Author: Yatharth Kumar
Entry Number: 2020CS10413

Question 1

a)

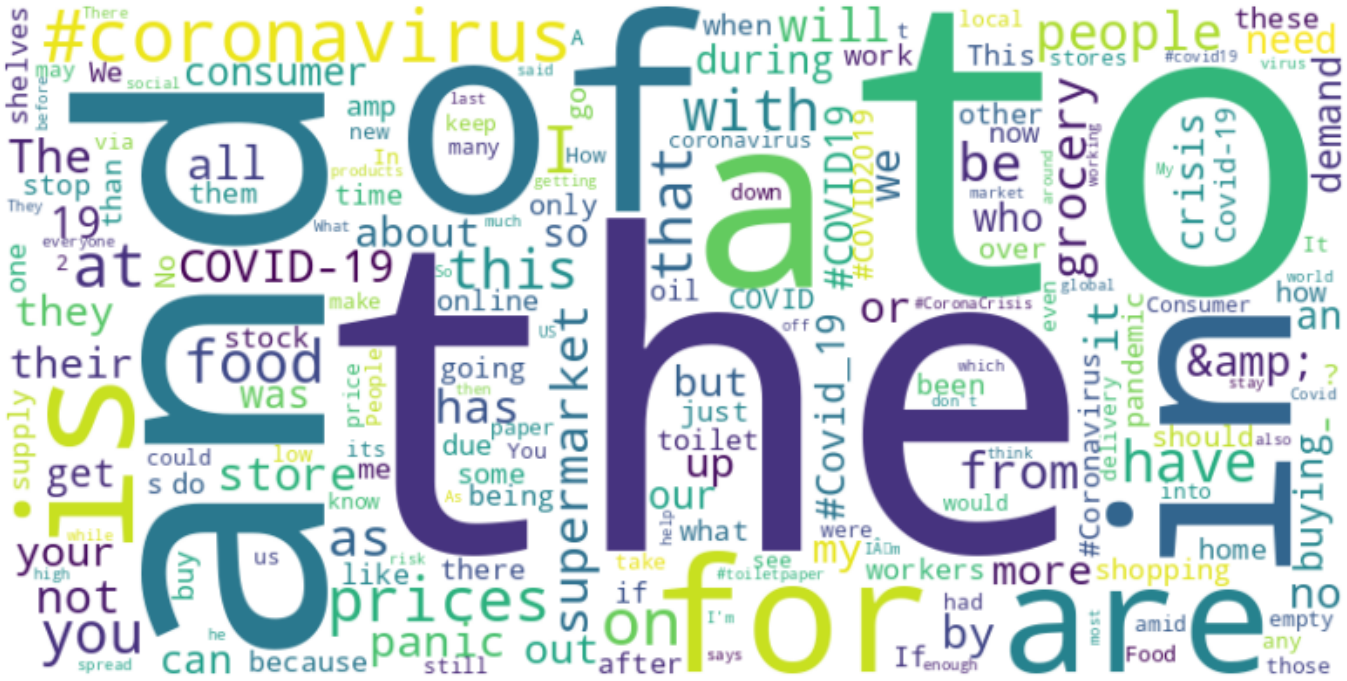
Accuracy for training data: 85.04648214663004%
Accuracy for test data: 67.05132098390525%

