COL774
Assignment 2

Vaibhav Seth

October , 2023

Contents

1 Text Classification

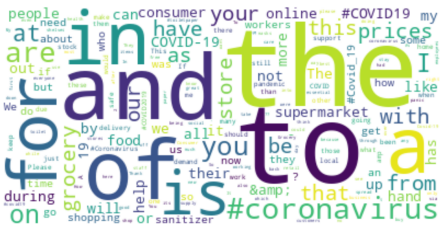
1.1 Naïve Bayes

Using a multinomial classifier model, the following observations were made:

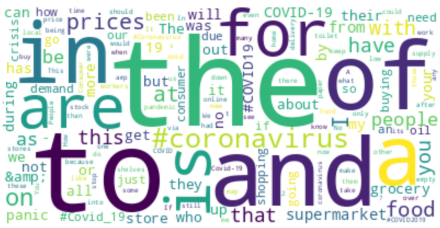
1.1.1 Accuracies

1. Training accuracy is 85.04648214663004%
2. Test accuracy is 66.80838141512299%

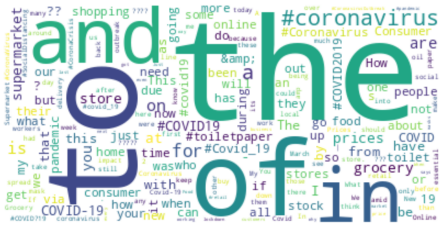
1.1.2 Word Clouds



(a) Word Cloud for positive sentiment words



(b) Word Cloud for negative sentiment words



(c) Word Cloud for neutral sentiment words

Figure 1: Word Clouds for part A (without pre-processing)

1.2 Random and Mode Models

1.2.1 Random Model

The probability of predicting each class is $\frac{1}{3}$ as there are 3 classes and each example is independent. Therefore, the expected accuracy is $\frac{m \times \frac{1}{3}}{m}$ which is 33.33%. The accuracy obtained is 32.6450045551116913% which is consistent with our calculations.

1.2.2 Positive Prediction Model

The accuracy obtained in this case is 43.85059216519891%. This is primarily because the percentage of positive tweets = 43.84556473974704%

1.3 Confusion Matrix

The confusion matrix for the given task :

	Negative	Neutral	Positive
Negative	931	41	260
Neutral	173	170	274
Positive	237	69	1138

Figure 2: Confusion matrix over val set for classification task using Naive Bayes

The confusion matrix for the given task :

	Negative	Neutral	Positive
Negative	12917	171	1078
Neutral	1364	3574	2158
Positive	714	177	15711

Figure 3: Confusion matrix over train set for classification task using Naive Bayes

The confusion matrix for random prediction :

	Negative	Neutral	Positive
Negative	389	425	418
Neutral	190	220	207
Positive	452	514	478

Figure 4: Confusion matrix over val set for classification task by randomly predicting

The confusion matrix for random prediction :

	Negative	Neutral	Positive
Negative	4746	4648	4772
Neutral	2407	2353	2336
Positive	5550	5502	5550

Figure 5: Confusion matrix over train set for classification task by randomly predicting

The confusion matrix for positive prediction :

	Negative	Neutral	Positive
Negative	0	0	1232
Neutral	0	0	617
Positive	0	0	1444

Figure 6: Confusion matrix over val set for classification task by predicting positive

The confusion matrix for positive prediction :

	Negative	Neutral	Positive
Negative	0	0	1232
Neutral	0	0	617
Positive	0	0	1444

Figure 7: Confusion matrix over train set for classification task by predicting positive

We see that in each case the Positive-Positive diagonal entry has the highest value. Even for positive prediction we had a 43% accuracy. This shows that the data is slightly skewed towards the positive side.

