Indian Institute of Technology, Delhi
Computer Science
Name : Divyansh Mittal
Entry No: 2020CS10342

Semester 1, 2023
Assignment 2

COL774: Machine Learning

1 Text Classification

Question Question 1.a

(10 points) Implement the Naıve Bayes Multiclass classification algorithm to classify each sample into one of the given three categories. You should implement the model where for each word position in a document, we generate the word using a single (fixed across word positions) Multinoulli distribution.

- Report the accuracy over the training as well as the validation set.
- Read about word cloud. Construct a word cloud representing the most frequent words for each class.

The accuracy are as follows

- **Training:** 85.04%
- **Validation:** 67.05%

The word clouds are as follows. For these, the links were removed, else the most common word was "https"

Figure 1: Positive Word Cloud

Figure 2: Negative Word Cloud

Figure 3: Neutral Word Cloud

Question Question 1.b

(2 points)

- What is the validation set accuracy that you would obtain by randomly guessing one of the categories as the target class for each of the reviews (random prediction)?
- What accuracy would you obtain if you simply predicted each sample as positive?
- How much improvement does your algorithm give over the random/positive baseline?

- Randomly guessing gives us an accuracy of 33.46%
- Positively guessing gives us an accuracy of 43.85%
- We get an improvement of 2.005678422 times than randomly guessing. Further we get a 1.53044469783 times improvement over guessing positive only.

Question

(2 points) Read about the confusion matrix.

- Draw the confusion matrix for your results in parts (a) and (b) above (for both training and validation data).
- For each confusion matrix, which category has the highest value of the diagonal entry? What does that mean?