Word Clouds for Training Data

Neutral

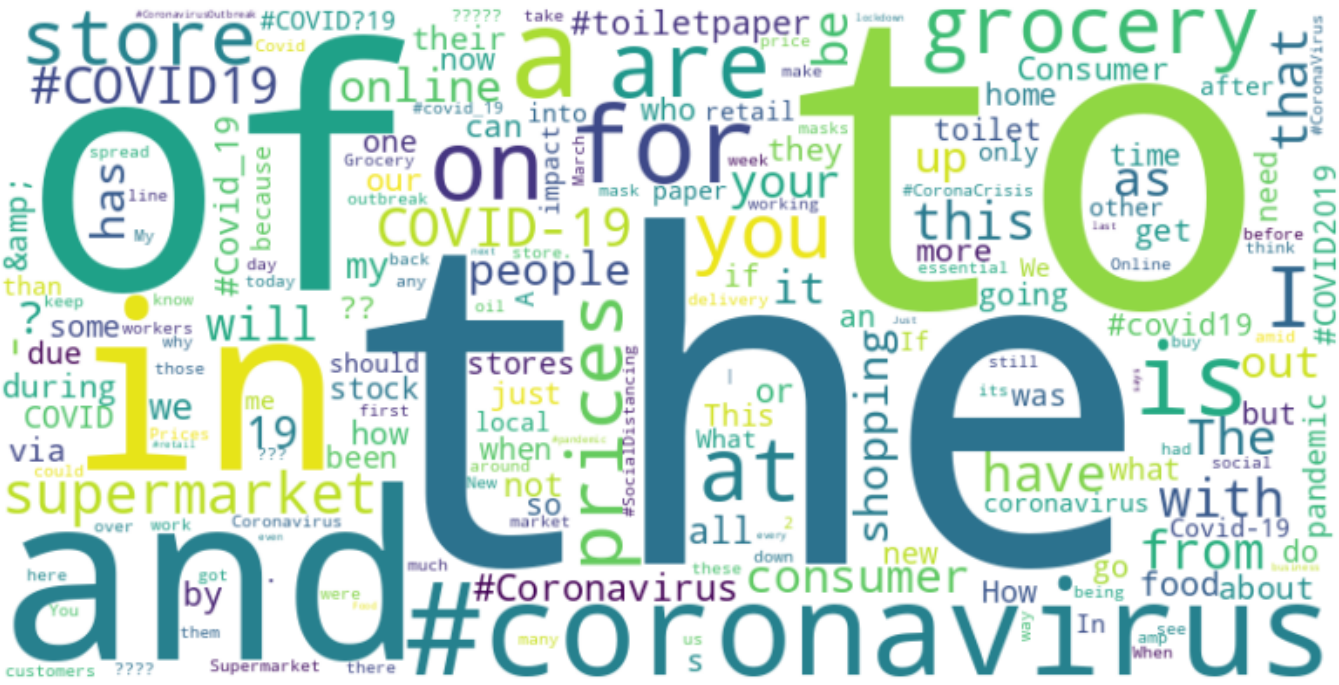
Positive

Negative

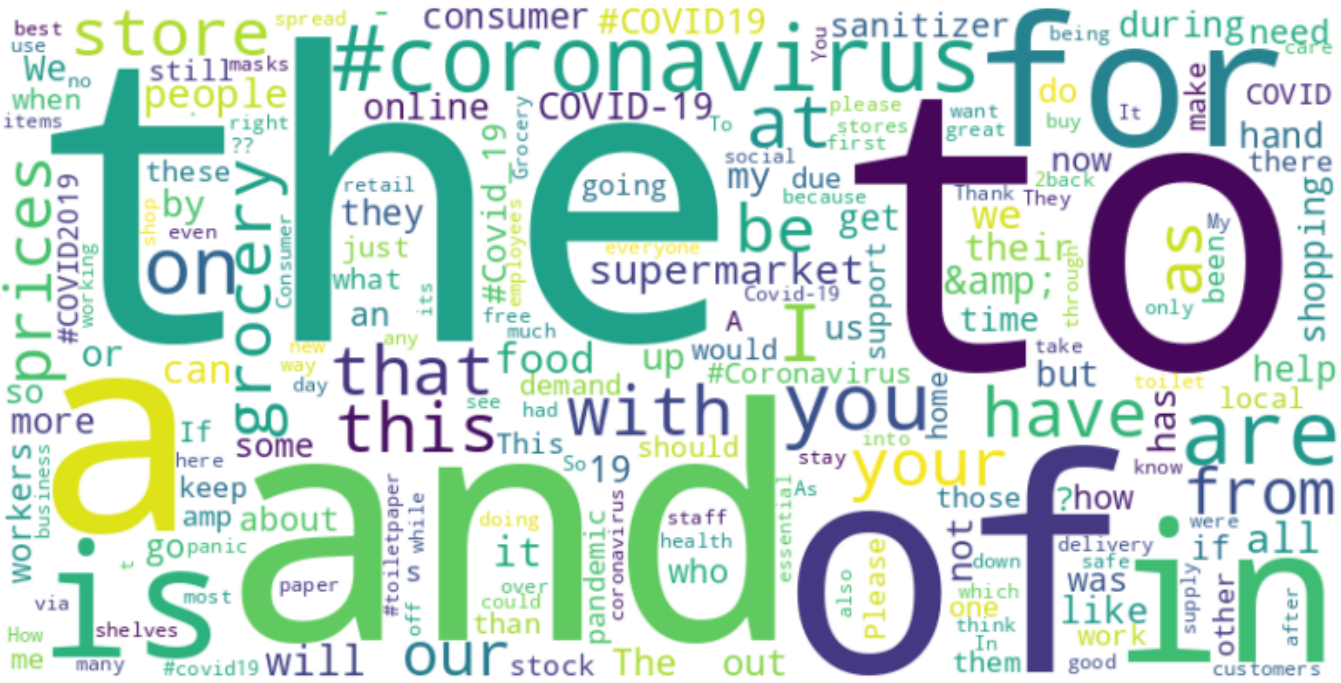


Word Clouds for Training Data

Neutral



Positive



Negative

b)

Random Heuristic:
Test Accuracy: 33.768600060734894%
Train Accuracy: 33.2426579336573%

Positive Heuristic:
Test Accuracy: 43.85059216519891%
Train Accuracy: 43.84639763363617%

Improvement done by Naive Bayes Algorithm over Random prediction - 33.29 %
Improvement done by Naive Bayes Algorithm over Positive prediction - 23.21 %

c)

Rows denote actual labels and columns represent predicted labels.

Naive Bayes

Test Accuracy: 67.05132098390525%
Confusion Matrix:

	NEGATIVE	NEUTRAL	POSITIVE
Negative	910	17	305
Neutral	178	101	338
Positive	226	21	1197

Train Accuracy: 85.04648214663004%
Confusion Matrix:

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

Random Heuristic:

Test Accuracy: 31.52140904949894%
Confusion Matrix:

	NEGATIVE	NEUTRAL	POSITIVE
Negative	389	436	407
Neutral	212	186	219
Positive	500	481	463

Train Accuracy: 33.662581871962816%
Confusion Matrix:

	NEGATIVE	NEUTRAL	POSITIVE
Negative	4829	4690	4647
Neutral	2368	2400	2328
Positive	5526	5559	5517

Positive Heuristic:

Test Accuracy: 43.85059216519891%
Confusion Matrix:

	NEGATIVE	NEUTRAL	POSITIVE
Negative	0	0	1232
Neutral	0	0	617
Postitive	0	0	1444

Train Accuracy: 43.84639763363617%
Confusion Matrix:

	NEGATIVE	NEUTRAL	POSITIVE
Negative	0	0	14166
Neutral	0	0	7096
Positive	0	0	16602

In all the confusion matrices we've discussed, the box where actual positive and predicted positive values align has the highest number along the diagonal. This shows that in all cases, the algorithms accurately predict positive examples most of the time. This means that the model is most accurate at predicting this particular class.

d)

Validation Set Accuracy: 66.56544184634072%

Neutral

Positive

Negative

After applying stemming and removing stop words, I noticed significant changes in our word clouds. However, these changes did not lead to improved accuracies in our model. In fact, our accuracies slightly decreased.

Surprisingly, when we opted to only lowercase words and remove stopwords without stemming, we achieved a considerably higher training accuracy and higher validation accuracy compared to stemming and removing stop words. This result suggests that stemming may not be as effective for our specific dataset.

e)

Accuracy after addition of Bigrams along with Unigrams:
Accuracy on training data: 95.3649904922882%
Accuracy on validation data: 66.56544184634072%

After the addition of bigrams along with unigrams, the training accuracy jumps to 95.36% from previous 85%, mostly because bigrams capture the notion of both syntax and semantics. They are generally better compared to unigrams.
However, the validation accuracy is pretty much similar and has actually decreased by nearly half a percent

Accuracy after further addition of TF-IDF:
Accuracy on training data: 96.1308894992605
Accuracy on validation data: 69.78439113270574

I used TF-IDF (Term Frequency-Inverse Document Frequency) to assess word importance in text documents:

- **Term Frequency (TF)** measures word frequency within a document.
- **Inverse Document Frequency (IDF)** gauges word importance across a document collection.
- Combining TF and IDF yields a score that signifies word significance in a document compared to the entire collection.
- I used a weighted combination of Conditional Word Probabilities along with TF-IDF Score to predict the label.

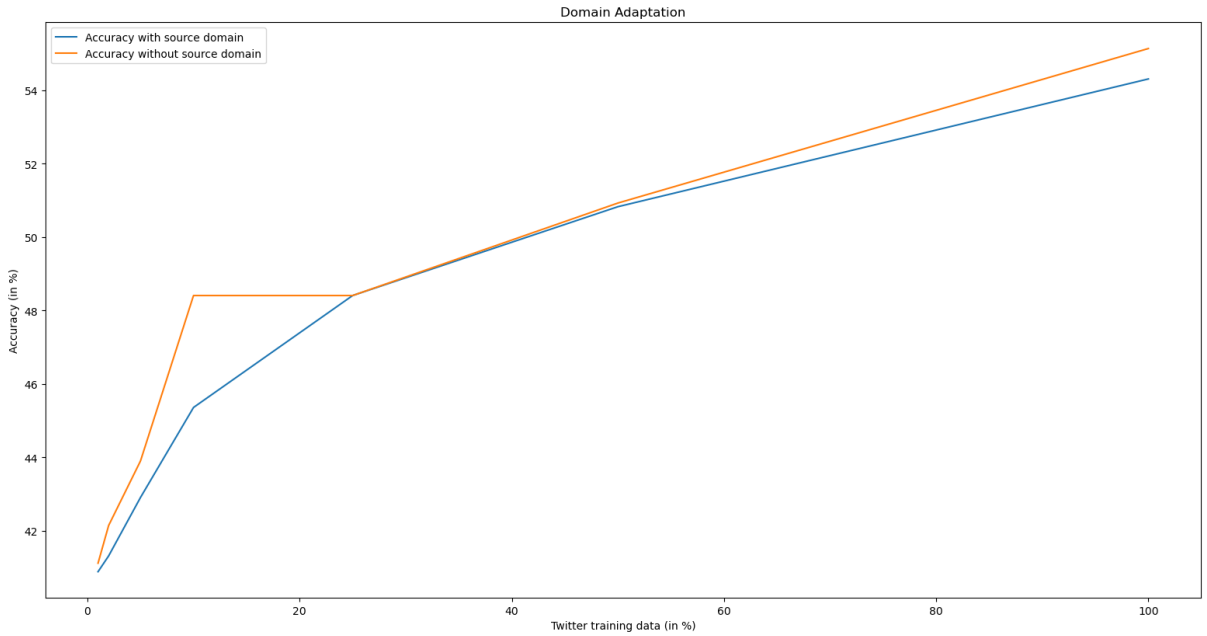
This new feature increases both my training and validation accuracy. My training accuracy jumps by 1% extra and validation accuracy increased by 3%.

f)

Accuracy with source domain:
Accuracy with corona and 1% twitter training data: 40.88800530152419
Accuracy with corona and 2% twitter training data: 41.31875414181577
Accuracy with corona and 5% twitter training data: 42.909211398277
Accuracy with corona and 10% twitter training data: 45.36116633532141
Accuracy with corona and 25% twitter training data: 48.40954274353877
Accuracy with corona and 50% twitter training data: 50.828363154406894
Accuracy with corona and 100% twitter training data: 54.30748840291584

Accuracy without source domain:

Accuracy with only 1% twitter training data: 41.119946984758116
Accuracy with only 2% twitter training data: 42.14711729622266
Accuracy with only 5% twitter training data: 43.90324718356528
Accuracy with only 10% twitter training data: 48.40954274353877
Accuracy with only 25% twitter training data: 48.40954274353877
Accuracy with only 50% twitter training data: 50.92776673293572
Accuracy with only 100% twitter training data: 55.13585155732274



The graph illustrates that as we increase the proportion of target domain data, our validation accuracy in the target domain improves, as expected. When we train only on the target domain without the source data, we achieve better validation accuracies because the model is tailored to that specific domain.

