COL 774: Assignment2

VIVEK SINGH

2019MCS2574

1 Text Classification: Naive Bayes

1.1 Implement Naive Bayes

Data Description : Class 4: Positive Class, Class 0: Negative class

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
Polarity	TweetID	Date of tweet	query	User	Tweet

- Laplace smoothing C = 1
- Took log of the parameters to avoid underflow
- As asked data split only on White-spaces, '.' and ','
- **Accuracy**

Test	Train
80.7799 %	84.9328 %

1.2 Random & Majority Prediction

- **Random Prediction**

Attempt 1	Attempt 2	Attempt 3
50.6963 %	48.7465 %	51.253 %

- **Majority Prediction**:

  Both Class are equally distributed in Train Data.

  Accuracy on train data:

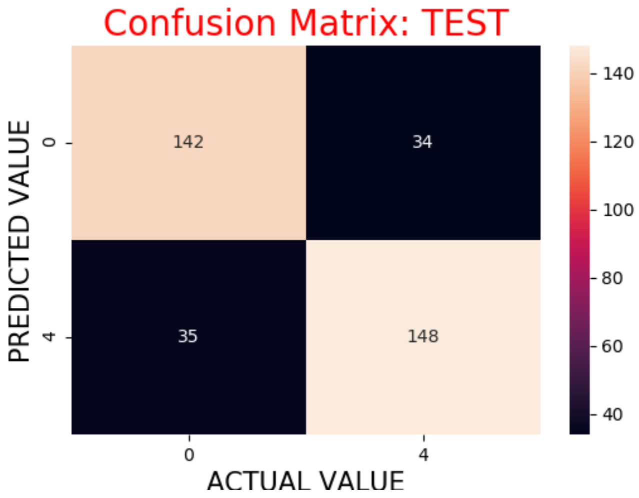
Class 0	Class 4
50.0 %	50.0 %

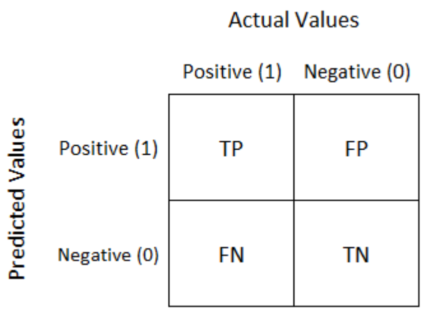
Accuracy on test data:

Class 0	Class 4
50.6963 %	49.3036 %

Our implementation of Naive Bayes gives significant improvement on both test and train data set over base-line of Random as well as Majority prediction.

1.3 Confusion Matrix





- True Positive: 142, implies our algo classified 142 positive class correctly.
- False Postive: 34, implies our algo classified 34 as postive class but actually they were of negative class.
- False Negative: 35, implies our algo classified 35 as negative class but actually they were of positive class.
- True Negative: 148, implies our algo classifed 149 negative class correctly.
- Diagonal elements show how much classification of the classifiers are correct.
- Non-doagonal elements show how much classification of the classsifier are incorrect.

1.4 Cleaning Data

- **Cleaning of the URLs:** since we need to classify tweets sentiment as postive aand negative we can remove the URL's as they do not convey any sentiment in the tweet.
- **Clean Punctuation:** Removing unwanted English punctuation form the tweets.
- **Tokenization:** Splitting the tweets into tokes by white-spaces, ',' and '.'.
- **Stopping :** Here we remove all the english stop words from the tweet using **NLTK** english stopper.
- **Stemming :** Have performed stemming using **SnowballStemmer** for English language.
- **User Name :** Removed all the user names from each tweet in data-set .

Accuracy Comparison:

Test Data:

Original Data	Clean Data
80.7799 %	82.1727 %

Comment: After removing the unwanted words and stemming the words in tweet we get a better set of data which convey sentiment more than the previous original data. Hence we could observe improvement in accuracy in our Cleaned Data.

1.5 Feature Engineering

- Feature engineering

  NLTK Part of Speech Tagging

  Bi-grams

- Comparing test accuracy:

Original Data	Clean Data	Feature Engineering
80.7799 %	82.1727 %	83.5654 %

- The features are build upon the cleaned data-set obtained from part 4.
- Used NLTK POS_TAG we tagged each word according to english .
- **VERB :**a word used to describe an action, state, or occurrence, and forming the main part of the predicate of a sentence, such as hear, become, happen.
- Hence increased the weight of the verb in each tweet.
- Time taken to tag each word in tweet in data-set took 20 min.
- Then we used Bi-grams together with POS_tag to train the data-set and then test on appropriate cleaned test Data.

