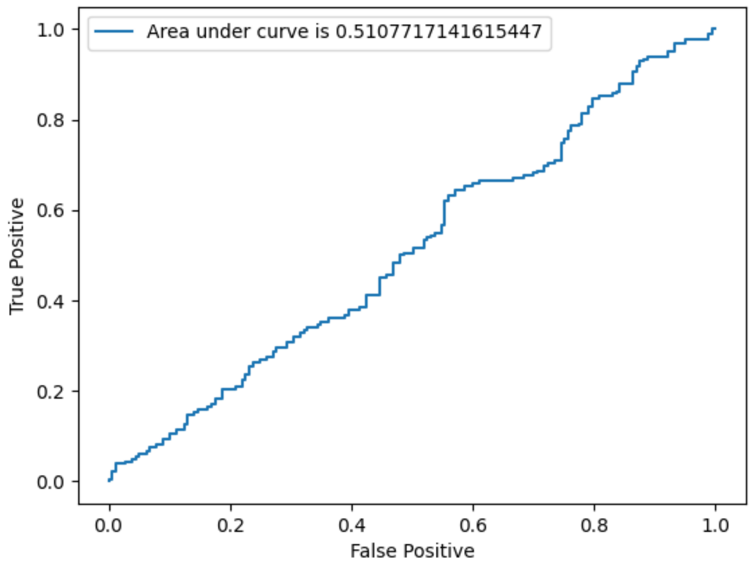


Figure 3: ROC Curve Through Model Trained On Cleaned Data

In this diagram also, area under curve is near 0.5, so even after cleaning the data our model is not able to correctly classify the data properly.

1.7.3 For cleaned Test Data after using TF-IDF without MinDf

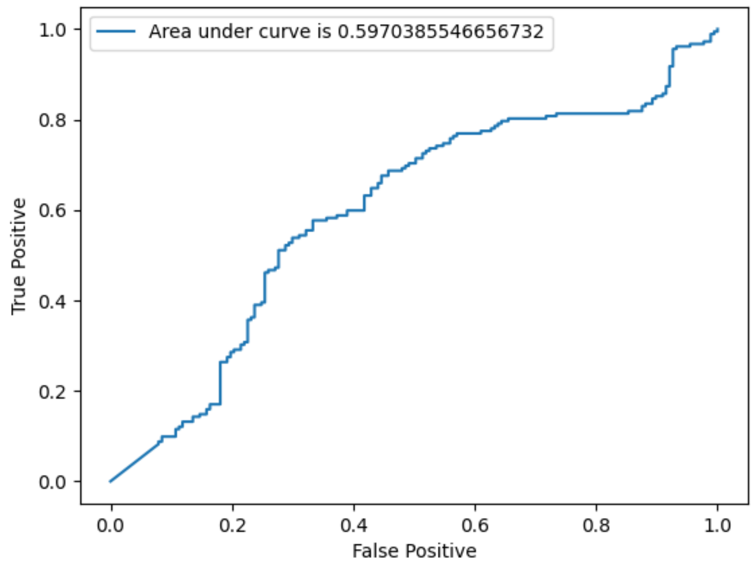


Figure 4: ROC Curve Through Model Trained using TF-IDF without MinDf

In this we have used TF-IDF and area under curve is near 0.6 which has clearly improved from earlier case, but it'd still be considered poor classification. MinDf feature of TF-IDF isn't used in this case.