However, as we incorporate more target data, the performance gap between the two approaches narrows. This happens because the target-only model benefits from all the necessary information within the target domain, while the first model has additional information from the source domain, making it seem initially better.

In essence, this demonstrates how domain adaptation can be used to build effective models for a target domain with limited data by leveraging a portion of data from a source domain.

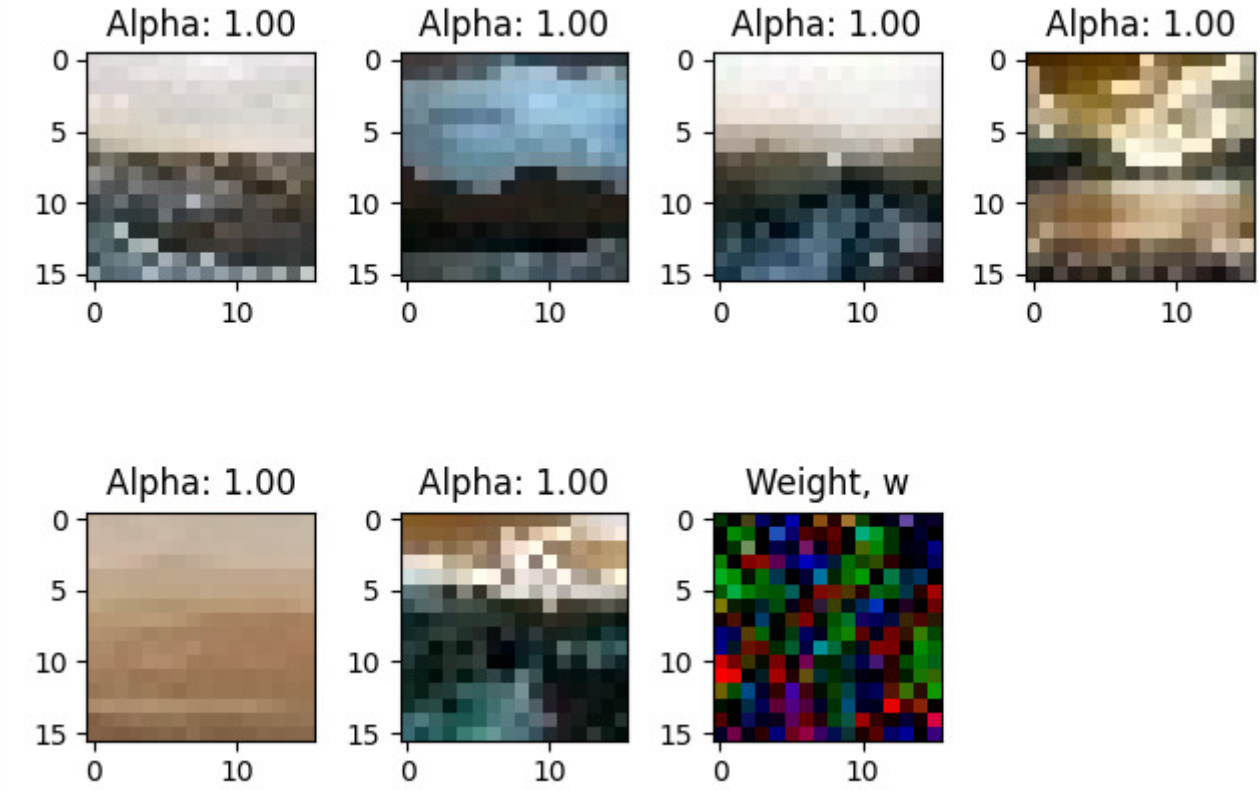
Question 2

Binary Classification

The last digit of my entry number is 3. So I am making a classifier for class 3 and class 4.

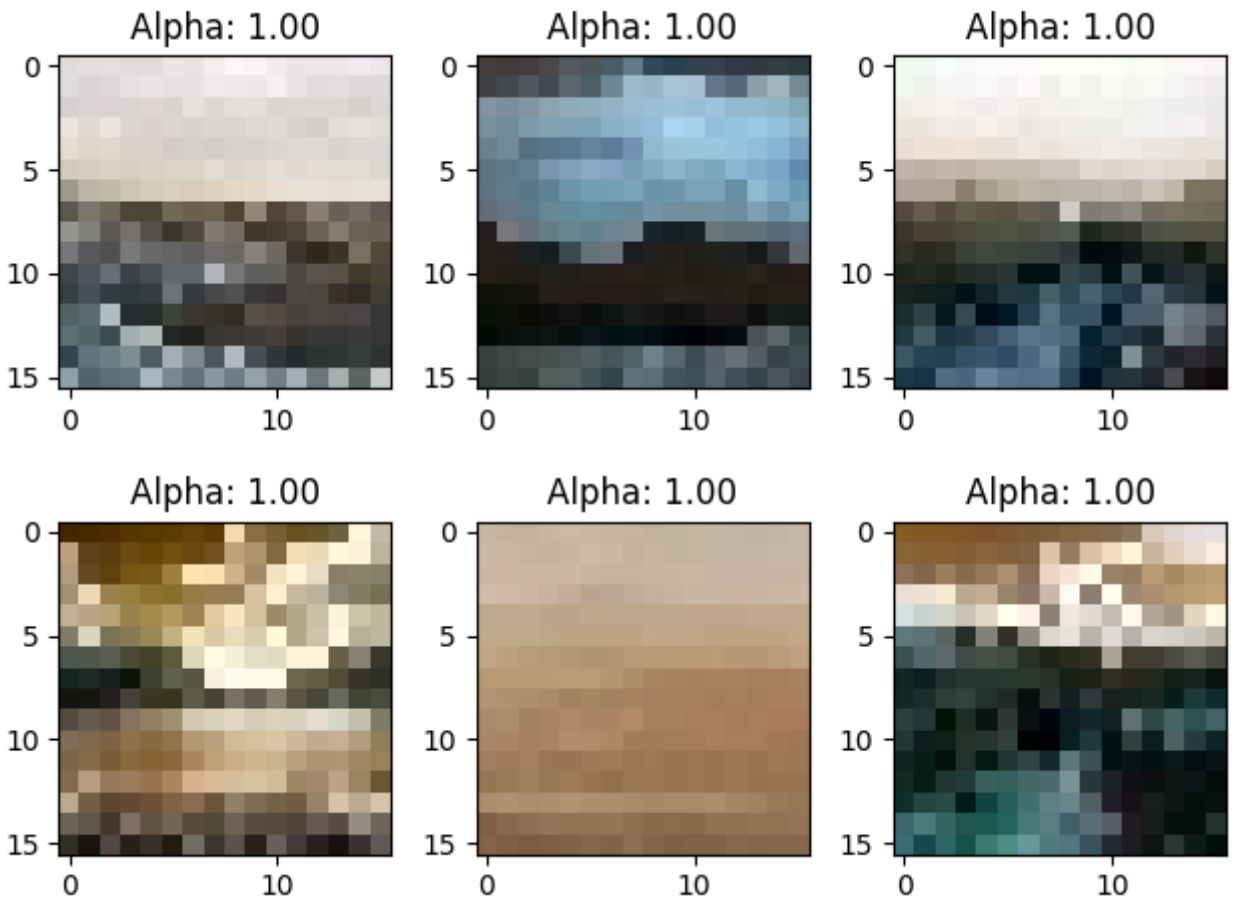
a)