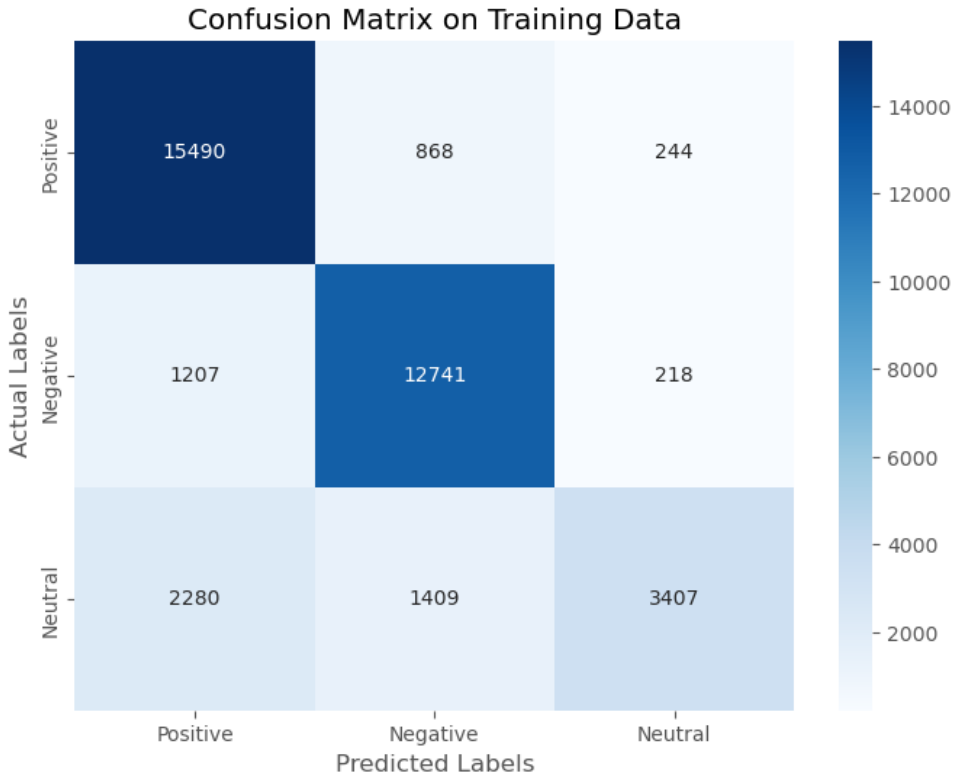


Figure 4: Confusion matrix on training

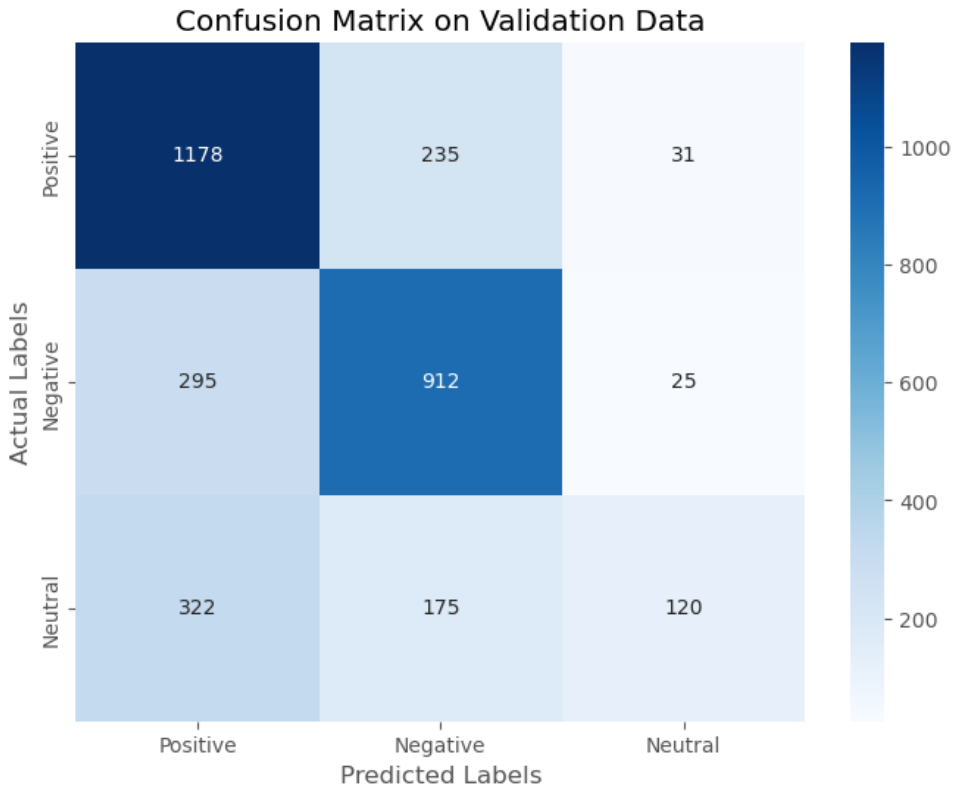


Figure 5: Confusion matrix on verifying

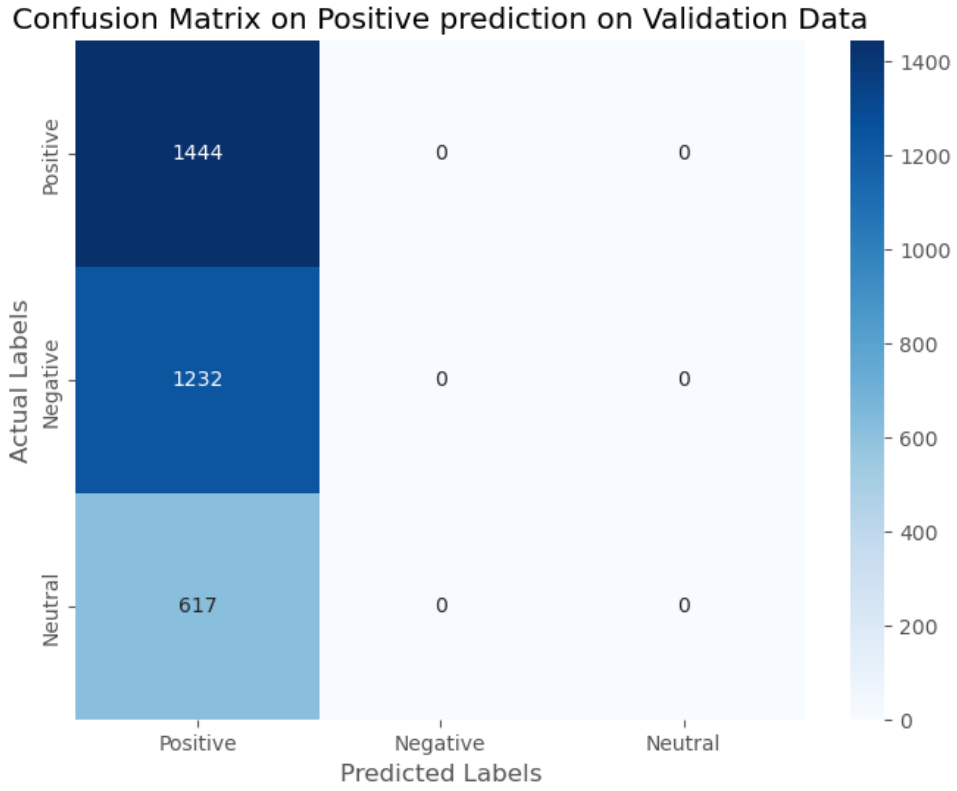


Figure 6: Confustion matrix on just predicting positive

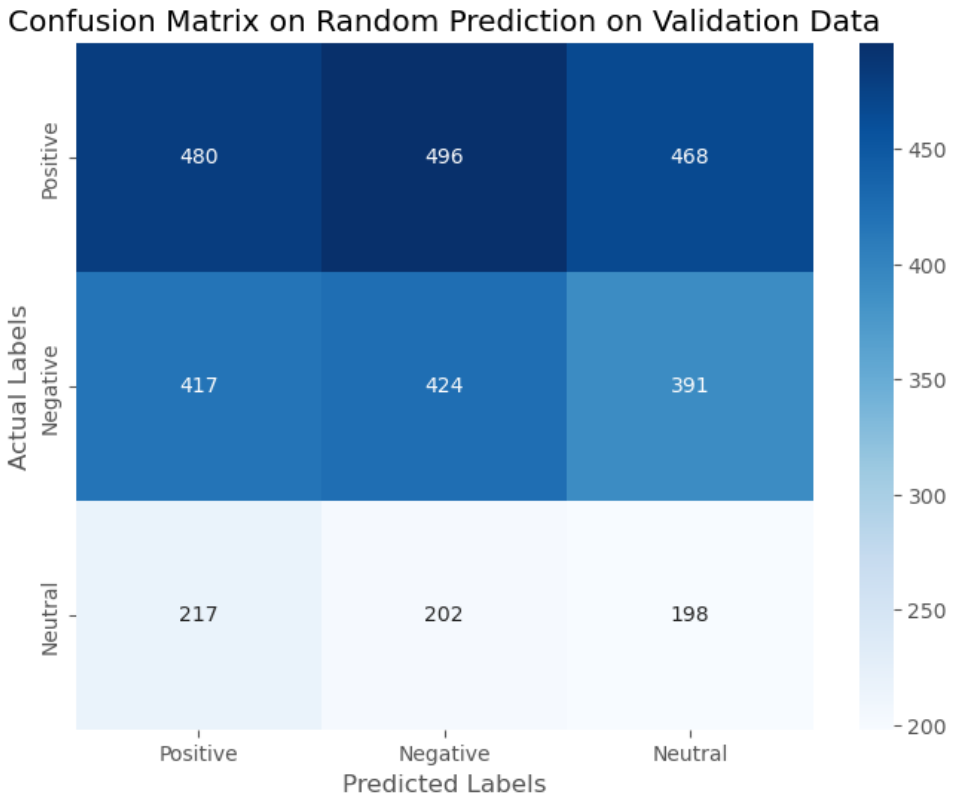


Figure 7: Confustion matrix on just predicting randomly

The positive label have the highest count. It is the one which is predicted correctly the most number of times. Even percentage wise, the positive labels are predicted correctly the most number of times.

Question

The dataset provided to you is in the raw format i.e., it has all the words appearing in the original set of articles. This includes words such as 'of', 'the', 'and' etc. (called stopwords). Presumably, these words may not be relevant for classification. In fact, their presence can sometimes hurt the performance of the classifier by introducing noise in the data. Similarly, the raw data treats different forms of the same word separately, e.g., 'eating' and 'eat' would be treated as separate words. Merging such variations into a single word is called stemming. Read about stopword removal and stemming (for text classification) online.

- Perform stemming and remove the stop-words in the training as well as the validation data.
- Construct word clouds for both classes on the transformed data.
- Learn a new model on the transformed data. Report the validation set accuracy.
- How does your accuracy change over the validation set? Comment on your observations.