1.4 Cleaning and Stemming

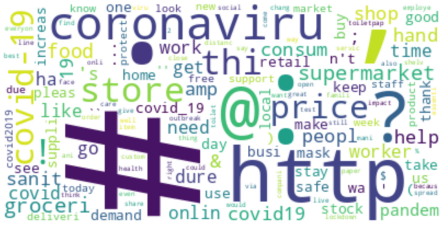
1.4.1 Accuracies

The results on cleaning and stemming are:

1. Training accuracy is 81.53655186984999%
2. Test accuracy is 69.23777710294564%

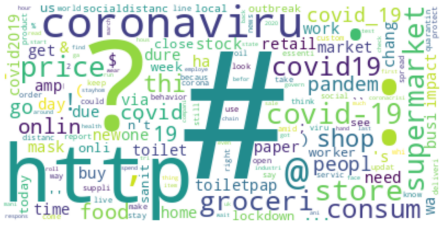
We observe that the training accuracy decreases and the testing accuracy improves over Naive bayes without any pre-processing. The result is that without preprocessing we give a very high weightage to stopwords which results in distributions being skewed towards stop-words and in a way we lose out on the sentiment information of the token sequence. As a result of removing stopwords, we get a better distribution and as a result test accuracy increases. I do no think stemming contibutes that much to the perofrmance because on using only stopword removal, a better accuracy was achieved.

1.4.2 Word Clouds



(a) Word Cloud for positive sentiment words

(b) Word Cloud for negative sentiment words



(c) Word Cloud for neutral sentiment words

Figure 8: Word Clouds for part D (with pre-processing and Stemming)

1.4.3 Accuracies with Lemmatizing

1. Training accuracy is 82.90460595816607%
2. Test accuracy is 70.23990282417248%

We observe that the training accuracy decreases and the testing accuracy very slightly improves.

1.4.4 Word Clouds

(a) Word Cloud for positive sentiment words

(b) Word Cloud for negative sentiment words

(c) Word Cloud for neutral sentiment words

Figure 9: Word Clouds for part D (with Cleaning and Lemmatizing)

1.5 Feature Engineering

1.5.1 Bi-Grams

1. Training Accuracy = 94.90809211916332%
2. Validation Accuracy = 68.2356513817188%

The training accuracy increases substantially, but the there is not much increase in validation when compared to pre-processing and stemming/lemmatizing steps.

1.5.2 Bi-Grams with Stemming and punctuation removal

1. Training Accuracy = 96.44253116416649%
2. Validation Accuracy = 69.66292134831461%

There is an increase in accuracy as compared to normal Bi-gram case. This is because we have carried out an extra cleaning process (removing puncs).

1.5.3 Bi-grams with Lemmatization

1. Training Accuracy = 96.98130150010564%
2. Validation Accuracy = 69.57181901002125%

The training accuracy increases slightly compared to Bi-grams with stemming, but validation accuracy slightly decreases.

1.5.4 Only punctuation and stopword removal and Unigram

1. Training Accuracy = 83.99535178533699%
2. Validation Accuracy = 71.09019131491041%

Just data cleaning performs fairly better than a lot other pre-processing steps. A possible reason could be that stemming/lemmatizing results in a loss of information which can be seen here. Punctuation removal does help as tokens like "corona," and "corona" won't be considered different.

(a) Word Cloud for positive sentiment words

(b) Word Cloud for negative sentiment words

(c) Word Cloud for neutral sentiment words

Figure 10: Word Clouds for punctuation and stopword removal

1.5.5 Tri-grams with Stemming

1. Training Accuracy = 98.61081766321572%

2. Validation Accuracy = 65.62405101730945%

The training accuracy increases , but validation accuracy steeply drops. Possible reason : Overfitting

1.5.6 TF-IDF with Naive-Bayes and Punctuation Removal

1. Training Accuracy = 84.00591590957109%
2. Validation Accuracy = 69.63255390221682%

TF-IDF performs better than plain unigram technique but not as good as naive bayes with basic cleaning. This could be because TF-IDF, because of high dimensions, could overfit, and then there's a loss of information as in other techniques.