Count of support vectors = 2905
Support Vector % = 61.029411764705884
b = 0.759490905995707
Accuracy: 72.25%



b)

Count of support vectors = 3410
Support Vector % = 71.63865546218487
b = -3.078119549810012
Accuracy: 76.5%



Although the accuracies with both Linear and Gaussian kernel are almost comparable, the gaussian kernel outperforms the linear kernel by 4%. Using a Gaussian (RBF) kernel often results in better accuracy for image classification because images contain complex, non-linear patterns. The Gaussian kernel is flexible and can adapt to these patterns, making it suitable for capturing intricate details and relationships. Image datasets have high dimensionality, and the Gaussian kernel handles this complexity effectively. It's justified to get better results with the Gaussian kernel when working with images due to its ability to model non-linear relationships and adapt to the data's complexity.

c)

Linear Kernel using scikit-learn:

Count of support vectors = 2904
Bias with linear kernel = 0.79824293
Accuracy: 72.25%
Common Support Vectors: 2904

Gaussian Kernel using scikit-learn:

Count of Support Vectors: 3402
Bias with gaussian kernel = -3.0045752
Accuracy: 78.0%
Common Support Vectors: 3402

The weight and bias values obtained using the CVXOPT library (0.759) and scikit-learn (0.798) for the linear kernel are quite close to each other. The bias values differ by only 0.039.

SVM Running Time Comparison

KERNEL	LIBRARY	RUNNING TIME (SECONDS)
Linear	CVXOPT	1m24.235s
Linear	scikit-learn	10.040s
Gaussian (RBF)	CVXOPT	1m16.509s
Gaussian (RBF)	scikit-learn	7.921s

Multi-Class Classification

a)

Validation Set Accuracy: 55.58333333333333%
Time Taken: 24m46.157s

b)

Validation Set Accuracy: 55.75%
Time Taken: 1m26.100s

c)

Rows denote actual labels and columns represent predicted labels.

Confusion Matrix for part (a):

	LABEL 0	LABEL 1	LABEL 2	LABEL 3	LABEL 4	LABEL 5
Label 0	77	21	21	29	19	33
Label 1	4	151	1	6	12	26
Label 2	12	4	125	28	19	12
Label 3	24	6	25	128	12	5
Label 4	18	18	60	37	62	5

	LABEL 0	LABEL 1	LABEL 2	LABEL 3	LABEL 4	LABEL 5
Label 5	26	23	11	8	8	124

Confusion Matrix for part (b):

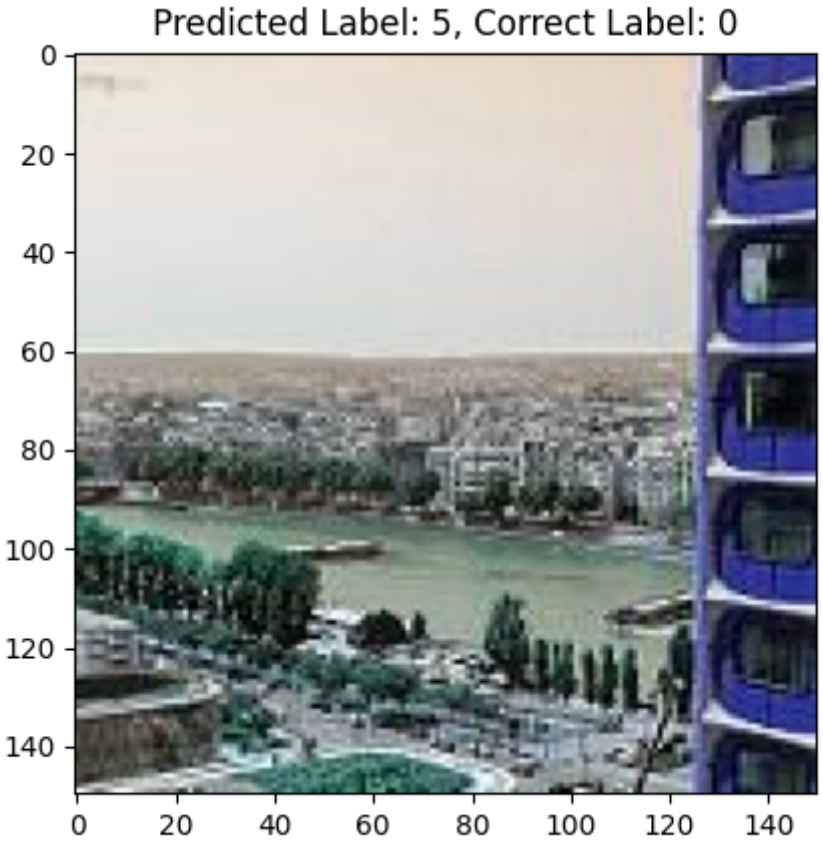
	LABEL 0	LABEL 1	LABEL 2	LABEL 3	LABEL 4	LABEL 5
Label 0	75	22	21	28	21	33
Label 1	5	150	1	5	12	27
Label 2	12	4	123	26	22	13
Label 3	25	6	25	126	13	5
Label 4	19	17	55	33	71	5
Label 5	25	23	11	8	9	124

Class 0: Buildings
Class 1: Trees
Class 2: Ice Mountains/Glaciers
Class 3: Sky/Nature/Mountains(mostly top view)
Class 4: Oceans/water bodies
Class 5: Metro Cities

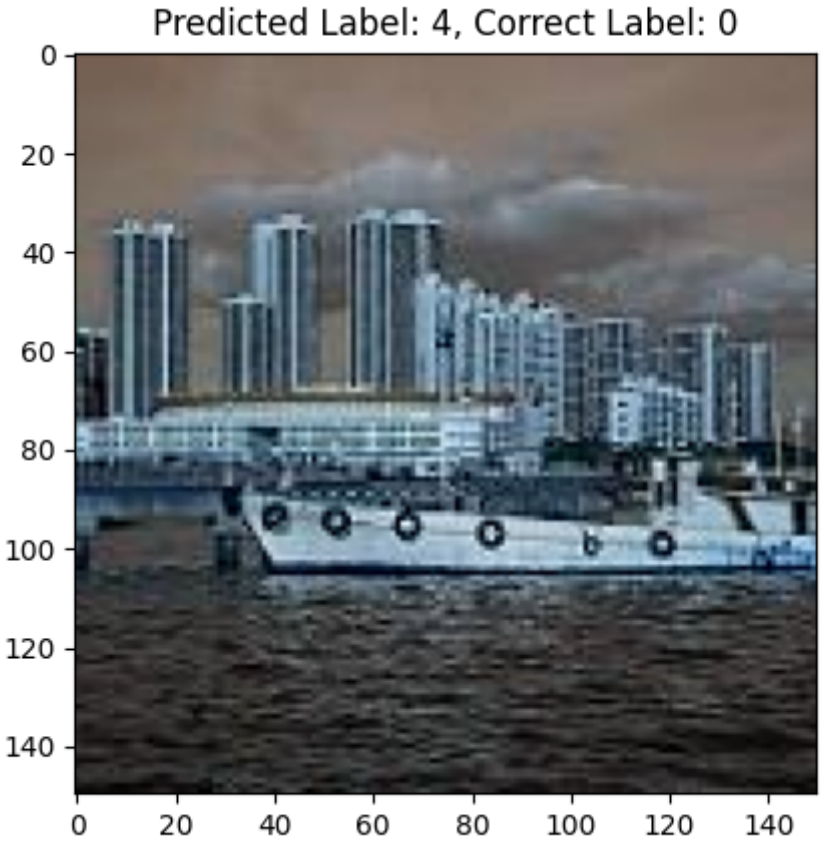
The confusion matrices reveal some interesting patterns in the misclassifications:

1. **Class 0 and Class 5**: These classes have the most misclassified examples, which is understandable as they share common features related to urban environments, buildings, lights, and glass. Distinguishing between them can be challenging due to these similarities.
2. **Classes 2, 3, and 4**: These classes are often confused because they represent natural scenes, such as mountains, sky, and ice. Their visual similarities can lead to misclassifications, especially between Class 2 and Class 4, where it's difficult to differentiate liquid and solid ice in low light conditions.
3. **Class 1**: Class 1 seems to perform better in classification, possibly because it predominantly contains trees, making it distinct from the other classes. Its unique characteristics make it less prone to misclassification.

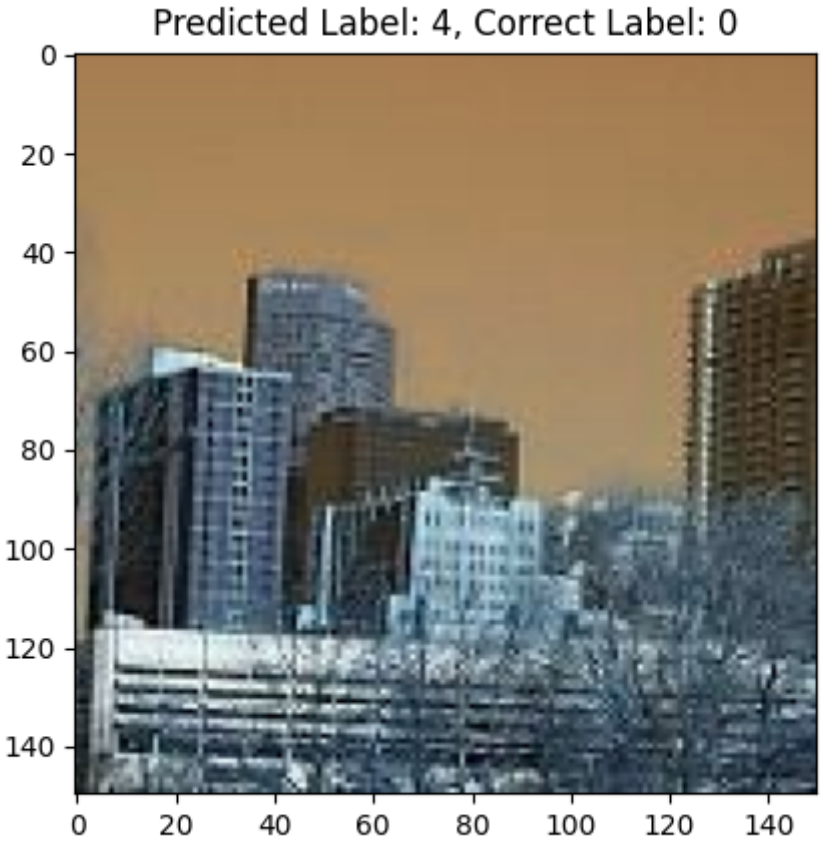
Now, I present 12 examples of misclassified images.



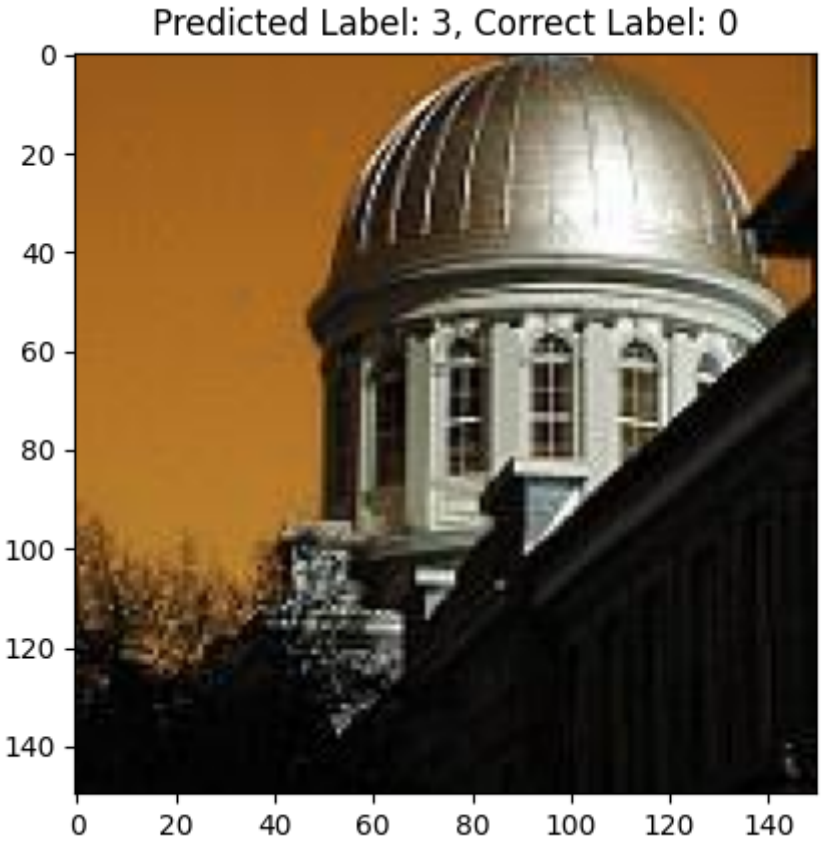
The presence of so many buildings also resemble the metro cities, which throws off the model into thinking that it is Class 5, when in fact it is Class 0.



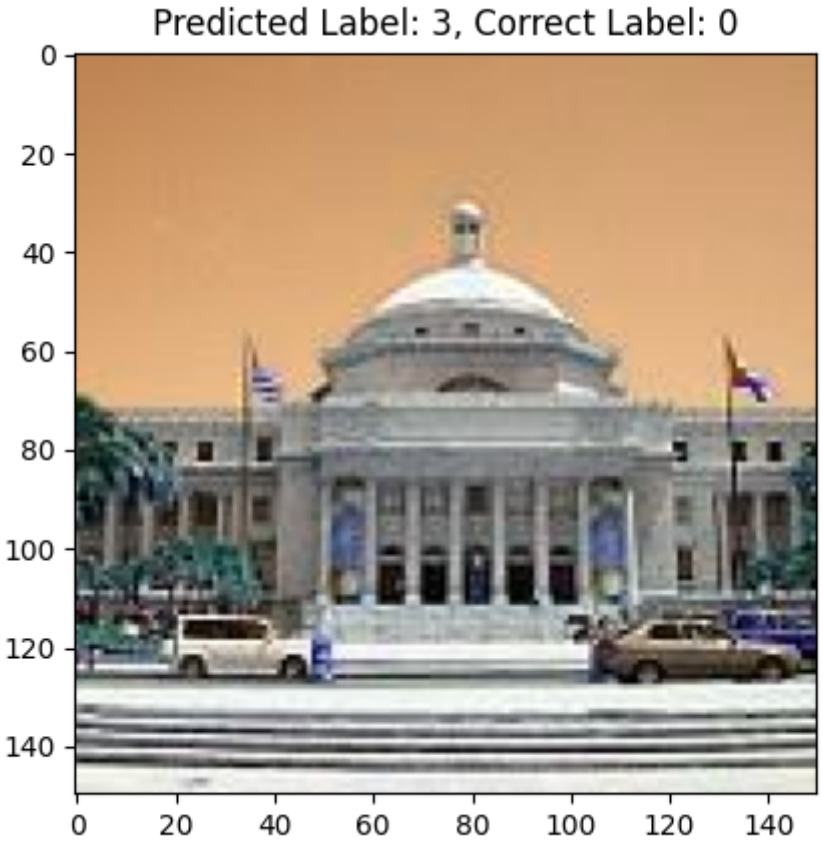
Since this image contains oceans/water bodies, this image has been classified into Class 4, whereas it is Class 0. This image is ambiguous, it can be classified as either Class 4 or Class 0.