1.6 TF-IDF features with Gaussian Naive Bayes Model

- Using Scikit-Learn's TFIDF-Vectorizer and GaussianNB

  Accuracy on Test Data-set:

Clean Data	TF-IDF	SelectPercentile =5	SelectPercentile =10	SelectPercentile =20
82.1727 %	49.897 %	62.674 %	56.545 %	50.974 %
3 min	96 min	4 min	9 min	17 min

Clean Data	TF-IDF $\min_d f = 5e^{-4}$	SelectPercentile =5	SelectPercentile =10	SelectPercentile =20
82.1727 %	78.830 %	65.73 %	74.651 %	77.437 %
3 min	1 min	3 sec	10 sec	15 sec

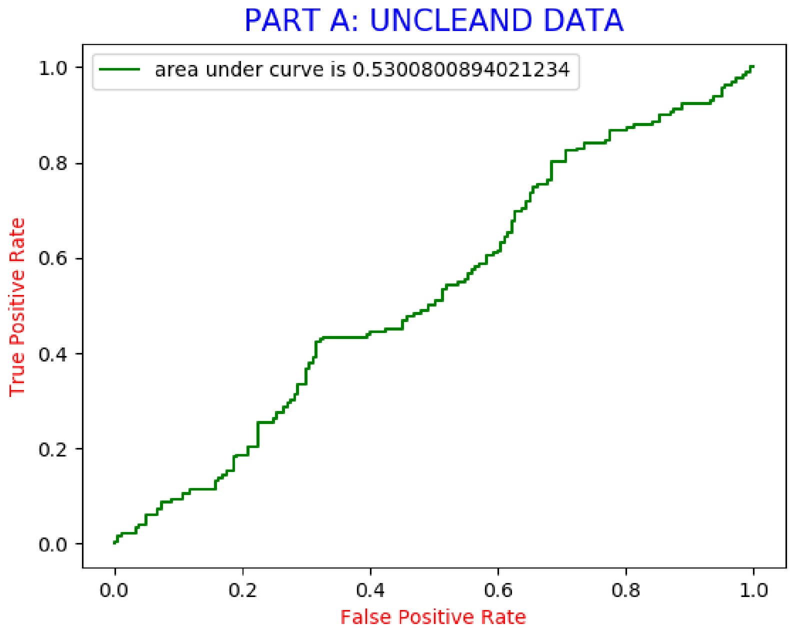
Clean Data	TF-IDF $\min_d f = 6e^{-4}$	SelectPercentile =5	SelectPercentile =10	SelectPercentile =20
82.1727 %	80.501 %	64.62 %	71.587 %	76.880 %
3 min	30 sec	3 sec	6 sec	10 sec

Clean Data	TF-IDF $\min_d f = 7e^{-4}$	SelectPercentile =5	SelectPercentile =10	SelectPercentile =20
82.1727 %	81.058 %	65.459 %	71.030 %	76.323 %
3 min	35 sec	3 sec	6 sec	10 sec