1.7.4 For Cleaned Test Data after using TF-IDF with MinDf

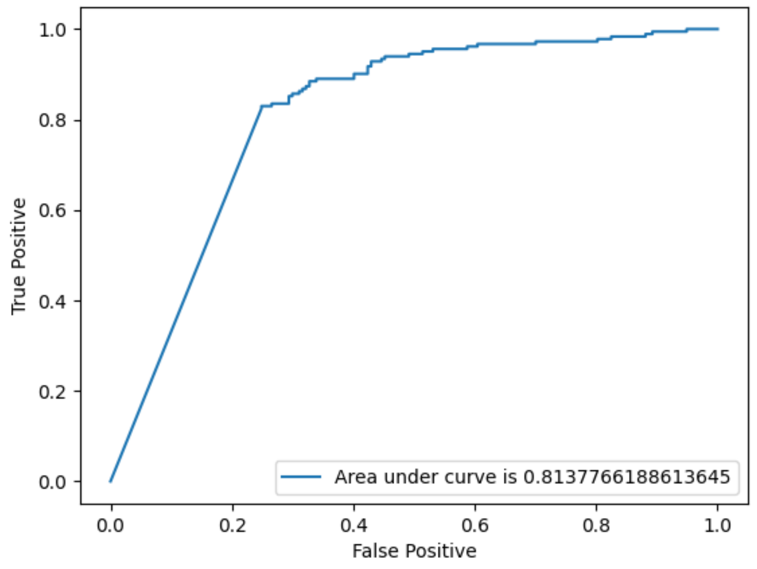


Figure 5: ROC Curve Through Model Trained using TF-IDF with MinDf

In this we have used TF-IDF with MinDf and our area under curve is increased to 0.8, which clearly tells that this model has been classifying data correctly far better than earlier ones. Infact, this is the best Model obtained so far.

1.7.5 For Cleaned Test Data after using Select 35 Percentile without MinDf

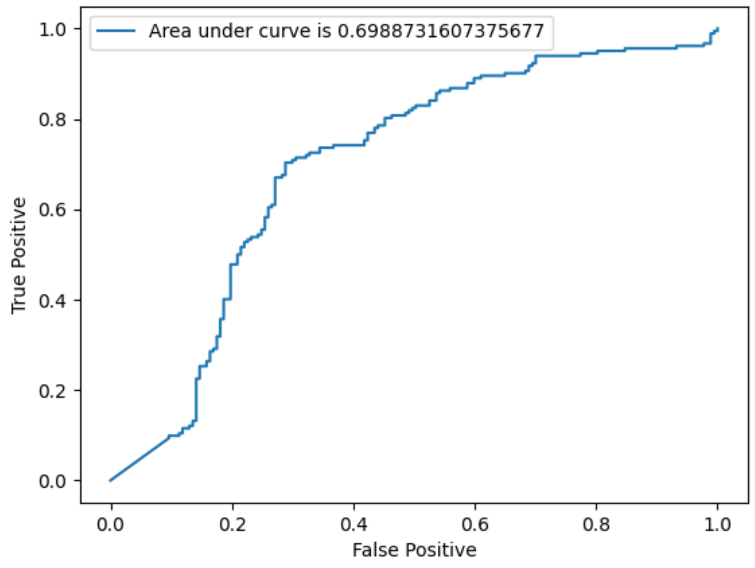


Figure 6: ROC Curve Through Model using TF-IDF Select Percentile

This model is using TF-IDF select percentile functionality for features, area of curve obtained is near 0.7 which would be considered as very average performance and it isn't classifying the data more accurately.

1.7.6 For Cleaned Test Data after using Select 35 Percentile with MinDf

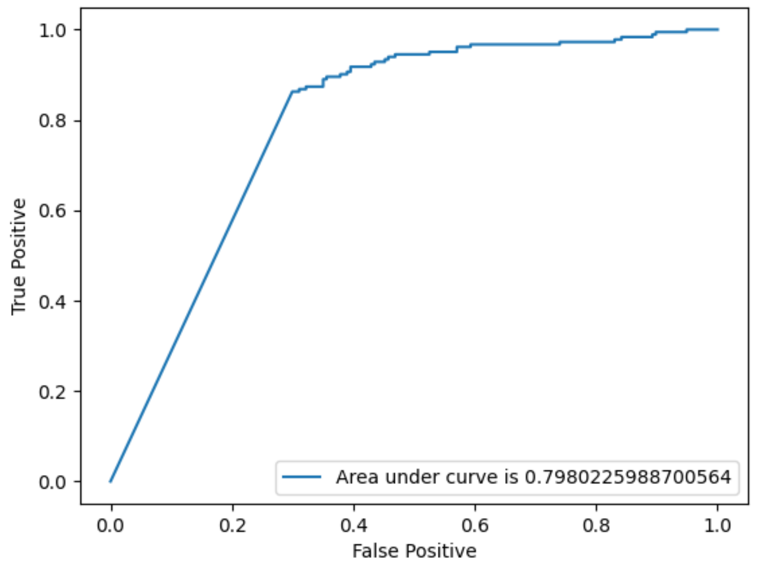


Figure 7: ROC Curve Through Model using TF-IDF MinDf Select Percentile

This model is using TF-IDF select percentile with MinDf and we have got area under curve near 0.8 which is clearly quite good performance. So it is classifying data more correctly that select percentile without MinDf. We have selected percentile as 35 as it was giving better accuracy that other ones.