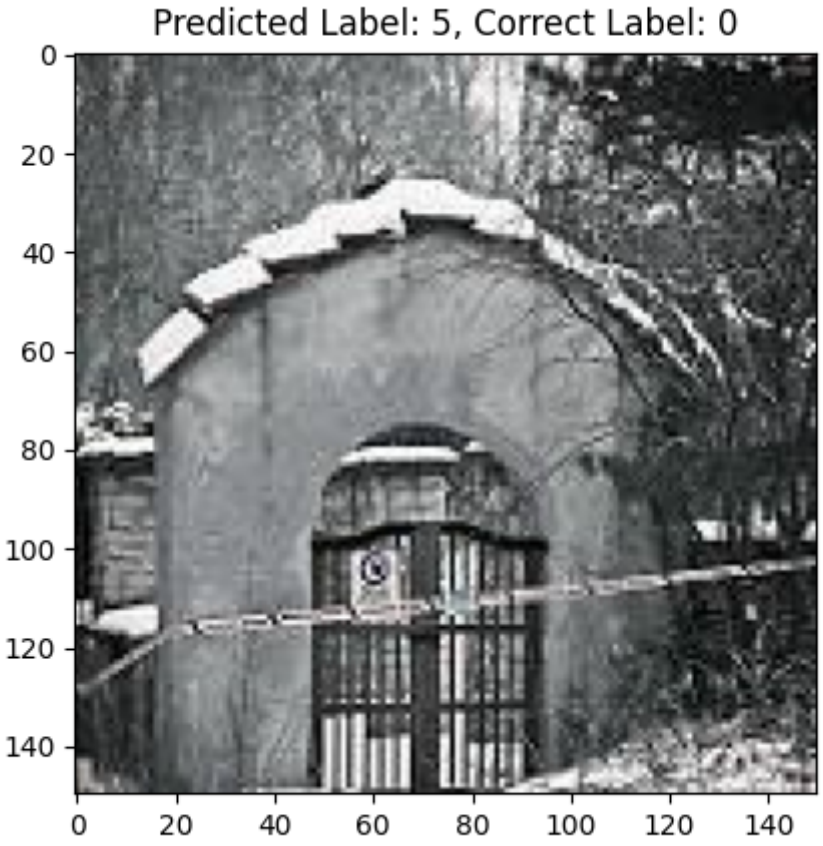
The trees at the bottom appear blue similar to the image above making the model think that this image contains oceans/water bodies, and hence classifies it into Class 4, whereas it is Class 0.



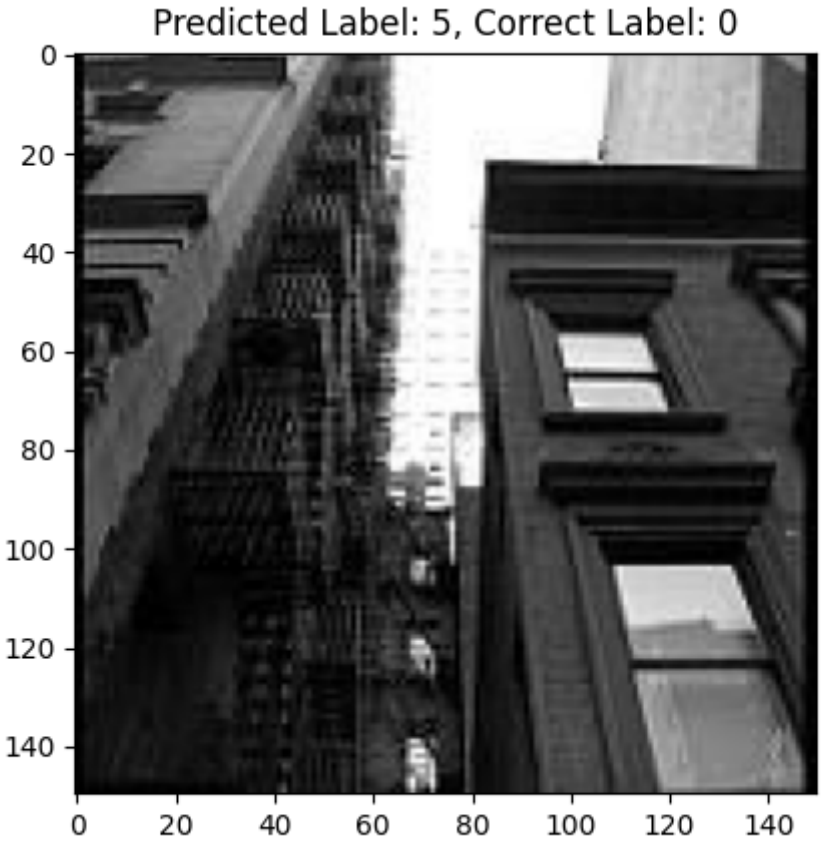
This image contains significant amount of sky, therefore it has been classified into Class 3 instead of Class 0.



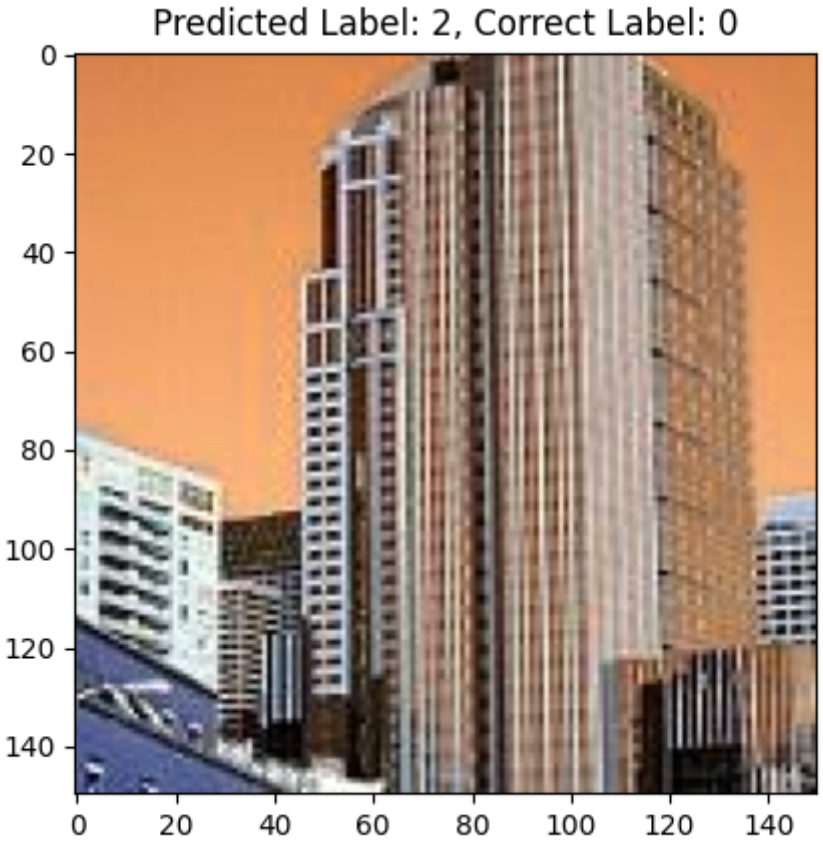
Similar to image above, this image also contains significant amount of sky, therefore it has been classified into Class 3 instead of Class 0.