1.6 Domain Adaptation

1.6.1 Using Source Data as well

Percent Size of Source	Train Accuracy	Val Accuracy
1	83.40085231756721%	45.46056991385023%
2	83.40058694057226%	45.46056991385023%
5	83.42570052312632%	46.4546056991385%
10	83.36297124275989%	47.6805831676607%
25	83.10184072667852%	47.514910536779326%
50	82.64703288951591%	48.641484426772696%
100	81.79101089588378%	49.00596421471173%

1.6.2 Without Source Data

Percent Size of Source	Train Accuracy	Val Accuracy
1	98%	35.48707753479125%
2	97.33%	39.595758780649437%
5	96.4%	44.30086149768058%
10	94.867%	48.60834990059642%
25	91.5733%	48.906560636182905%
50	88.6267%	51.55732273028496%
100	86.67%	56.03048376408217%

1.6.3 Plot of Validation Accuracies

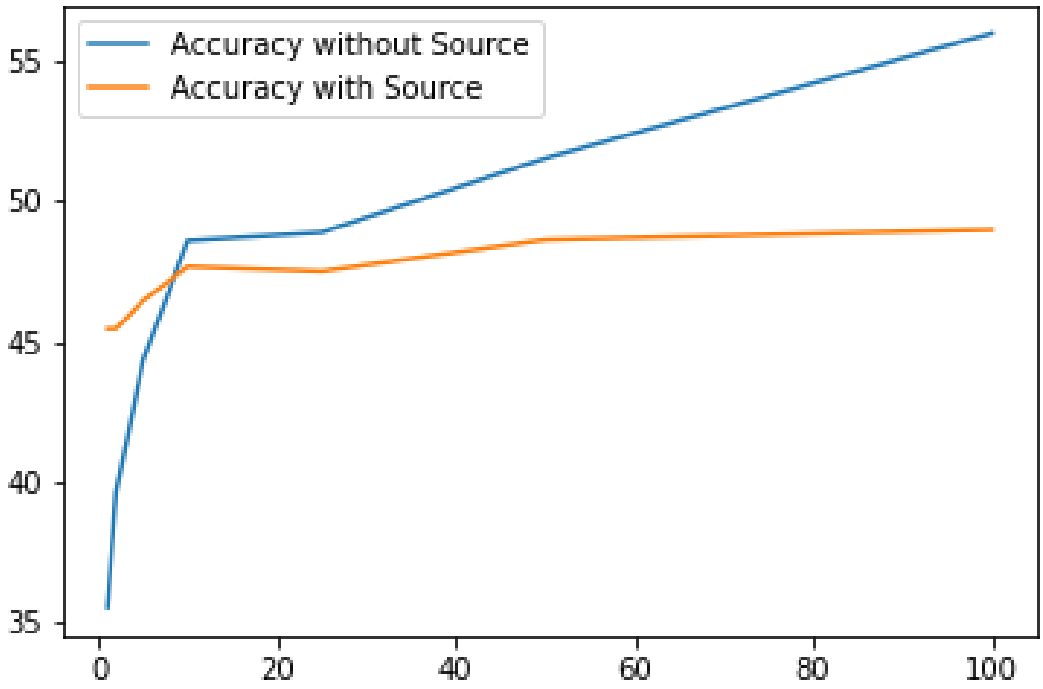


Figure 11: Plot of Validation Accuracies

We observe that accuracy without source is much less than accuracy with source initially (till percent = 10) and after that it increases. The reason behind this is that for lesser target data, when we train without source data we do not have much information about the general structure of the language or what's generally positive or negative irrespective of whether the data comes from source or target. Meanwhile, the accuracy is higher with source data because the source data provides the initial information of what's negative or positive or neutral. That helps the perform better on target validation data because as compared to just training on target, we have much more data.

When the amount of target data increases, accuracy without source surpasses accuracy with source as the model is more focused on the target space. Having data from source data-set hampers the performance of models with source data as the probabilities are distributed over a combination of source and target words while in model without source, the distribution is more focused on target words.

2 Support Vector Machines

2.1 Part A : Binary Classification

Entry Number : 2021MT10236. So, we will be working with class-0 and class-1 images.