Figure 8: Word cloud after stemming for positive case

Figure 9: Word cloud after stemming for negative case

Figure 10: Word cloud after stemming for neutral case

The new accuracy on validation is 67.32%

After performing stemming the accuracy is still about 67.32%, not much improvement was seen. Since these are tweets, maybe stemming here is not relevant, because in the word cloud we can see most of the words still stay the same, and they are mostly nouns, talking about everyday things like grocery or coronavirus.

Question

(6 points) Feature engineering is an essential component of Machine Learning. It refers to the process of manipulating existing features/constructing new features in order to help improve the overall accuracy of the prediction task. For example, instead of using each word as a feature, you may treat bi-grams (two consecutive words) as a feature.

- You can read here about Bi-grams. Use word-based bi-grams to construct new features. You should construct these features after doing the pre-processing described in part d(i) above. Further, these should be added as additional features, on top of existing unigram (single word) based features. Learn your model again, and report validation set accuracy.
- Come up with at least one additional set of features to further enhance your model. Learn the model after doing pre-processing as described in part d(i) above, and report the validation set accuracy.
- For both your variations tried above, compare with the validation set accuracy that you obtained in parts (a) and parts (d). Do the additional set of features constructed in each case help you improve the overall accuracy? Comment on your observations

Bigrams hurt our accuracy and it dropped to 65.35%. By introducing bigrams as a feature, we are just overfiting and introducing noise into our model, which just hurts our accuracy.

For an aditional feature, I look at giving '#' extra weight than usual. Since these are tweets, we have hastags to tag topics, and thus these are words that sum up in just a few terms the entire tweet. Thus words starting with hash, are given extra weightage by assuming these as occouring multiple times. Eg if the weighing factor is 3, for single '#COVID' will be considered to be like 3 times COVID came in the tweet.

Here I double the weight to words with a hashtag and get an accuracy of 68.05%, which is about 1% more than the base model.

Question

(6 points) In Domain Adaptation, given a model trained on data from the source domain, we apply it on data from the target domain. The idea is to exploit the similarity between source and target domains, and leverage the parameters learned on the target domain, making up for lack of sufficient training data in the target domain. In our setting, we will assume the source domain is coronavirus tweets, and the target domain is represented by another dataset of general purpose tweets. You are given subsets of different sizes of the training data (in the target domain), varying from 1%, 2%, 5%, 10%, 25%, 50%, 100% of the total training set.

- Learn a Naıve bayes model, on entire source training data, combined with the partial target training data for each of the splits above. Your vocabulary should be constructed on the combined dataset. You should use the model in part (d) above, i.e., after removing stopwords, and performing stemming but before doing any additional feature engineering. Compute (target) validation set accuracy for each split.
- Do the same thing as in the part above, but without adding any data from source domain. This represents learning from scratch on the target data. Compute the accuracy on target validation set for each of the splits.
- Plot on a single graph the validation set accuracy for the two algorithms described above, as

we vary the size of target training data.

- What do you observe? Explain.