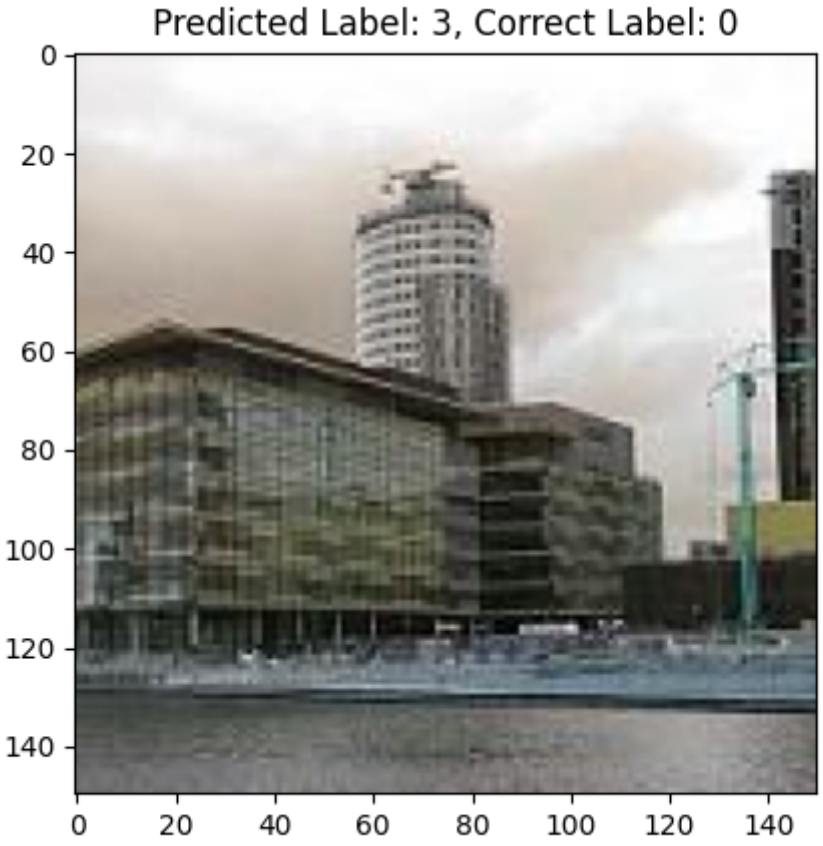
The presence of buildings also resemble the metro cities, which throws off the model into thinking that it is Class 5, when in fact it is Class 0.



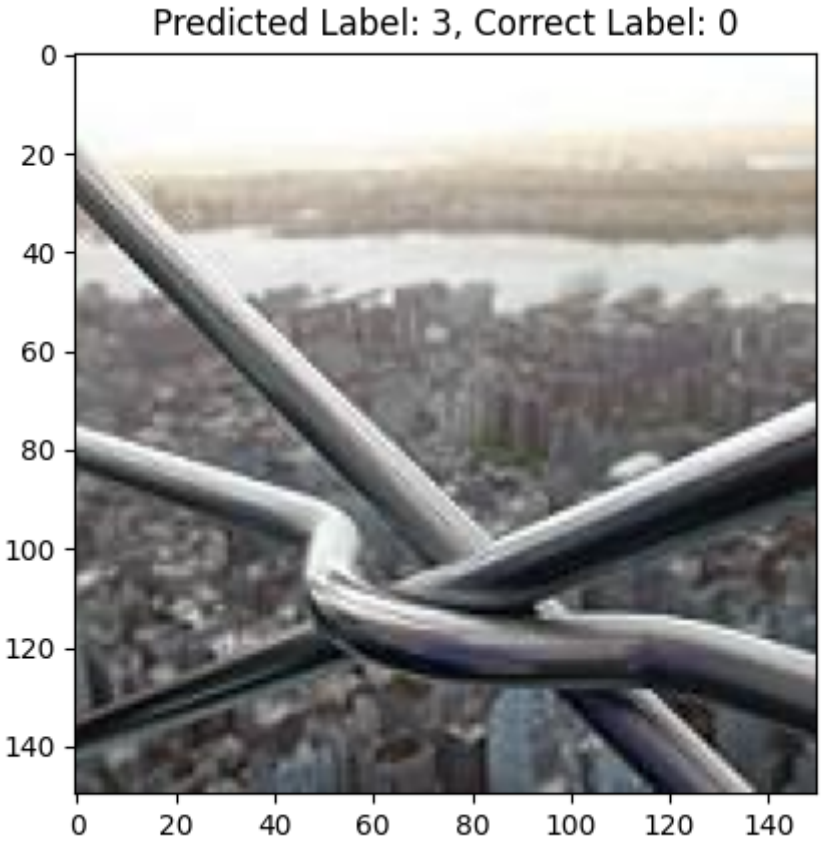
The presence of buildings here also resemble the metro cities, which throws off the model into thinking that it is Class 5, when in fact it is Class 0.



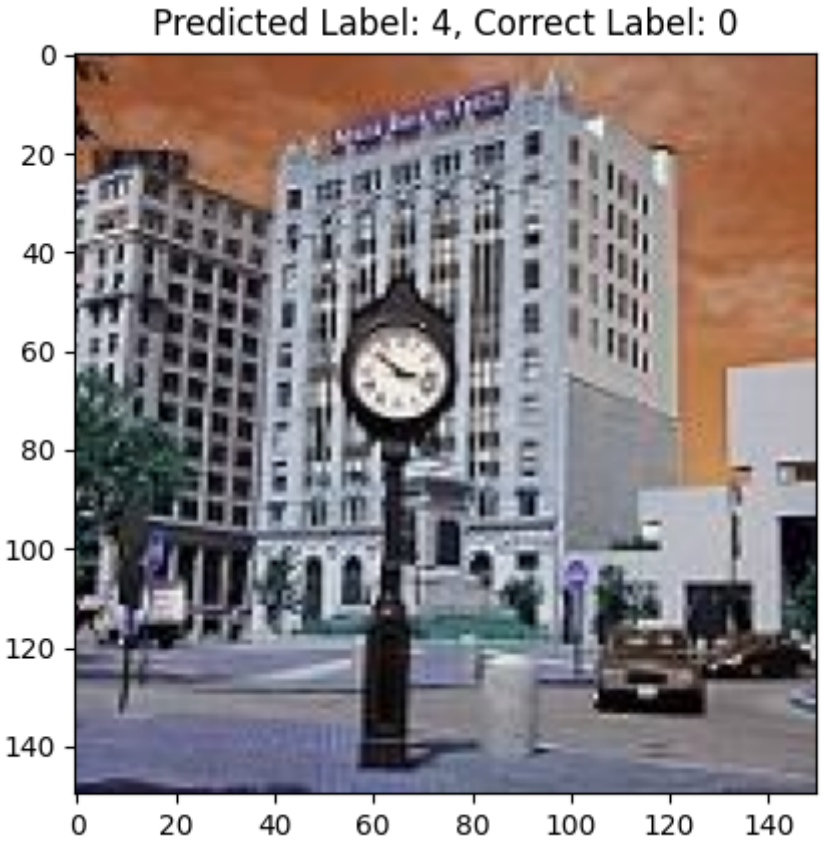
The shiny surface of the building makes the model believe that it represents a glacier and hence classifies it as Class 2, when it is in fact Class 0.