2.1.1 Linear Kernel

We use CVXOPT package to solve our SVM optimization problem. The problem is to be formulated as follows:

$$\begin{aligned} &\alpha^T P\alpha + q^T\alpha + d \\ &G\alpha \preceq H \\ &A\alpha = b \end{aligned} \tag{1}$$

The dual in summation format is given as:

$$min\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^i y^j (x^i)^T x^j - \sum_{i=1}^{m}\alpha_i \tag{2}$$

We start by calculating the P matrix by.

$$\begin{aligned} P_{ij} &= y^i y^j (x^i)^T x^j \\ \implies P &= (y * X) \times (y * x)^T \end{aligned} \tag{3}$$

where y*X means each i-th row of X is multiplied by Yi
d is 0 and q is a vector with all -1:

$$\begin{pmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{pmatrix} \tag{4}$$

α_i is constrained as $0 \leq \alpha_i \leq c$. We can compute G and H as:

$$\begin{aligned} G &= \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -1 \end{pmatrix} \\ H &= \begin{pmatrix} c \\ c \\ \vdots \\ c \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \end{aligned} \tag{5}$$

The equality condition is $\sum_{i=1}^{m}\alpha_i y^i = 0$. A and b are computed as:

$$\begin{aligned} A &= \begin{pmatrix} y_1 & y_2 & \dots & y_m \end{pmatrix} \\ b &= 0 \end{aligned} \tag{6}$$

We can compute the required matrices and pass them to CVXOPT to get the following results:

$$\begin{aligned} \text{training time} &= 66.932s \\ nSV &= 1379 \\ \%SV &= 28.971\% \\ b &= 2.4334437936343076 \\ \text{training accuracy} &= 91.176\% \\ \text{test accuracy} &= 86.25\% \end{aligned} \tag{7}$$