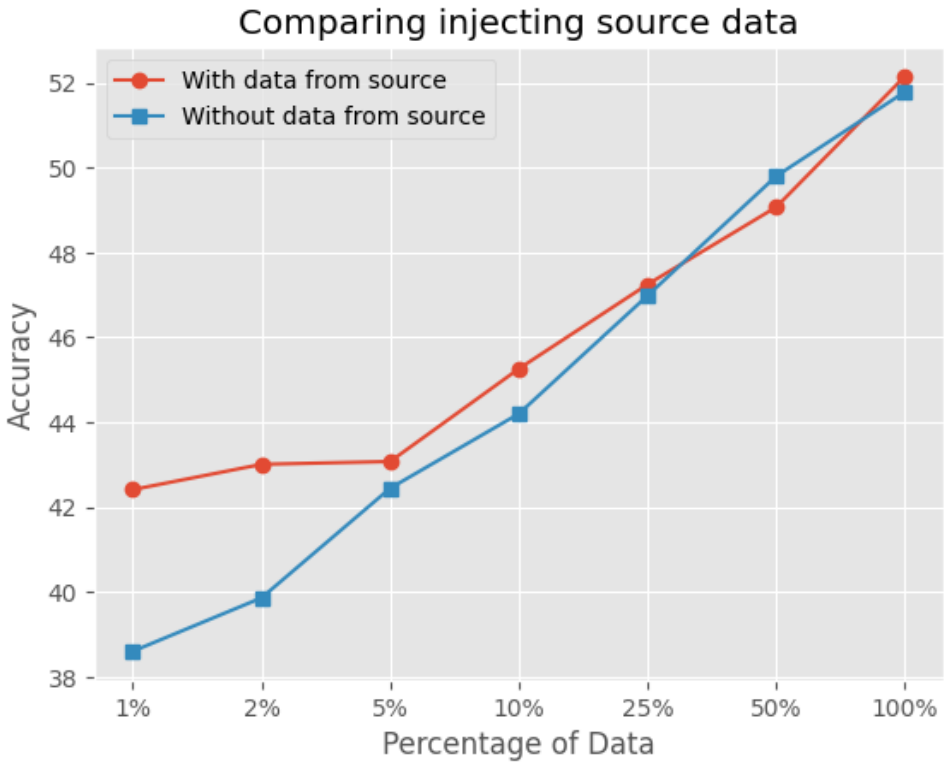


Figure 11: Accuracy with domain adaptation data

When very little source data is there, our model is still pretty good. As more and more data comes in, we converge together. Further this accuracy always keeps climbing. Although it could very well be, that now since more of the source data is incorporated, any negative effect added by the original data is relatively reduced.

2 Image Classification

2.1 Binary Classification

Question

Download and install the CVXOPT package. Express the SVM dual problem (with a linear kernel) in a form that the CVXOPT package can take. You will have to think about how to express the SVM dual objective in the form $\alpha^T P \alpha + q^T \alpha + c$ matrix where P is an mm matrix (m being the number of training examples), q is an m-sized column vector and c is a constant. For your optimization problem, remember to use the constraints on $\alpha_i' s$ in the dual. Use the SVM formulation which can handle noise and use C = 1.0 (i.e. C in the expression $\frac{1}{2} w^T w + C \cdot \sum_i \mathcal{E}_i$). You can refer this link to get a working overview of cvxopt module and it's formulation.

- How many support vectors do you get in this case? What percentage of training samples constitute the support vectors?

- Calculate the weight vector w and the intercept term b and classify each of the examples in the validation file into one of the two labels. Report the validation set accuracy. You will need to carefully think about how to represent w and b in this case.
- Reshape the support vectors corresponding to the top- 6 coefficients to get images of 16 × 16 × 3 and plot these. Similarly, reshape and plot the weight vector w.

- I get that 2326 number of $\alpha_i's$ out of 9520 data points are greater than 10^{-3}, and thus these are the supports. About 24% are a support.
- The accuracy on validation is 80.0 %