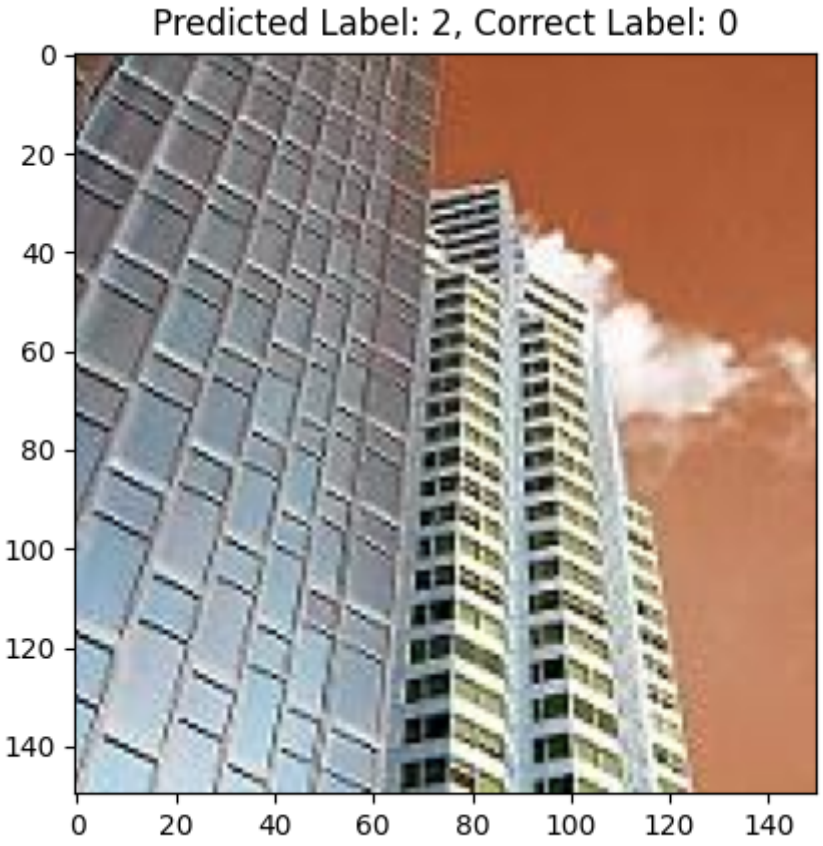
This image also contains significant amount of sky, therefore it has been classified into Class 3 instead of Class 0.



Similar to image above, this image also contains significant amount of sky and in fact it contains a top view as well, therefore it has been classified into Class 3 instead of Class 0.



The blueish tint at the bottom part of the image makes the model think that this image contains oceans/water bodies, and hence classifies it into Class 4, whereas it is Class 0.



The shiny surface of the building makes the model believe that it represents a glacier and the clouds at the top too add to that belief and hence classifies it as Class 2, when it is in fact Class 0.

d)

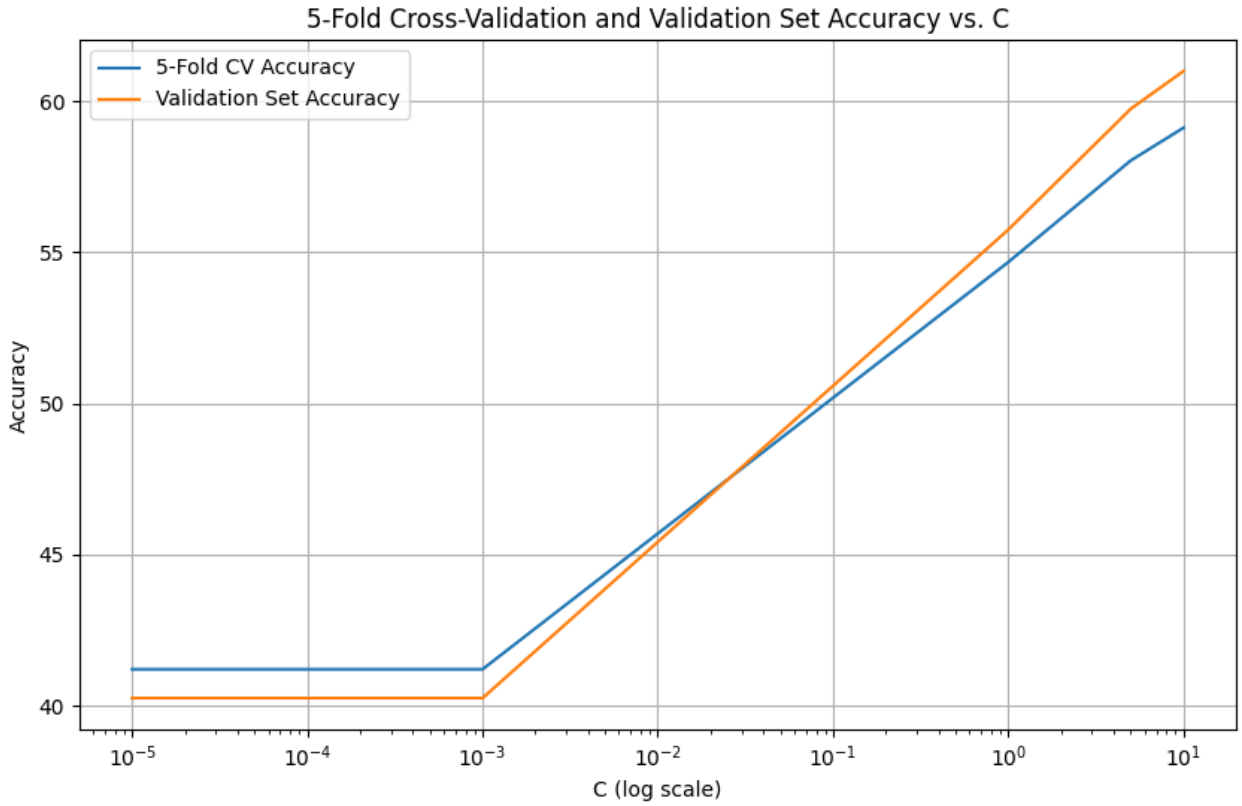
C = 1e-05
5-Fold Cross Validation Accuracy: 41.20448179271709%
Validation Accuracy: 40.25%

C = 0.001
5-Fold Cross Validation Accuracy: 41.20448179271709%
Validation Accuracy: 40.25%

C = 1
5-Fold Cross Validation Accuracy: 54.67787114845939%
Validation Accuracy: 55.75%

C = 5
5-Fold Cross Validation Accuracy: 58.0392156862745%
Validation Accuracy: 59.75%

C = 10
5-Fold Cross Validation Accuracy: 59.12464985994398%
Validation Accuracy: 61.0%



Observing the impact of different values of C on our model's performance, we find that both the validation accuracy and the 5-fold cross-validation accuracy tend to increase with higher C values. When we increase C, we effectively relax the constraints on our support vectors, allowing them to encompass training examples that might not be linearly separable. This flexibility can be advantageous, but it's essential to strike the right balance.

However, it's important to note that boosting C excessively can lead to a counterproductive outcome. A significantly large C might result in overfitting, where our model finds a somewhat random separator that doesn't generalize well to unseen data. Hence, the choice of C is crucial to fitting our problem effectively.

The value of C that gives the best 5 -fold cross-validation accuracy is 10 corresponding to a logarithmic value of 1. This value of C also gives the best validation set accuracy.

Regarding the two accuracy metrics, the validation accuracy and 5-fold cross-validation accuracy are quite similar, implying that using one portion of the training data for validation and the rest for training is essentially equivalent to the 5-fold cross-validation approach in this scenario.