Figure 12: Top-6 Support Vectors(images) plotted

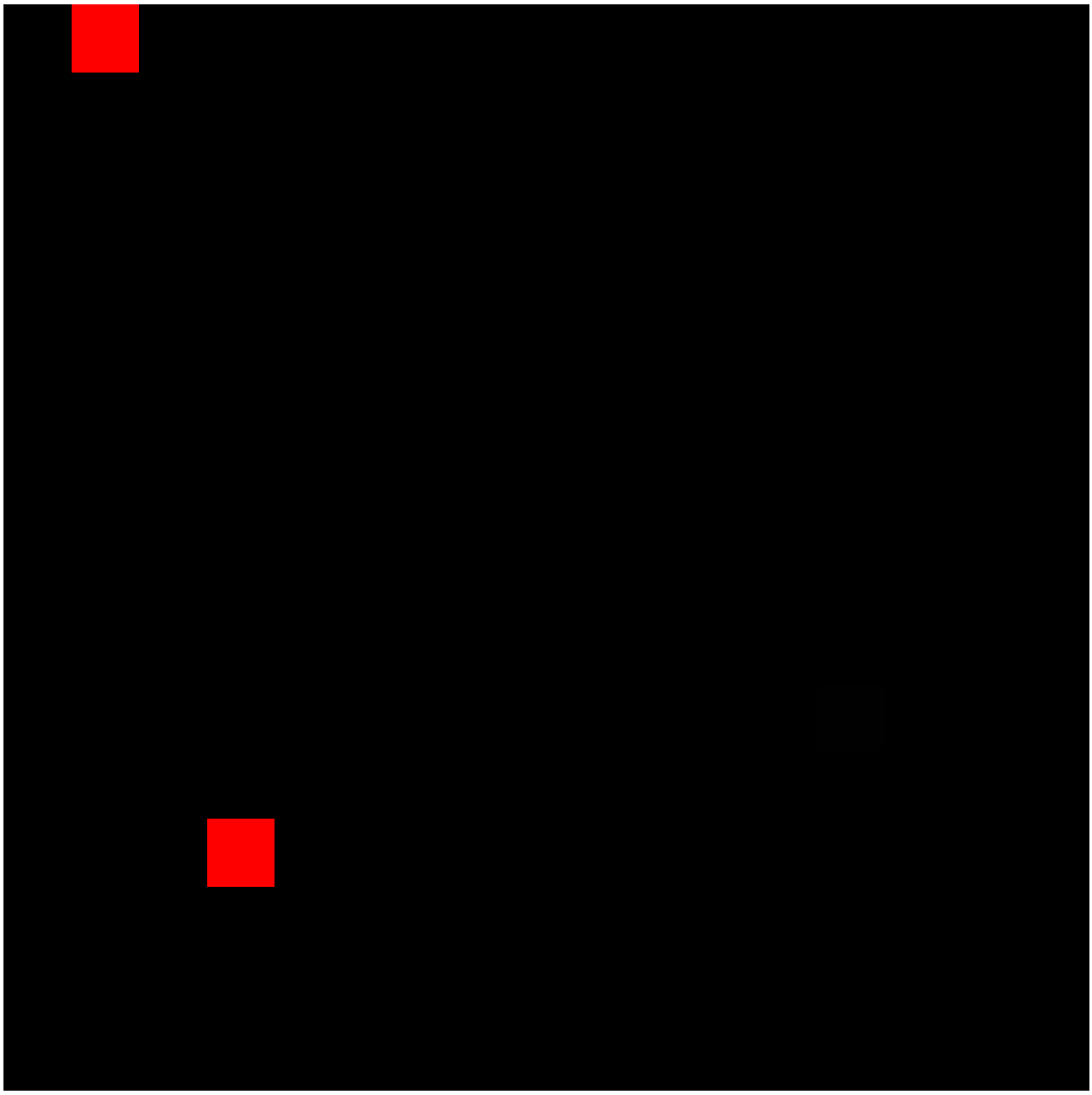


Figure 13: Plot of weight vector as an image

2.1.2 Gaussian Kernel

The only difference will be in the value of P, which is given as:

$$\begin{aligned} P_{ij} &= y^i y^j \phi(x^i)^T \phi(x)^j \\ \implies P_{ij} &= y^i y^j \exp(-\gamma ||x^i - x^j||^2) \\ \implies P_{ij} &= y^i y^j \exp(-\gamma (||x^i||^2 + ||x^j||^2 - 2x^i x^j)) \end{aligned} \tag{8}$$

We generalise this equation for computing the *product* of any two vectors X, Z of sizes n, m respectively as:

$$\mathcal{P}(X,Z) = \begin{pmatrix} ||X_1||^2 & ||X_1||^2 & (m \text{ times}) \dots & ||X_1||^2 \\ ||X_2||^2 & ||X_2||^2 & (m \text{ times}) \dots & ||X_2||^2 \\ \vdots & \vdots & \ddots & \vdots \\ ||X_n||^2 & ||X_n||^2 & (m \text{ times}) \dots & ||X_n||^2 \end{pmatrix} + \begin{pmatrix} ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \\ ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \\ \vdots & \vdots & & \vdots \\ n \text{ times} & n \text{ times} & \ddots & n \text{ times} \\ \vdots & \vdots & & \vdots \\ ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \end{pmatrix} - 2(XZ^T) \tag{9}$$

$$P = (YY^T) \circ \exp(-\gamma \mathcal{P}(X,X)) \tag{10}$$

$\circ$ is Element-wise product. The values are then again passed to the CVXOPT quadratic problem solver. Predictions are made as:

$$(\alpha \circ Y) \circ \exp(-\gamma \mathcal{P}(X_{SV}, X_{data})) + b \tag{11}$$

The results obtained are:

$$\begin{aligned} \text{training time} &= 65.105 \\ nSV &= 1919 \\ \%SV &= 40.315\% \\ b &= -8.15582215973239 \\ \text{test accuracy} &= 84.50\% \end{aligned} \tag{12}$$

Figure 14: Top-6 Support Vectors(images) plotted

2.1.3 SK-LEARN

Linear Kernel The results are:

$$\begin{aligned} \text{training time} &= 5.633 \\ nSV &= 1379 \\ b &= 0.29308328093984604 \\ \text{training accuracy} &= 90.88235294117647\% \\ \text{test accuracy} &= 85\% \\ \text{norm of difference between w for two parts} &= 0.0208 \\ \text{difference between b} &= 0.00025620636569234634 \end{aligned} \tag{13}$$

Gaussian Kernel The results are:

$$\begin{aligned} \text{training time} &= 4.576 \\ nSV &= 1916 \\ b &= -8.15717718 \\ \text{training accuracy} &= 88.61344537815126\% \\ \text{test accuracy} &= 84.5\% \end{aligned} \tag{14}$$

We can clearly see that sklearn takes much less time for training as compared to CVXOPT. The accuracies do not differ much.