(a) Support 1

(b) Support 2

(c) Support 3

(d) Support 4

(e) Support 5

(f) Support 6

Figure 12: Visualising supports

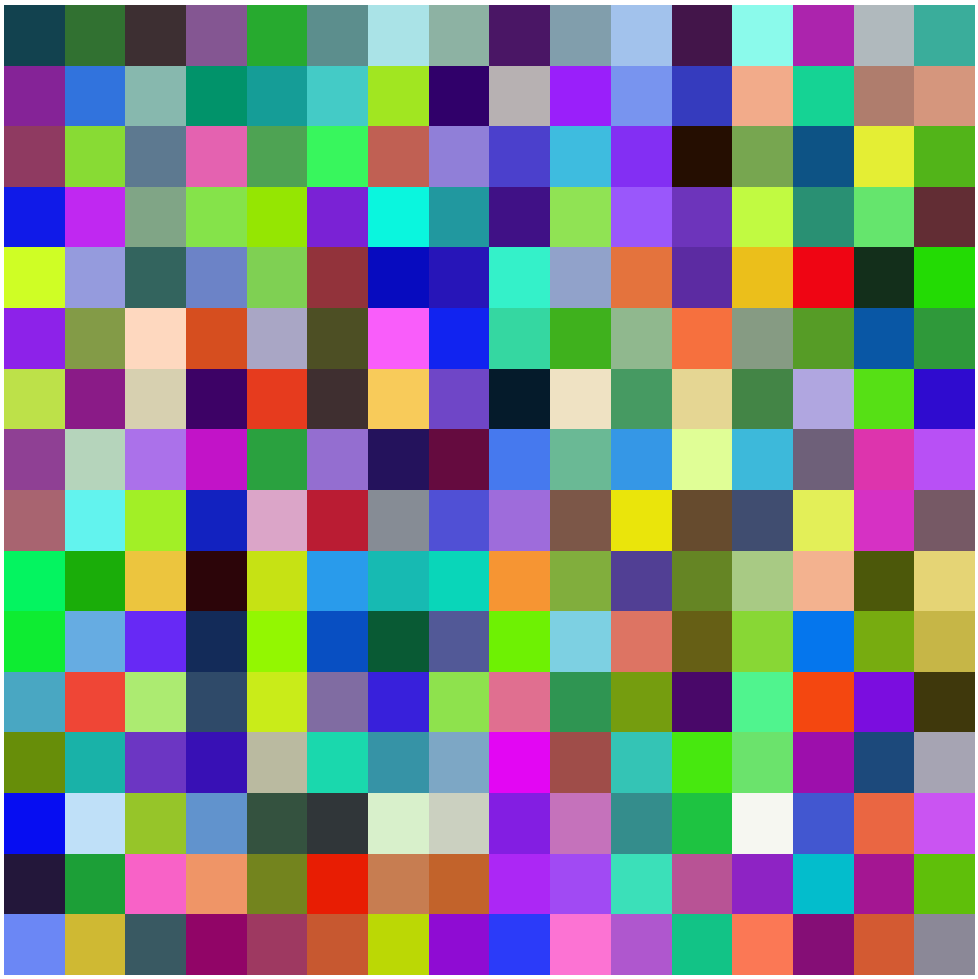


Figure 13: Weight as an image

Question

(6 points) Again use the CVXOPT package to solve the dual SVM problem using a Gaussian kernel. Think about how the P matrix will be represented. Use C = 1.0 and $\gamma = 0.001$ (i.e. γ in K(x, z) = $e^{\gamma \cdot ||x-z||^2}$ for this part.

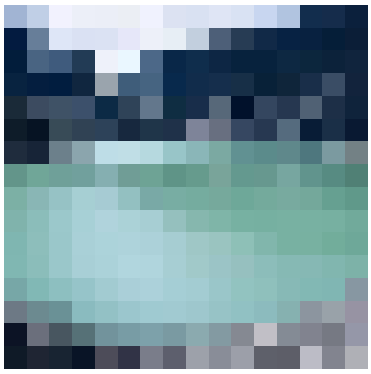
- How many support vectors do you get in this case as compared to the linear case above? How many support vectors obtained here match with the linear case above?
- Note that you may not be able to explicitly store the weight vector (w) or the intercept term (b) in this case. Use your learned model to classify the validation examples and report the validation accuracy.
- Reshape the support vectors corresponding to the top- 6 coefficients to get images of 16 × 16 × 3 and plot these.
- Compare the validation accuracy obtained here with part (a).

- I get that 2783 number of $\alpha_i's$ out of 9520 data points are greater than 10^{-3}, and thus these are the supports. About 2133 of them are the same.
- The accuracy on validation is 79.75 %
- The accuracy in both the cases is nearly the same.

(a) Support 1

(b) Support 2

(c) Support 3



(d) Support 4

(e) Support 5

(f) Support 6

Figure 14: Supports for gaussian

Question

Repeat parts -(a) (b) with the scikit-learn SVM function, which is based on the LIBSVM package.

- Compare the nSV (Number of Support Vectors) obtained here with part (a) for linear kernel and part (b) for Gaussian kernel. How many of the support vectors obtained here match with the support vectors obtained in both these cases?
- Compare weight (w), bias (b) obtained here with part (a) for linear kernel.
- Report the validation accuracy for both linear and Gaussian kernel.
- Compare the computational cost (training time) of the CVXOPT with the sklearn implementation in both the linear and Gaussian case

- For linear, I get 2325 number of support vectors with scikit. Further all of them were also classified as support by our algorithm. For the gaussian case, I get 2780 supports, and all of these were also support for our case as well.
- The maximum absolute difference between any single coeffecient of our weights and sklearns weights is 0.0024000225132783104. The mean absolute coefficient in the weights is 0.33393146734863083. Thus these are very close.

  The ski based bias is 0.6756906666142197, we compute 0.6762966689106026 the difference is 0.0006060022963828571, and thus these two are extremely close.
- The validation accuracy matched with our solution ans was also 80% and 79.75% respectively.