1.7.7 For Cleaned Test Data after using Bigram with Unigram

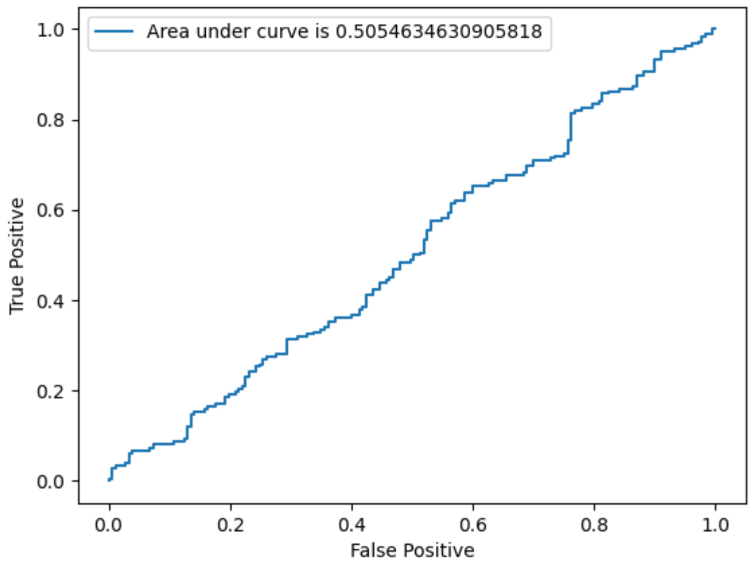


Figure 8: ROC Curve Using Unigram And Bigram As Features

This model is trained with feature as unigram and bigram together, and area under curve is near 0.5 which clearly depicts that this model hasn't been able to classify the data properly and it has performed more like some random prediction.

2 Fashion MNIST Article Classification

2.1 Binary Classification

2.1.1 Accuracies of Linear SVM using CVXOTP Package

In this we have formulated our dual objective into a new problem that can use CVXOTP package in order to solve our problem.

This model is trained for class 0 and class 9 as two classes where class 0 is considered positive and class 9 negative.

The values of C is 1.0 for this problem.

The value of b obtained is 0.76385055.

The accuracies obtained on validation set and test set are as follows:

Validation Accuracy	**Test Set Accuracy**
99.8 %	100.0 %

2.1.2 Accuracies of Gaussian SVM using CVXOTP Package

In this also we have formulated our dual objective for gaussian kernels into a new problem that can use CVXOTP package in order to solve our problem.

This model is trained for Class 0 and Class 9 as two classes where class 0 is considered positive and class 9 negative.

The values of C and γ is 1.0 and 0.05 for this problem.

The accuracies obtained on validation set and test set are as follows:

Validation Accuracy	**Test Set Accuracy**
100 %	100.0 %

If we compare this gaussian svm model to earlier linear svm model, we can only change we have obtained is in Validation Accuracy. Validation Accuracy has improved from 99.8 % to 100 % while Test Accuracy has remained the same.

This clearly tells us that our this one to one for class 0 and class 9 is clearly able to classify the data perfectly.

2.1.3 Accuracies the Scikit Learn Linear and Gaussian SVMs

Accuracies on Linear SVM using Scikit Learn

Validation Accuracy	**Test Set Accuracy**
99.8 %	100.0 %

We are getting perfect accuracy on validation and test set, which tells that SkLearn linear model is able to classify the data properly.

Accuracies on Gaussian SVM using Scikit Learn

Validation Accuracy	**Test Set Accuracy**
100 %	99.8 %

We are getting perfect accuracy on validation and test set, which tells that SkLearn gaussian model is able to classify the data properly

Comparison of b for Linear Kernel

Through CVXOTP Package	**Through Scikit Learn**
0.76385055	0.76384274

As we can see from above, value of b is almost same for both.