2.2 Multi-Class Classification

2.2.1 One-for-One Classifier using CVXOPT

The results are:

$$\begin{aligned} \text{training time} &= 777.813 \\ \text{test accuracy} &= 55.9167\% \end{aligned} \tag{15}$$

2.2.2 Classification using SK-LEARN

The results are:

$$\begin{aligned} \text{training time} &= 160.103 \\ \text{test accuracy} &= 56.083\% \end{aligned} \tag{16}$$

The training time is much lesser than using CVXOPT (takes $1_{\frac{}{5}}$ the time). The prediction accuracy is similar to CVXOPT implementation.

2.2.3 Confusion Matrix and Plotting

	0	1	2	3	4	5
0	76	21	22	24	26	31
1	5	154	1	5	11	24
2	7	4	127	24	25	13
3	26	6	24	128	12	4
4	25	16	56	36	59	8
5	21	23	9	7	13	127

Figure 15: Confusion Matrix for Custom SVM

	0	1	2	3	4	5
0	79	21	22	24	23	31
1	6	154	1	5	10	24
2	8	4	130	25	21	12
3	26	6	24	128	12	4
4	26	16	58	36	58	6
5	22	23	12	7	12	124

Figure 16: Confusion Matrix for SKLearn

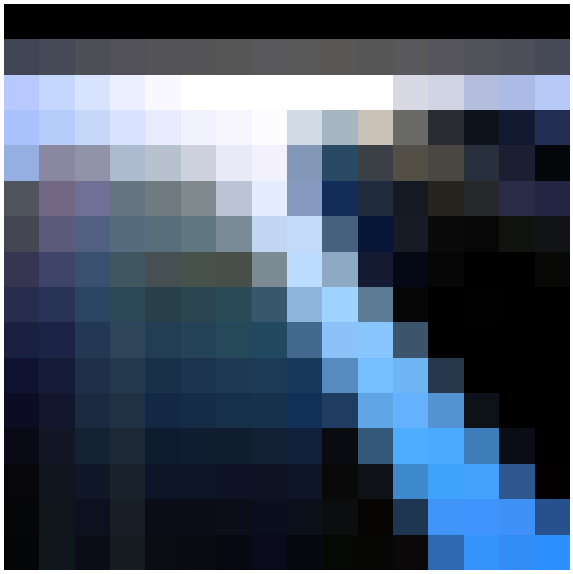
We can see that there are a lot of common entries and almost similar entries in the confusion matrix, indicating a very similar performance. We can see that the model confuses between 4 and 2 quite often. If we look at images from classes 4 and 2, we see that they are images of water bodies. Class 4 consists of tropical water bodies whereas class2 consists of iced water bodies. Because both are water bodies, the model seems to confuse between them. Also, along similar lines, model does not confuse much between 2 and 0 as 0 consists of images of buildings.

We conclude that images with similar features are the most confused by the model. Another example is 5 and 0, both have man-made features like buildings. The results do make sense.

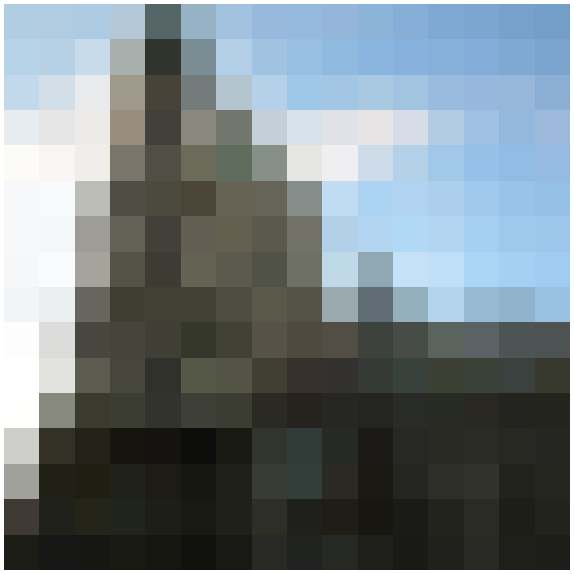
I have plotted 12 randomly chosen misclassificed images.