- There was a huge difference in the time our implementation takes vs scikit's implementation. For linear, they took 9.0s, aund we took 32.7 seconds. Further for Gaussian, they take only 5.6s, whereas our code requires 26s.

2.2 Multiclass Classification

Question

(4 points) In class, we described the SVM formulation for a binary classification problem. In order to extend this to the multi-class setting, we train a model on each pair of classes to get $\binom{k}{2}$ classifiers, k being the number of classes (here, k = 6). During prediction time, we output the class which has the maximum number of votes from all the $\binom{k}{2}$ classifiers. You can read more about one-vs-one classifier setting at the following link. Using your CVXOPT solver from the previous section, implement one-vs-one multi-class SVM. Use a Gaussian Kernel with C = 1.0 and $\gamma = 0.001$.

- Classify the validation examples and report validation set accuracy. In the case of ties, choose the label with the highest score.

- Validation accuracy is 55.58%

Question

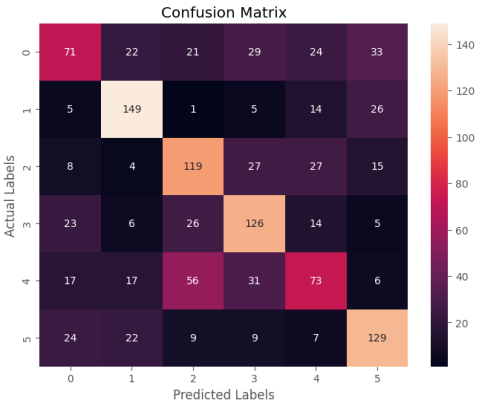
(3 points) Now train a multi-class SVM on this dataset using the scikit-learn SVM function, which is based on the LIBSVM package. Repeat part (a) using a Gaussian kernel with $\gamma = 0.001$. Use C = 1.0 as earlier.

- Classify the validation examples and report validation set accuracy.
- How do the validation set accuracy and the training time obtained here compare with part (a) above?

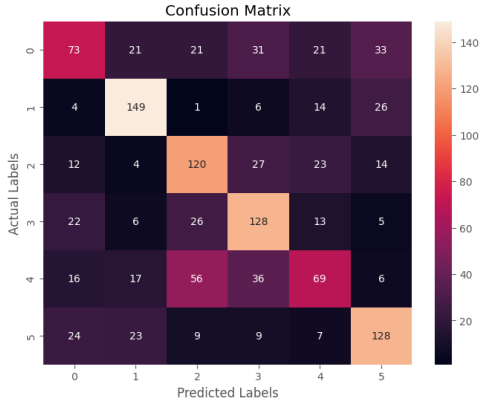
- With scikit as well, validation accuracy is 55.58%
- The accuracy in both the cases comes out to be the same. Further I ran this parallely on a server with 100+ cores, and thus my implementation ran in 2min 29.2s, the built-in library took only 49.9s

Question

(4 points) Draw the confusion matrix for both of the above parts (2(a) and 2(b)). What do you observe? Which classes are miss-classified into which ones most often? Visualize (and report) 12 examples of miss-classified objects. Do the results make sense? Comment.



(a) Confusion matrix with Scikit



(b) Confusion matrix with our implementation

There is slight difference in both these classifications. Further class 1 and class 4 are the ones mis classified the most. For class 2 and class 5, we mostly get very few mis-classifications.

These makes sense, as class 1 tries to classify forests, and thus these type of images probably have a lot of green and brown colors, that probably other images don't have and thus it might be easy to classify these. Further class 0 are maybe buildings, thus they have a lot of sky, and sourrounding areas, thus probably hard to classify. Further 2,3,4 are very similar and thus, these get mis-classified among themselves a lot.