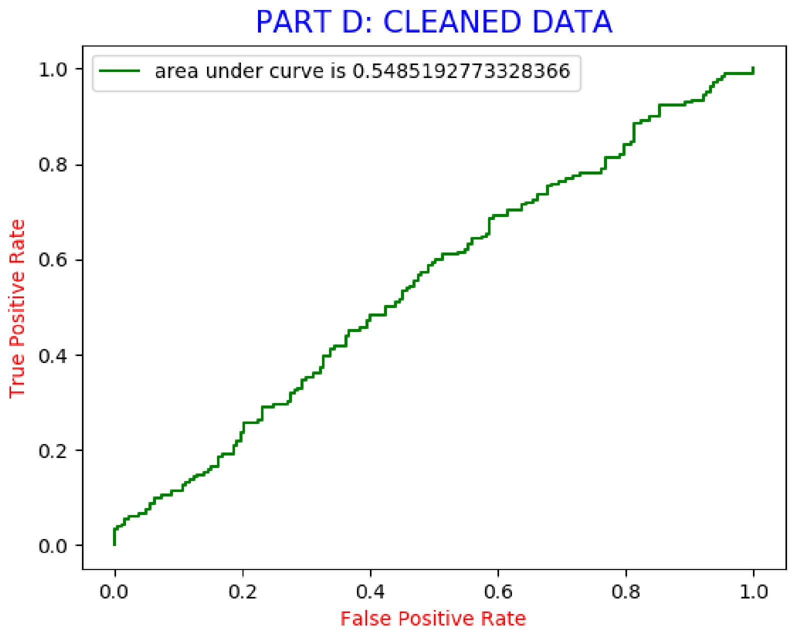
- **Observation :** We can observe that when use use all the vectorized data form tf-idf we get a low accuracy, but when we use vector of data which have a $min_d f$ value above than a given threshold we can observe significant improvement in accuracy. This is because we are considering features whose weight to describe sentiment of tweet is high.
- As we start choosing select-percentile from 5, 10 to 20 % we can can see improvement is decreasing with more the selectPercentile is, this is because we are again moving towards the original tf-idf vectors as we keep increasing out percentage.

1.7 ROC : RECEIVER OPERATING CHARACTERSTIC

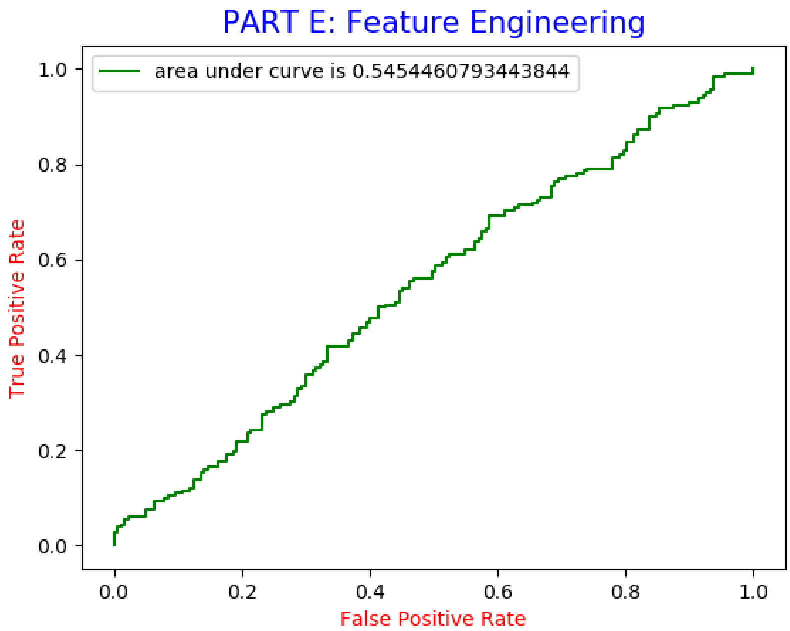
- ROC curve visualizes all possible threshold. A classifier which does very poor job of separating the classes, will have an ROC very close to diagonal.
- ROC curve summarises all of the confusion matrix that each threshold produces.
- **Original Data**



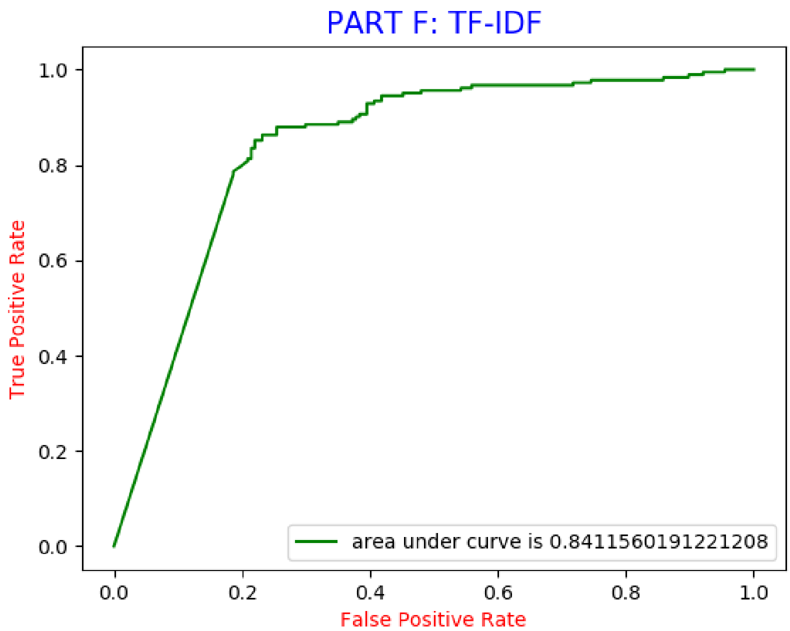
- The above curve is very close to diagonal as well as the AUC is also very low .Hence the classifier is not a very good classifier.
- **Cleaned Data**