(a) 4 as 0

(b) 0 as 2



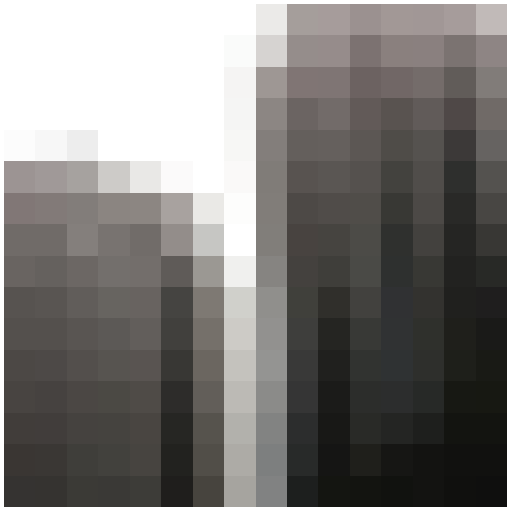
(c) 0 as 4



(d) 0 as 1

(e) 3 as 0

(f) 0 as 1



(a) 2 as 5

(b) 4 as 0

(c) 4 as 5

(d) 0 as 5

(e) 4 as 0

(f) 1 as 2

Figure 18: 12 misclassified images

2.2.4 K-Fold Cross Validation

The average of K-Fold Cross Validation for different values of c are:

$$\begin{aligned} c = 10^{-5} &: 15.896358543417366 \\ c = 0.001 &: 15.896358543417366 \\ c = 1 &: 54.71288515406163 \\ c = 5 &: 57.92016806722688 \\ c = 10 &: 5923669467787115 \end{aligned} \tag{17}$$

The training accuracies are:

$$\begin{aligned} c = 10^{-5} &: 40.25, \\ c = 0.001 &: 40.25 \\ c = 1 &: 56.0833 \\ c = 5 &: 58.5833 \\ c = 10 &: 59.8333 \end{aligned} \tag{18}$$

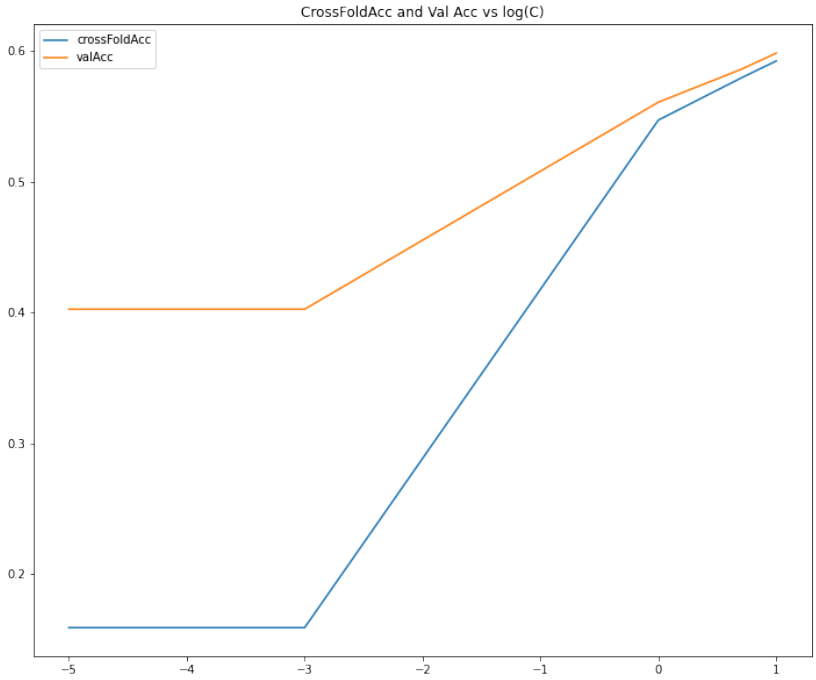


Figure 19: Cross-fold and validation accuracy vs log(C)

The accuracy for smaller values of c are fairly low for cross validation and testing accuracy. This is an example of underfitting. For larger values of c there is overfitting since the model performs relatively much better on the validation data compared to the testing data.