(a) Misclassied label 2 as 0

(b) Misclassied label 1 as 5

(c) Misclassied label 2 as 0

(d) Misclassied label 2 as 4

(e) Misclassied label 5 as 0

(f) Misclassied label 0 as 5

(g) Misclassied label 0 as 3

(h) Misclassied label 0 as 4

(i) Misclassied label 4 as 1

(j) Misclassied label 4 as 3

(k) Misclassied label 4 as 3

(l) Misclassied label 5 as 0

Figure 16: Misclassifications

Question

(3 points) Validation set is typically used to estimate the best value of the model hyperparameters (e.g., C in our problem with the Gaussian kernel) by randomly selecting a small subset of the training data as the validation set and then training on the modified training data (original training data minus the validation set) and making predictions on the validation set. For a detailed introduction, you can refer to this video. You can check the correctness of your intuition by trying this test. K-fold cross-validation is another such technique in this regard that we use in practice. In this technique, we divide our training data into K-folds or parts and then treat each part as our validation set once and train on the remaining K-1 parts. You can read more about cross-validation here 1 (see Section 1) for more details. This process is repeated for a range of model hyper-parameter values and the parameters which give best K-fold cross-validation accuracy are reported as the best hyper-

parameters. We will use scikit-learn SVM function for this part. For this problem, we will do a 5-fold cross-validation to estimate the best value of the C parameter for the Gaussian kernel case. Validation data should not be touched.

- Fix γ as 0.001 and vary the value of C in the set $\{10^{-5}, 10^{-3}, 1, 5, 10\}$ and compute the 5 -fold cross-validation accuracy and the validation accuracy for each value of C.
- Now, plot both the 5-fold cross-validation accuracy as well as the validation set accuracy on a graph as you vary the value of C on x-axis (you may use log scale on x-axis). What do you observe? Which value of C gives the best 5 -fold cross-validation accuracy? Does this value of the C also give the best validation set accuracy? Comment on your observations.

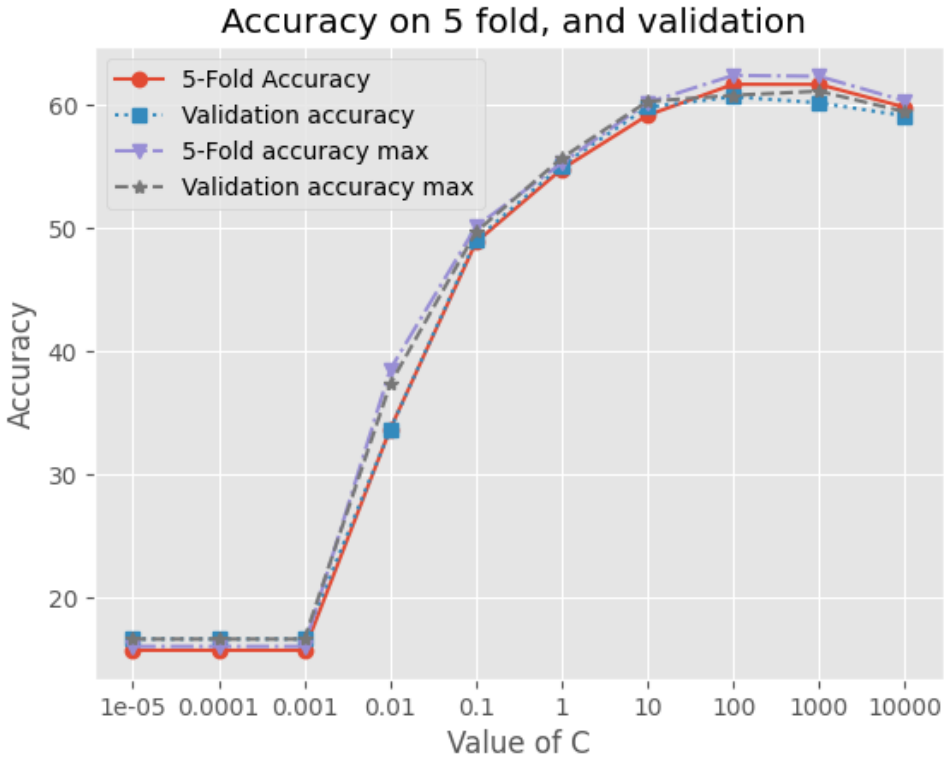


Figure 17: 5 Fold cross validation accuracy and validation set accuracy for multiple values of C

The best set accuracy is obtained for C = 100, after which it starts to dip. The cross-fold accuracy and the validation accuracy follow very closely, as the cross-fold validation accuracy peaks, we find that the accuracy on validation set also goes to a maximum. By heavily penalising outliers, we get a much better accuracy. When it is too harsh or too little we get an over-fit or underfit.