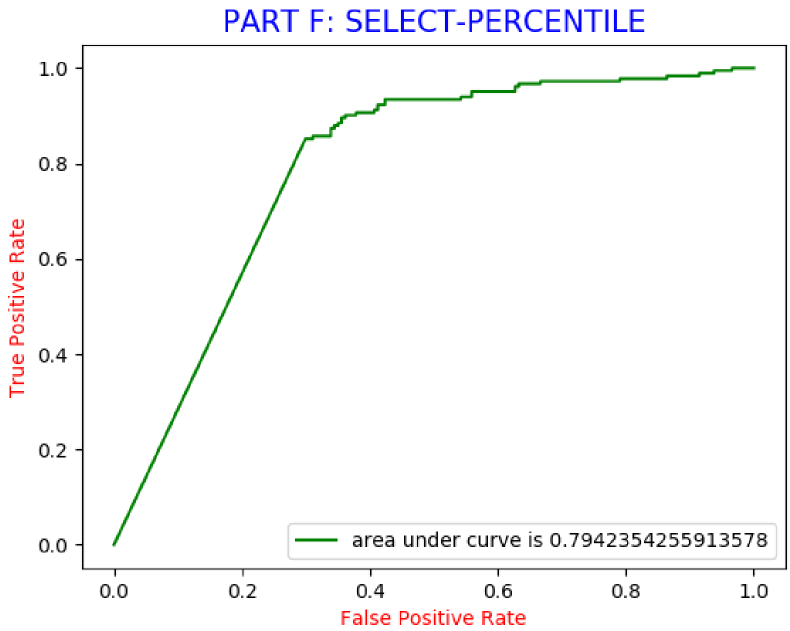
- Although the AUC here is better than pervious AUC ,still analysing the curve we can say the classifier does not do a good job for separating the classes.
- **Feature Engineering**



- ROC of Feature Engineering also depecits that it's not a very good classifier of data.
- AUC also confirms that our observation by ROC is correct.
- **TF-IDF**



- Here the ROC tell us that this is a good classifier. For this ROC we took $min_df = 7e^{-4}$. Since we are considering the features with a certain threshold set by min_df the classifier is separating the classes better.
- AUC is also high than all other cases, hence weighing on our argument above.
- **TF-IDF and SelectPercentile**



- This part after using TF-IDF with min_df =$7e^{-4}$ we use selectPercentile = 20 % to get our features.
- The ROC curve tells us that this is a good classifier for a given threshold.
- AUC also confirms our observation.
- Although above TF-IFD gave a better result still this is a good model for separating the classes.

2 Fashion MNIST Article Classification: SVM

2.1 Binary Classification

Class 4 and Class 5 : Classification

2.1.1 Linear kernel : CVXOPT

- We formulate our Dual objective in terms of CVXOPT solver for α.
- max α : 0.25238892773911575
- min α : 2.1183016408705166e-13
- Threshold α = 1e-10 for Support Vector
- Number of Support Vectors: 80
- b = -0.49688298421661503
- **Accuracy**

Train Data	Test data	Val data
100.0 %	99.8 %	99.6 %

2.1.2 Gaussian kernel : CVXOPT

- We formulate our Dual objective in terms of CVXOPT solver for α.
- max α : 0.9999999936128763
- min α : 2.7568175816981765e-09
- Threshold α = 1e-5 for Support Vector
- Number of Support Vectors: 1008
- b = 0.1504718237894755
- **Accuracy**

Train Data	Test data	Val data
100.0 %	98.8 %	99.6 %

- C =1.0 and Gamma = 0.05
- As compared to linear kernel test accuracy is 1% less. Rest all have same accuracy.