Comparison of b for Gaussian Kernel

Through CVXOTP Package	**Through Scikit Learn**
0.031048543371928056	0.18758554

As we can see from above, value of b is varying a bit and a greater value of b is observed through Scikit Learn.

Comparison of Number of Support Vectors for Linear Kernel

JJS height**Through CVXOTP Package**	**Through Scikit Learn**
57	57

Threshold to get support vectors is 1e-5 when we are using CVXOTP Package.

Both the models has given exactly same number of support vectors in case of linear kernel.

Comparison of Number of Support Vectors for Gaussian Kernel

Through CVXOTP Package	Through Scikit Learn
847	826

Threshold to get support vectors is 1e-5 when we are using CVXOTP Package.

Time Comparison For Linear Kernel in(Seconds)

Through CVXOTP Package	Through Scikit Learn
83.01809334754944	0.22699189186096191

Scikit Learn is training the model in fraction of seconds, while through CVX-OTP time taken is more than a minute.

Time Comparison For Gaussian Kernel in(Seconds)

Through CVXOTP Package	Through Scikit Learn
48.713380098342896	3.4515771865844727

Implementing through CVXOTP package is taking 16 times more time than what Scikit Learn is taking to train for Gaussian Model.

2.2 Multi-Class Classification

2.2.1 One-vs-One Multi Class SVM

In this we have implemented one vs one multi class SVM and classified the data using the model. Classification is based on maximum number of votes a class has got and if many classes has gotten same number of votes then their score is used for tie breaking.

The accuracies obtained are as follows:

Validation Accuracy	Test Set Accuracy
84.96 %	85.08 %

2.2.2 Multi Class SVM using Scikit Learn

In this we have implemented Multi Class SVM with the help of Scikit Learn. Then, the trained model is used to predict the accuracy of test and validation data.

The accuracies obtained are as follows:

Validation Accuracy	**Test Set Accuracy**
87.88 %	88.08 %

The model learned using Scikit Learn for Multi Class SVM has performed better than our model. Validation and test set accuracy is increased by approximately 3 %.

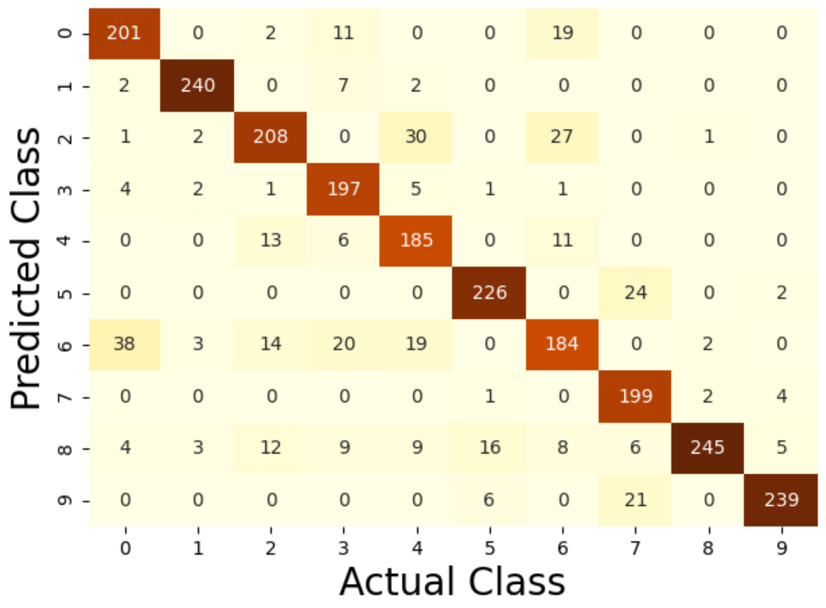
Time Comparison For Multi Class SVM in(Minutes)

Through CVXOTP Package	**Through Scikit Learn**
37.39	6.17

Clearly, we can see from the above table Scikit Learn is faster than our model, it is taking 1/6th time of what our model took for training.

