COL774: Assignment 2 Report

Aniruddha Chatterjee

2021MT10896

Q1. Text Classification

(a) In the first part, we implement Multinoulli Naïve Bayes utilizing the bag-of-words model.

Without any pre-processing (i.e., only splitting the tweet into space-separated words):

We obtain accuracy:
Train = 85.04648214663004% Validation = 67.05132098390525%

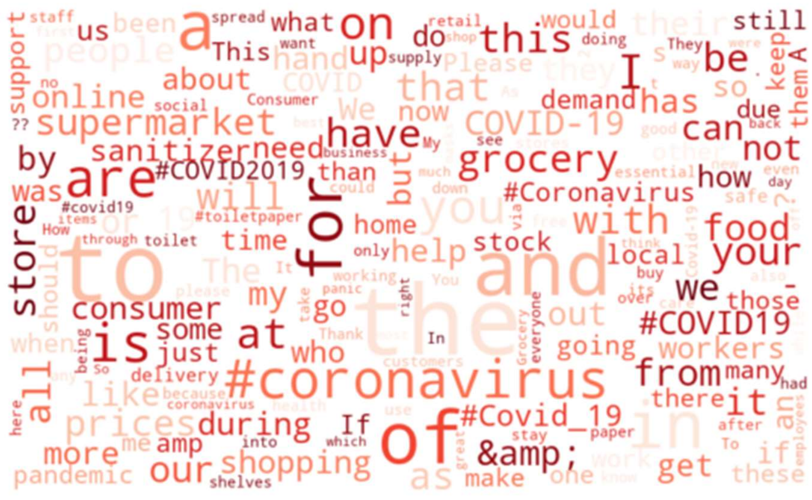
Also. only the following steps having been performed:

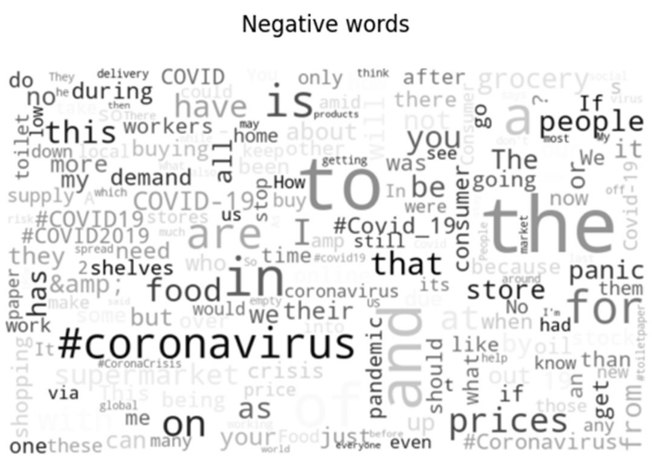
- Removing punctuation (including special characters, #, @ from tags/handles)
- Converting all words to lowercase

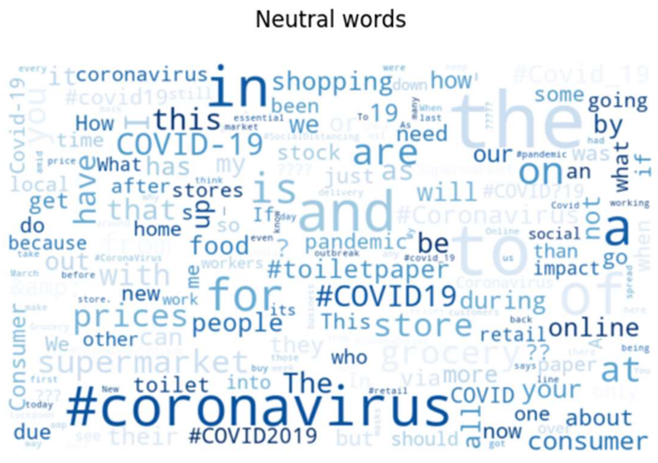
We obtain accuracy:
Train = 80.74160152123389% Validation = 68.99483753416338%

We obtain the following wordclouds based on the training data:

Positive words







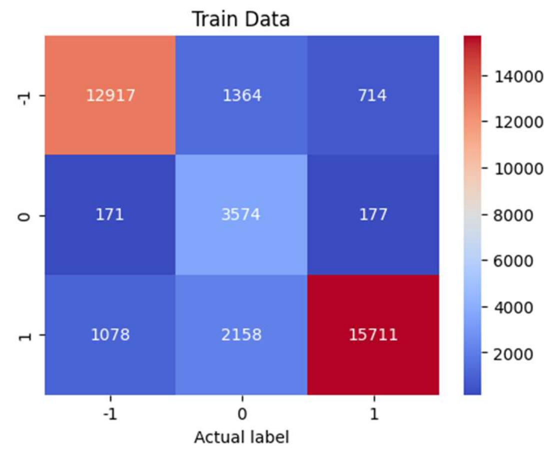
(b) Accuracy:

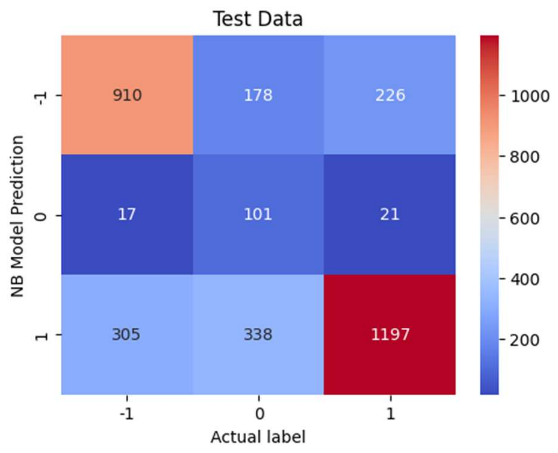
	Random Guess	Positive-only
Train	33.3588%	43.8464%
Validation	33.1005%	43.8506%

There is roughly an increase of 33% over random and 23% over positive guesses on validation data in terms of accuracy of our initial model.

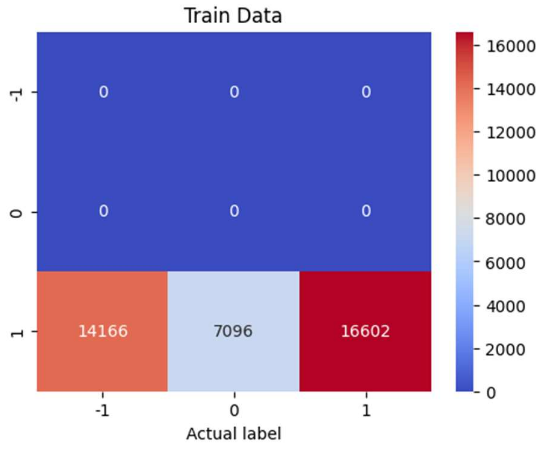
(c) We obtain Confusion Matrices as follows: -