2.1.3 Scikit SVM

- C =1.0 and Gamma = 0.05
- Using SKlearn : Linear SVM

Train Data	Test data	Val data
100.0 %	99.8 %	99.6 %

- Using SKlearn : Gaussian SVM

Train Data	Test data	Val data
100.0 %	99.6 %	99.6 %

- Comparison of Linear VS Gaussian Kernel Using CVXOPT

Kernel	b : intercept	No. of Support Vectors	Time
Linear	-0.49688298421661503	80	105 sec
Gaussian	0.1504718237894755	1008	81 sec
Scikit SVC Linear	-0.49688327	71	5 sec
Scikit SVC RBF	-0.32537323	981	5 sec

- **Observation**

  As we move form Linear to Gaussian kernel Number of support vectors increase in both our implementation as well as Scikit learn. Scikit learn runtime is faster than ours. As we analyse the intercept term we can see similarity in linear case but difference in Gaussian kernel.

2.2 Multi-Class Classification

2.2.1 SVM : Binary Classifier to Multiple Classifier

- Our data-set contains total 10 Class 0-9 as label for each image.
- C = 1 and Gamma = 0.05
- Here we use our built Gaussian Model to build One VS One Classifier for Multiple Classification.
- We train $\binom{K}{2}$ models where K = 10.
- Each model is then used to predict Multi-class data-set.
- To predict the class label we first take the max count for each class. In case of tie, we get tiebreaker by taking the class with highest net score.

Test data	Val data
85.08 %	84.96 %

2.2.2 Multi-Class SVM : Scikit SVM library

- C = 1 and Gamma = 0.05
- We use kernel = **'rbf'** in Scikit SVM.
- We again train $\binom{K}{2}$ models where K = 10.
- SImilar to previous part to predict the class label we first take the max count for each class. In case of tie, we get tiebreaker by taking the class with highest net score.

Kernel	Test data	Val data
Scikit Gaussian	88.08 %	87.88 %
Our Gaussian	85.08 %	84.96 %

- Time taken:

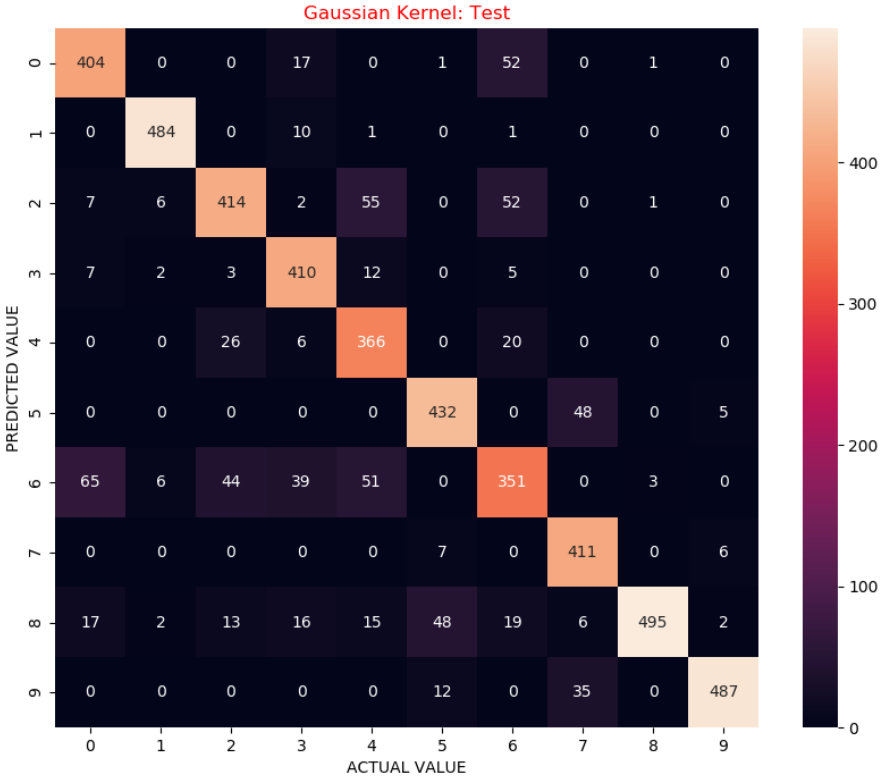
Time: Gaussian Kernel	Time: Scikit SVM kernel='rbf'
38 Min	9 Min

- **Comment ;**

  Gaussian Kernel of scikit learn gives better result. This follows from previous results obtained . The difference in intercept seems the reason for the accuracy difference.