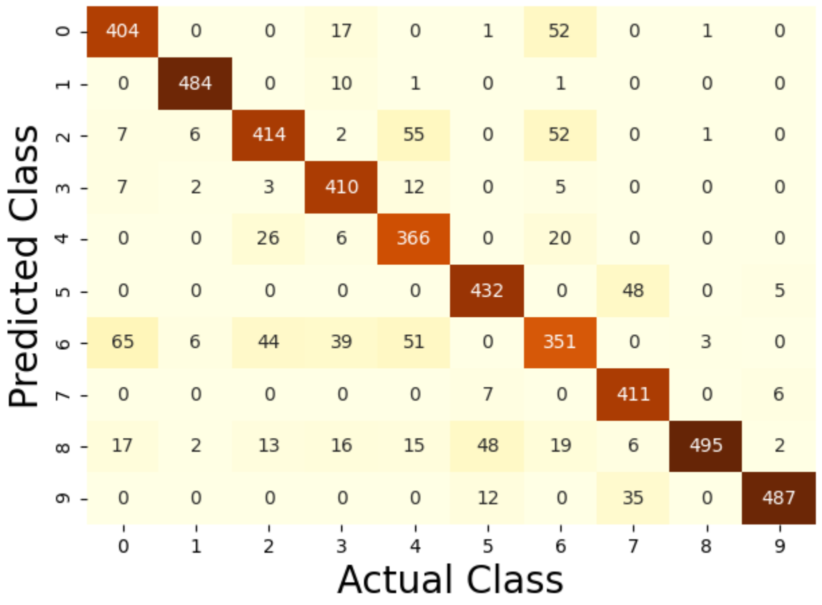
2.2.3 Confusion Matrix

<u>**Through CVXOTP Package**</u>



	0	1	2	3	4	5	6	7	8	9
0	201	0	2	11	0	0	19	0	0	0
1	2	240	0	7	2	0	0	0	0	0
2	1	2	208	0	30	0	27	0	1	0
3	4	2	1	197	5	1	1	0	0	0
4	0	0	13	6	185	0	11	0	0	0
5	0	0	0	0	0	226	0	24	0	2
6	38	3	14	20	19	0	184	0	2	0
7	0	0	0	0	0	1	0	199	2	4
8	4	3	12	9	9	16	8	6	245	5
9	0	0	0	0	0	6	0	21	0	239

Figure 9: Validation Set Confusion Matrix for model trained using CVXOTP package



	0	1	2	3	4	5	6	7	8	9
0	404	0	0	17	0	1	52	0	1	0
1	0	484	0	10	1	0	1	0	0	0
2	7	6	414	2	55	0	52	0	1	0
3	7	2	3	410	12	0	5	0	0	0
4	0	0	26	6	366	0	20	0	0	0
5	0	0	0	0	0	432	0	48	0	5
6	65	6	44	39	51	0	351	0	3	0
7	0	0	0	0	0	7	0	411	0	6
8	17	2	13	16	15	48	19	6	495	2
9	0	0	0	0	0	12	0	35	0	487

Figure 10: Test Set Confusion Matrix for model trained using CVXOTP package

Articles that are mis-classified into each other most often are:

1. Shirt and T-shirt are mis-classified into each other most of the times, their respective classes are 6 and 0.

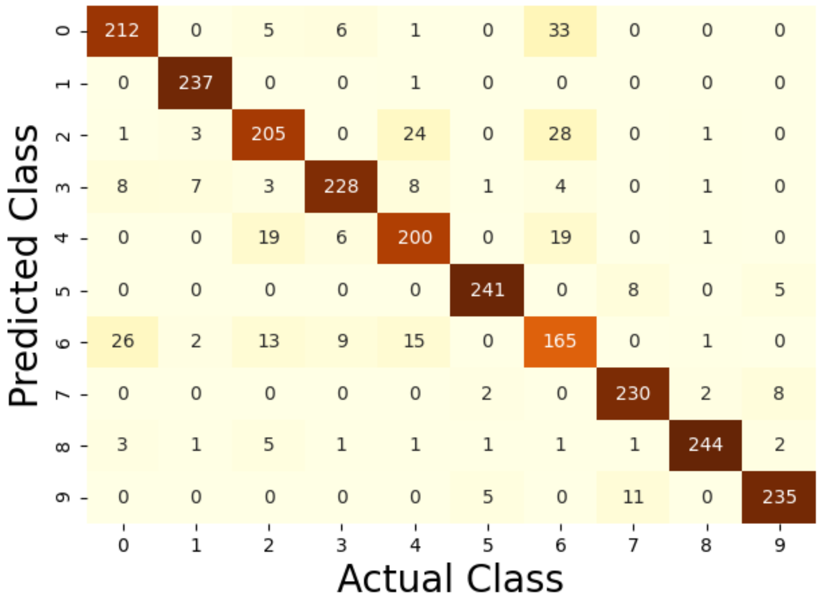
2. Shirt and Pullover are mis-classified into each other most of the times, their respective classes are 6 and 2.