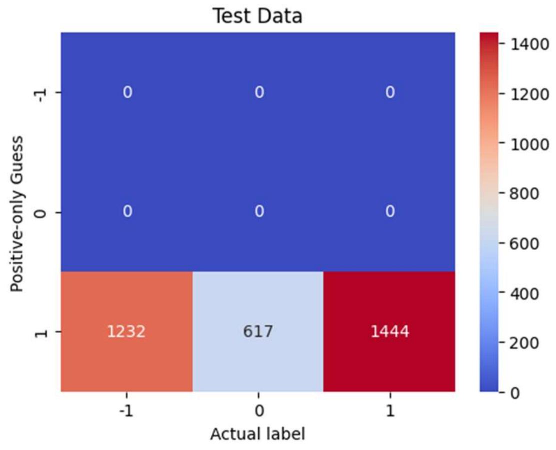
- Random Guess



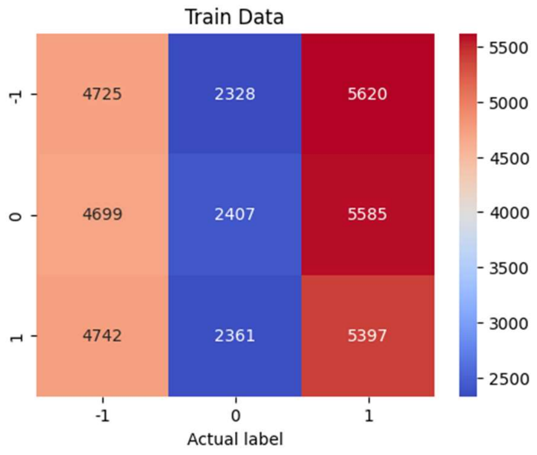


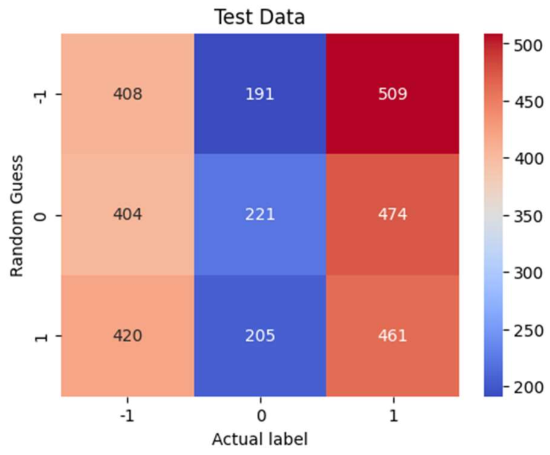
- Positive-Only Guess





- NB Model Predictions





- In random guessing, none of the categories is dominant as expected (all actual labels are equally split in columns).
- In positive guessing, all the rows except row=1 are naturally 0, since no guess except 1 is predicted.
- In our model predictions, we see that diagonal entries are dominant (i.e. correctly predicted) except for Neutral labels in test data (may be due to low occurrence/prior).

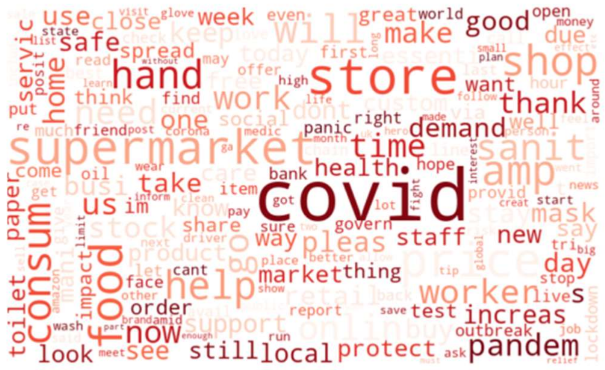
All 6 Confusion matrices have the largest diagonal entry at (1,1), which means Positive is the majority category in correct predictions.

(d) We perform further pre-processing by employing stopwords removal and stemming (using nltk library) individual words.

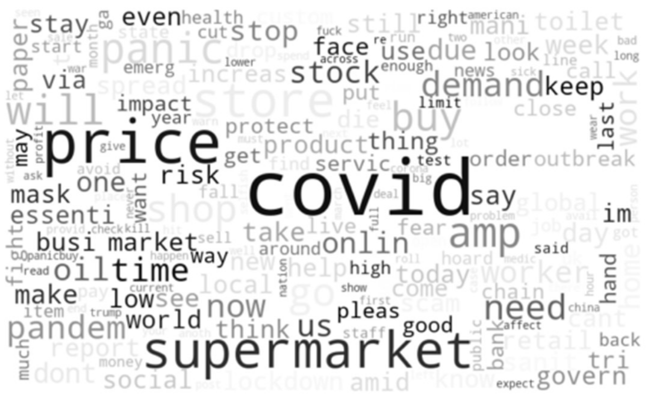
This marginally improves our accuracy on validation over our initial model:
Train = 79.66670188041411% Validation = 67.99271181293653%

We obtain the following wordclouds on the set of word stems generated from training data:

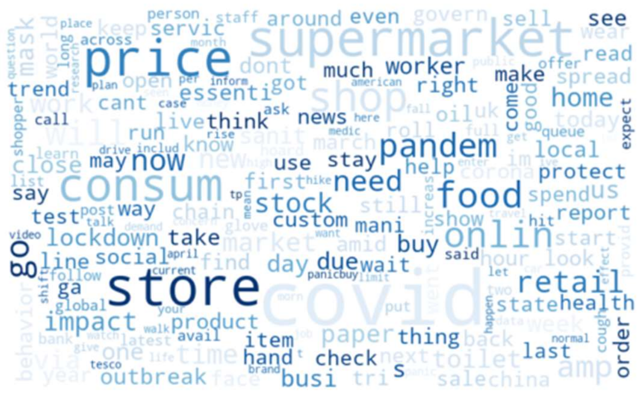
Positive words



Negative words



Neutral words



(e) Using bigrams, we generate a new "set of words" in the 'CoronaTweetStemBgm' column. We add this alongside our set of word stems as generated in (d) to fit our model.

- New accuracy:

Train = 93.5638% Validation = 66.2921%

Inferring these values, Bigrams coupled with stemmed vocabulary seems to overfit the training data, as accuracy goes from around 80% to 94%. While accuracy on validation set stays the same, if not decreased slightly.

Nevertheless, I tried using Trigrams as a first measure, which did not help with validation accuracy and instead led to further overfitting (may be expected).