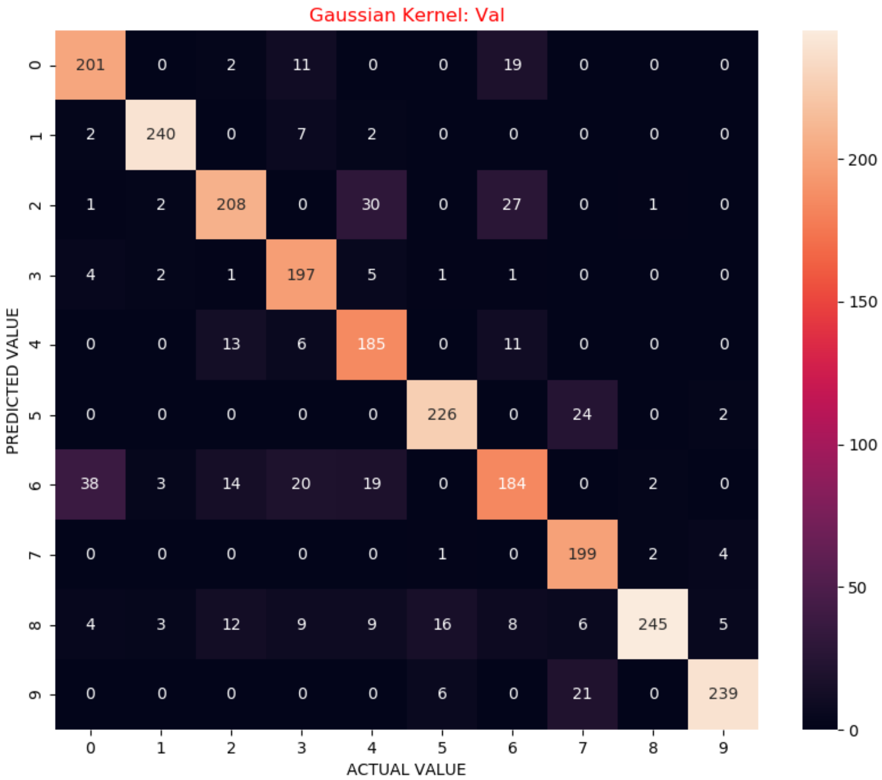
2.2.3 Confusion Matrix

- Confusion Matrix for 10 class.
- Gaussian Kernel : Test



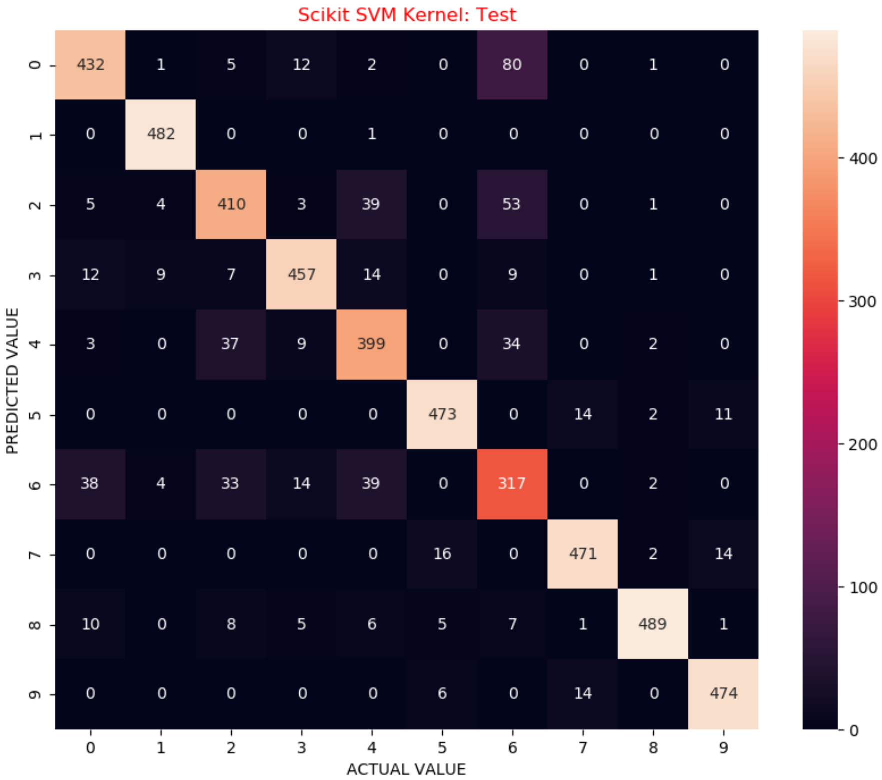
- Above is the confusion matrix of Gaussian Kernel for Test Data.
- Class 0 True positive: 404 and False positive: 71
- Class 1 True positive: 484 and False positive: 11
- Class 2 True positive: 414 and False positive: 123
- Class 3 True positive: 410 and False positive: 29
- Class 4 True positive: 366 and False positive: 52
- Class 5 True positive: 432 and False positive: 53
- Class 6 True positive: 351 and False positive: 208
- Class 7 True positive: 411 and False positive: 13
- Class 8 True positive: 495 and False positive: 138
- Class 9 True positive: 487 and False positive: 57