3. Pullover and Coat are mis-classified into each other most of the times, their respective classes are 2 and 4.

4. Coat and Shirt are mis-classified into each other most of the times, their respective classes are 4 and 6.

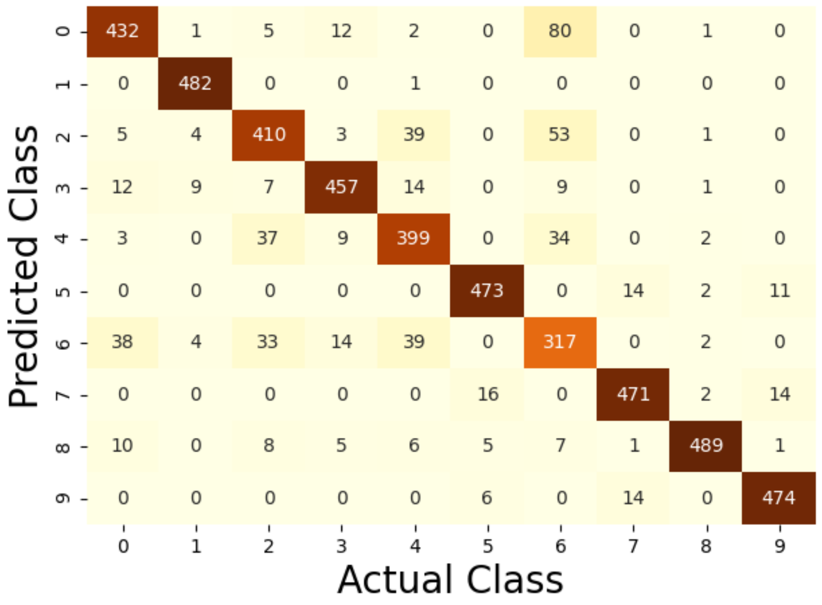
As we can see from the observation, classes which are mis-classified into each other are shirts and T-shirts which somewhat looks same, so yes this makes an intuitive sense why they are being mis-classified into each other and same analogy for rest of all combination as well.

Through Scikit Learn



Predicted Class \ Actual Class	0	1	2	3	4	5	6	7	8	9
0	212	0	5	6	1	0	33	0	0	0
1	0	237	0	0	1	0	0	0	0	0
2	1	3	205	0	24	0	28	0	1	0
3	8	7	3	228	8	1	4	0	1	0
4	0	0	19	6	200	0	19	0	1	0
5	0	0	0	0	0	241	0	8	0	5
6	26	2	13	9	15	0	165	0	1	0
7	0	0	0	0	0	2	0	230	2	8
8	3	1	5	1	1	1	1	1	244	2
9	0	0	0	0	0	5	0	11	0	235

Figure 11: Validation Set Confusion Matrix for model trained using Scikit Learn



	0	1	2	3	4	5	6	7	8	9
0	432	1	5	12	2	0	80	0	1	0
1	0	482	0	0	1	0	0	0	0	0
2	5	4	410	3	39	0	53	0	1	0
3	12	9	7	457	14	0	9	0	1	0
4	3	0	37	9	399	0	34	0	2	0
5	0	0	0	0	0	473	0	14	2	11
6	38	4	33	14	39	0	317	0	2	0
7	0	0	0	0	0	16	0	471	2	14
8	10	0	8	5	6	5	7	1	489	1
9	0	0	0	0	0	6	0	14	0	474

Figure 12: Test Set Confusion Matrix for model trained using Scikit Learn

Articles that are mis-classified into each other most often are:

1. Shirt and T-shirt are mis-classified into each other most of the times, their respective classes are 6 and 0.
2. Shirt and Pullover are mis-classified into each other most of the times, their respective classes are 6 and 2.
3. Pullover and Coat are mis-classified into each other most of the times, their respective classes are 2 and 4.
4. Coat and Shirt are mis-classified into each other most of the times, their respective classes are 4 and 6.