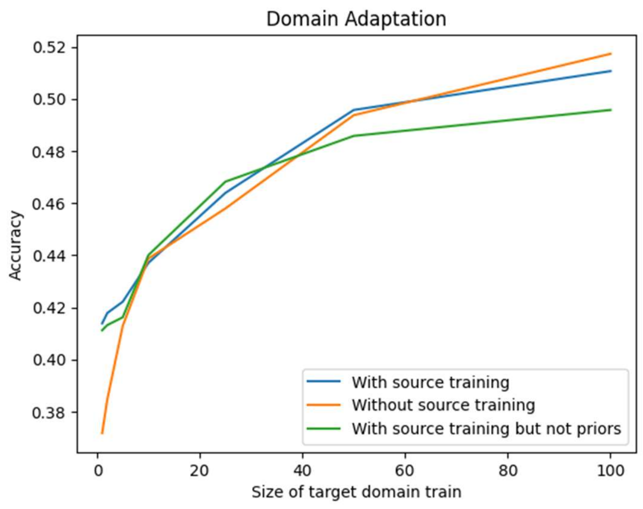
- New accuracy:
  Train = 93.5638%   Validation = 66.2921%

I also tried to make a new feature using hashtags and handles, but that too doesn't seem to perform very well. Now, the model was trained on 'CoronaTweetHash' where these special words are repeated twice.

- New accuracy:
  Train = 93.5796%   Validation = 65.9580%

Maybe more insight is needed on this data to come up with a better feature.

(f) Using domain adaptation along with the partial target datasets for training our model, we get the following trend:



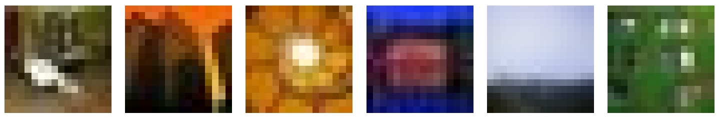
- We see a significant increase with source training at smaller sizes (this is because the model hasn't learned much on the smaller sized target training data).
- Only for size=100%, the model without source training performs better.

Q2. Image Classification

(a) We use the matrices as follows in the CVXOPT Quadratic Program solver for the SVM dual problem (with tolerance=1e-6):
  - P = matrix((y@y.T)*Ker(X,X))
  - q = matrix(-np.ones(m))
  - A = matrix(y.T)
  - b = matrix(0)
  - G = matrix(np.vstack([-np.eye(m), np.eye(m)]))
  - h = matrix(np.hstack([np.zeros(m), self.C*np.ones(m)]))

For the SVM problem on labels 0v1, using Linear kernel, we get 1384 support vectors out of 4760 samples (29.08% sv). With accuracy:
Train = 91.0714% Validation = 85.25%

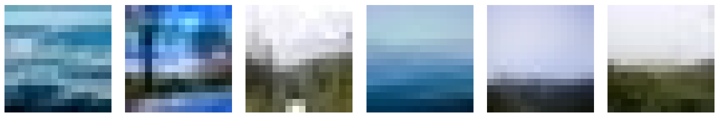
The top 6 SVs are (with indices [412, 171, 450, 1381, 719, 664]):



Descaling the weights vector the same way we did for

(b) Using the Gaussian kernel on labels 0v1, we get 2147 support vectors out of 4760 samples (45.11% sv). With accuracy:
Train = 88.6134% Validation = 84.5%

The top 6 SVs are (with indices [1079, 1269, 1083, 1474, 1080, 1822]):



From a visual inspection, the 5th SV seems to match with the linear case. While the linear kernel looks at direct boundaries, the gaussian kernel seems to allow finer boundaries.

The validation accuracy of both kernels is similar, and there isn't much improvement from using the Gaussian kernel on this pair of labels.

(c) No. of SVs in Scikit-learn SVM(lin) = 1379.
No. of SVs in Scikit-learn SVM(rbf) = 1918
All the SVs from both, the linear and the gaussian, kernel Scikit SVM are common to our

implementation (i.e. it is a superset of SVs, having a few more in both cases).

ii. Comparing w and b for the Linear kernel SVMs:

- Norm diff = || w – w_skl || = 0.014788937817747087
- Implemented b: 2.483130123329591
  Sklearn b: 2.4334948

iii. Accuracy for the Scikit-learn SVMs:

Train(lin) accuracy = 90.88%
Test(lin) accuracy = 85%
Train(rbf) accuracy = 88.6134
Test(rbf) accuracy = 84.5%

iv. Time taken for training each model:

Time taken using linear kernel(in a) = 195.8599410057068 s
Time taken using Gaussian kernel(in b) = 194.71329045295715 s
Time taken by linear Scikit-learn SVM = 6.036437511444092 s
Time taken by gaussian Scikit-learn SVM = 4.943080186843872 s