- Gaussian Kernel : Val



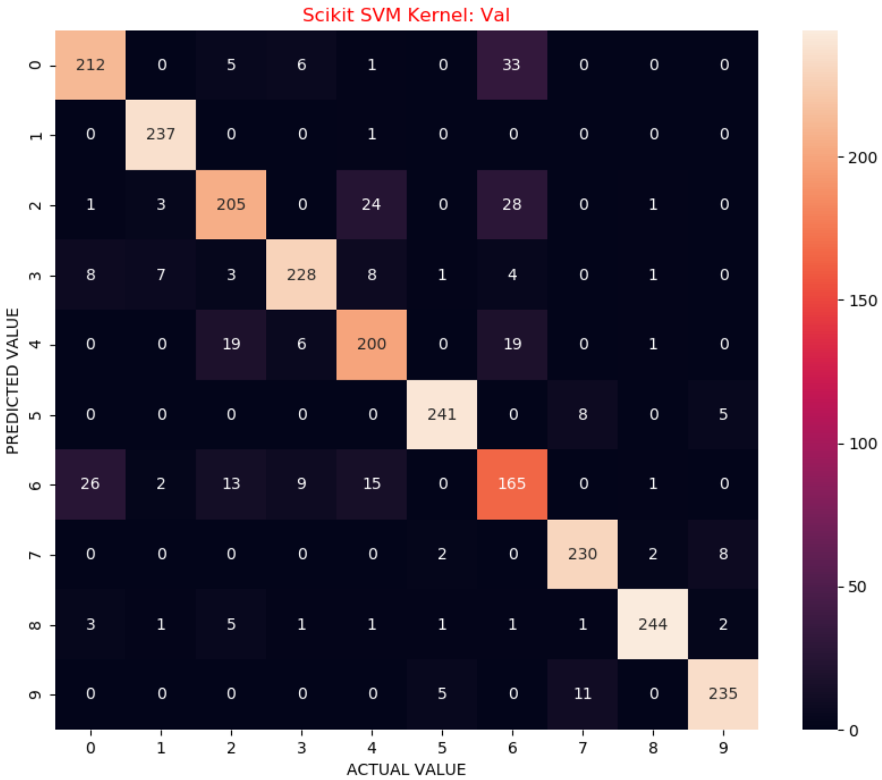
- Above is the confusion matrix of Gaussian Kernel for Val Data.
- Class 0 True positive: 201 and False positive: 32
- Class 1 True positive: 240 and False positive: 11
- Class 2 True positive: 208 and False positive: 61
- Class 3 True positive: 197 and False positive: 14
- Class 4 True positive: 185 and False positive: 30
- Class 5 True positive: 226 and False positive: 26
- Class 6 True positive: 184 and False positive: 96
- Class 7 True positive: 199 and False positive: 7
- Class 8 True positive: 245 and False positive: 72
- Class 9 True positive: 239 and False positive: 21
- **Observation** In both test and val data Class 2, 6 and 8 have high false positives.