As we can see from the observation, classes which are mis-classified into each other are shirts and T-shirts which somewhat looks same, so yes this makes an intuitive sense why they are being mis-classified into each other and same analogy for rest of all combination as well.

2.2.4 K-Fold Cross Validation

In this problem we have done 5-fold validation to estimate best value of C among the given 5 values i.e. 1e-5, 1e-3, 1, 5, 100 for the case of gaussian kernel.

5-Fold Cross Validation Accuracies by varying C

C=1e-5	**C=1e-3**	**C=1**	**C=5**	**C=10**
56.65 %	56.65 %	87.87 %	88.44 %	88.43 %

Test Set Accuracies by varying C

C=1e-5	**C=1e-3**	**C=1**	**C=5**	**C=10**
57.36 %	57.36 %	88.08 %	88.28 %	88.24 %

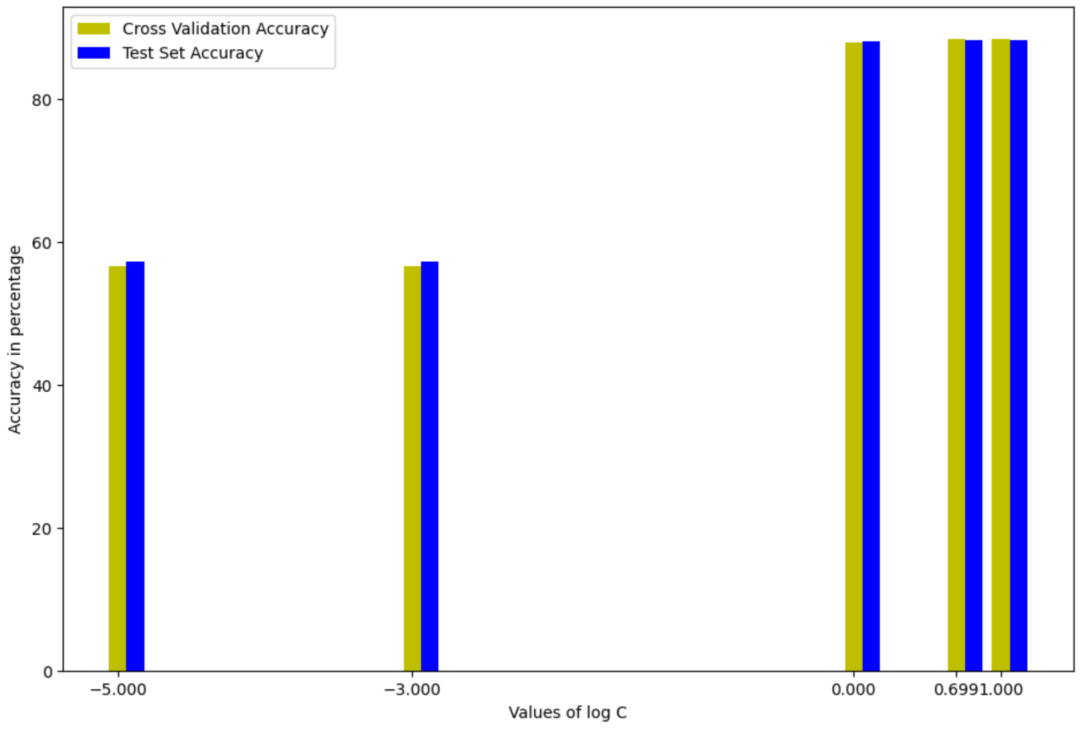


Figure 13: Plot of Cross Validation and Test Set Accuracy

Observations:

1. C=5 gives the best cross validation Accuracy.
2. For C=5, test set accuracy is also maximum.
3. So for same value of C, we are getting maximum accuracy in case of both.