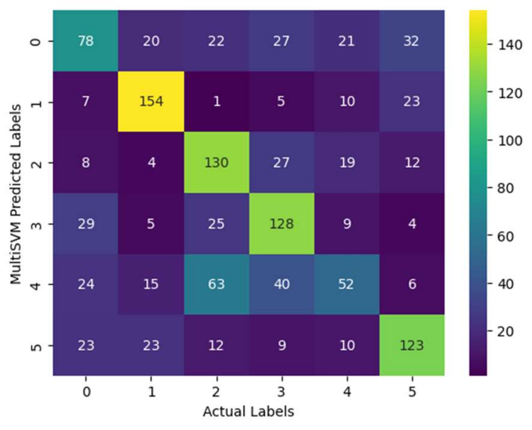
Multi-Class Image Classification

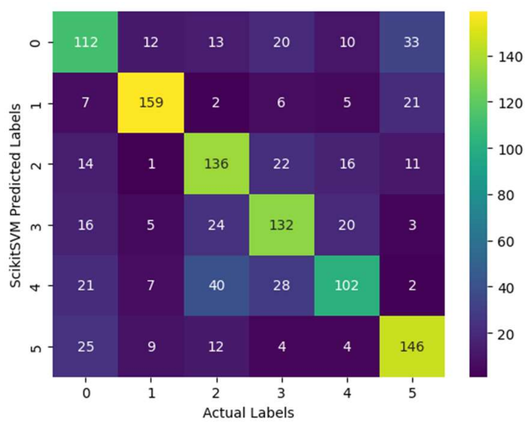
(a) We generate a one-vs-one multiclass model (with total 15 classifiers):
Time taken by Multiclass Scikit-learn SVM(lin) = 1060.900068283081 s
Obtained (multiclass) validation accuracy = 55.4166%

(b) Using the Scikit-learn implementation, with the same settings as our model('ovo' decision function, same gamma and C),
Time taken by Multiclass Scikit-learn SVM(rbf) = 69.44930624961853 s
Obtained (multiclass) validation accuracy = 65.5833%

Similar to the case with binary Gaussian kernel classifier, training time is much lower. Whereas, there is a significant improvement in accuracy for the Scikit model.

(c) Confusion matrices:





- We see a lot of 4 is classified into: 2,3 by SKL and 0,2 by MultiSVM (4 and 2 might be similar labels)

- A lot of 5 is predicted as 0 by both models (similar labels in terms of training vectors).

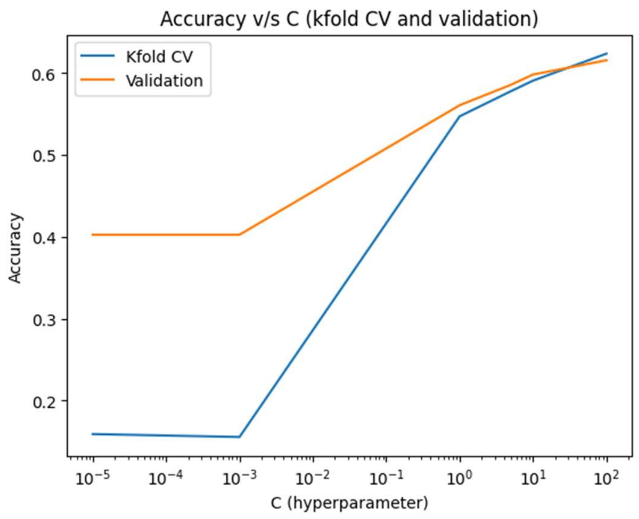
Here are 12 examples of misclassified objects,

The results seem to make sense as both 0 and 5 have images of man-made structures, while there are many instances 2(mountain/rivers?) and 4(beach/coast?) being both natural environments.

(d) 5-fold CV accuracy for the values of C = [1e-5,1e-2,1,5,10] + [100] is:
[0.158753, 0.1551820, 0.547268, 0.578011, 0.590896] + [0.623879]

We can observe the increasing trend with the value of C here.

Accuracy plot:



Accuracy for C=1: 0.5472689075630253 0.5608333333333333
Best accuracy (for C=10): 0.5908963585434174 0.5983333333333334

We see that out of C = [1e-5,1e-2,1,5,10], we obtain the best accuracy for C=10 in case of both Kfold cross-validation and train-test validation.
Upon going higher, it still seems to increase for C=100 in both metrics.