- Scikit Kernel : Test



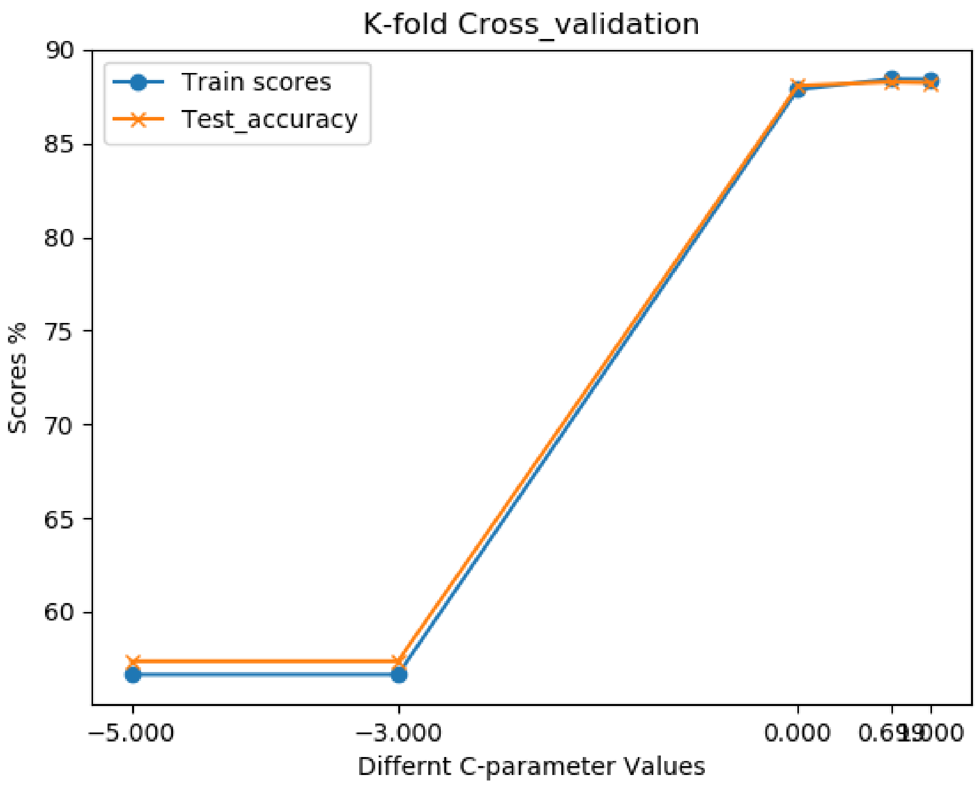
- Above is the confusion matrix of Scikit Gaussian Kernel for Test Data.
- Class 0 True positive: 432 and False positive: 106
- Class 1 True positive: 482 and False positive: 1
- Class 2 True positive: 410 and False positive: 105
- Class 3 True positive: 457 and False positive: 52
- Class 4 True positive: 399 and False positive: 85
- Class 5 True positive: 473 and False positive: 27
- Class 6 True positive: 317 and False positive: 130
- Class 7 True positive: 471 and False positive: 32
- Class 8 True positive: 489 and False positive: 43
- Class 9 True positive: 474 and False positive: 20

- Scikit Kernel : Val



- Above is the confusion matrix of Scikit Gaussian Kernel for Val Data.
- Class 0 True positive: 212 and False positive: 45
- Class 1 True positive: 237 and False positive: 1
- Class 2 True positive: 205 and False positive: 57
- Class 3 True positive: 228 and False positive: 32
- Class 4 True positive: 200 and False positive: 45
- Class 5 True positive: 241 and False positive: 13
- Class 6 True positive: 165 and False positive: 66
- Class 7 True positive: 230 and False positive: 4
- Class 8 True positive: 244 and False positive: 16
- Class 9 True positive: 235 and False positive: 16
- **Observation:** Class 6 have high false positive values as compared to others.

2.2.4 K-fold Cross Validation



- **Performance of different C's:**
- For given C's : $1e^{-5}, 1e^{-3}, 1, 5, 10$ following is the scores on train.
- 0.5664444444444444, 0.5664444444444444, 0.8787111111111111, 0.8844, 0.8842666666666666
- For each C the Test accuracy are as follows:
- 57.36, 57.36, 88.08, 88.28, 88.24
- Best performance in Train data using 5 -fold cross validation is **C = 5:** 0.8844
- Best performance on Test data is using **C = 5** 88.28 %
- For C= 5 it gives best performance on Both Test data as-well as test Data.
- Time taken to get Train Scores is 43 minutes.
- Time taken for test accuracy calculation is 20 min using parallel computation with 5 cores.