PROJECT REPORT

on

Intel Image Classification

Image Scene Classification of Multiclass

(2019-20)

Submitted to:

Submitted by:

Mr. Rahul Singh

Mr. Pankaj Rawat

Mr. Manjul Joshi

TABLE OF CONTENTS

Objective

The objective of this project to recognize the given images and classify the images among the class of 6 classes that are predefined as such. It is a multi-class classification problem where we have used the Image Scene Classification of Multiclass dataset. The Image Scene Classification of Multiclass is presented here follows the folder wise distribution with labels. The Image Scene Classification of Multiclass database of hand gestures represent a multi-class problem with 6 classes of images.

The problem we are investigating is image recognition with classification through machine learning algorithm. Being able to recognize images according to classes is an interesting machine learning problem while simultaneously being extremely useful for various purposes as it signifies how to distribute the images in groups in a sequential order of representation with minimal data inconsistency.

The dataset format is patterned to match closely with the formation of two sets of data.
First set of data is useful for training the data in the categorical distribution of data according to folder. The second set of data is useful for testing as to how much it is beneficial for classifying the images according to the arrangement of images in folder.

A robust visual recognition algorithm could provide not only new benchmarks that challenge modern machine learning methods such as Convolutional Neural Nets and Logistic Regression but also could help us to synchronously use the data and compare it accordingly and have it run the classification of images and help us minimize the inconsistency of data and improvise accordingly in an very profound and meaningful manner without any setbacks.

APPROACH

We have used Logistic Regression and Artificial Neural Network for classifying the images in the given six categories.

Tool used:

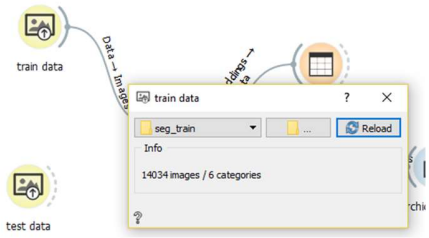
For this project we used the orange tool , Orange is a platform built for mining and analysis on a GUI based workflow. This signifies that you do not have to know how to code to be able to work using Orange and mine data, crunch numbers and derive insights.

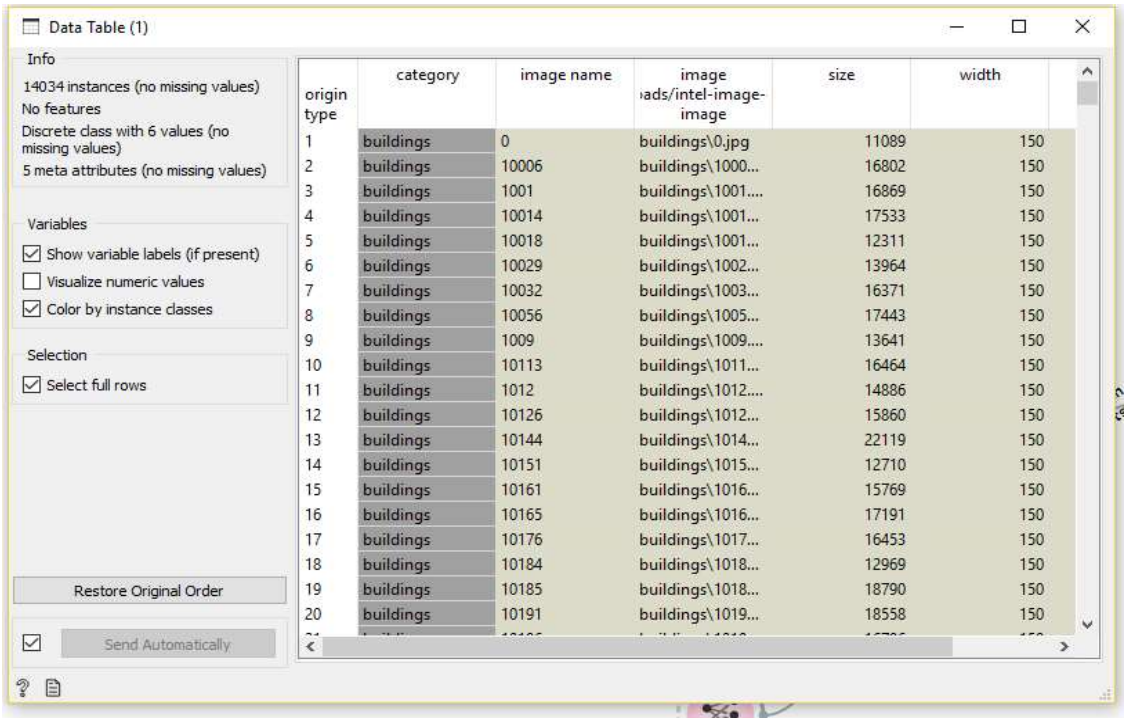
You can perform tasks ranging from basic visuals to data manipulations, transformations, and data mining. It consolidates all the functions of the entire process into a single workflow. The best part and the differentiator about Orange is that it has some wonderful visuals. You can try silhouettes, heat-maps, geo-maps and all sorts of visualizations available.

The steps involved in this project is mentioned below:

Understanding the dataset:

The dataset format is in form of a folder containing 6 folders which have images distributed in form of .jpeg form and are labled by using the folders the images are present in and are classified as such.



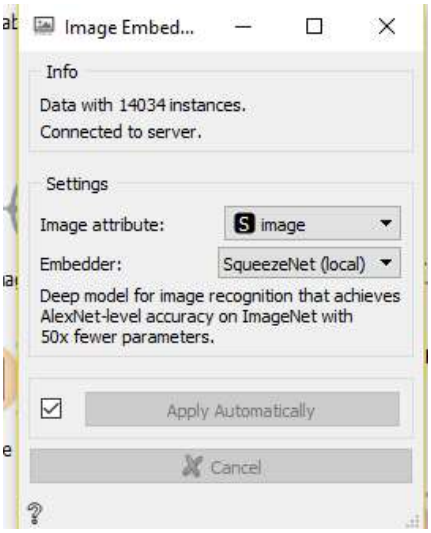


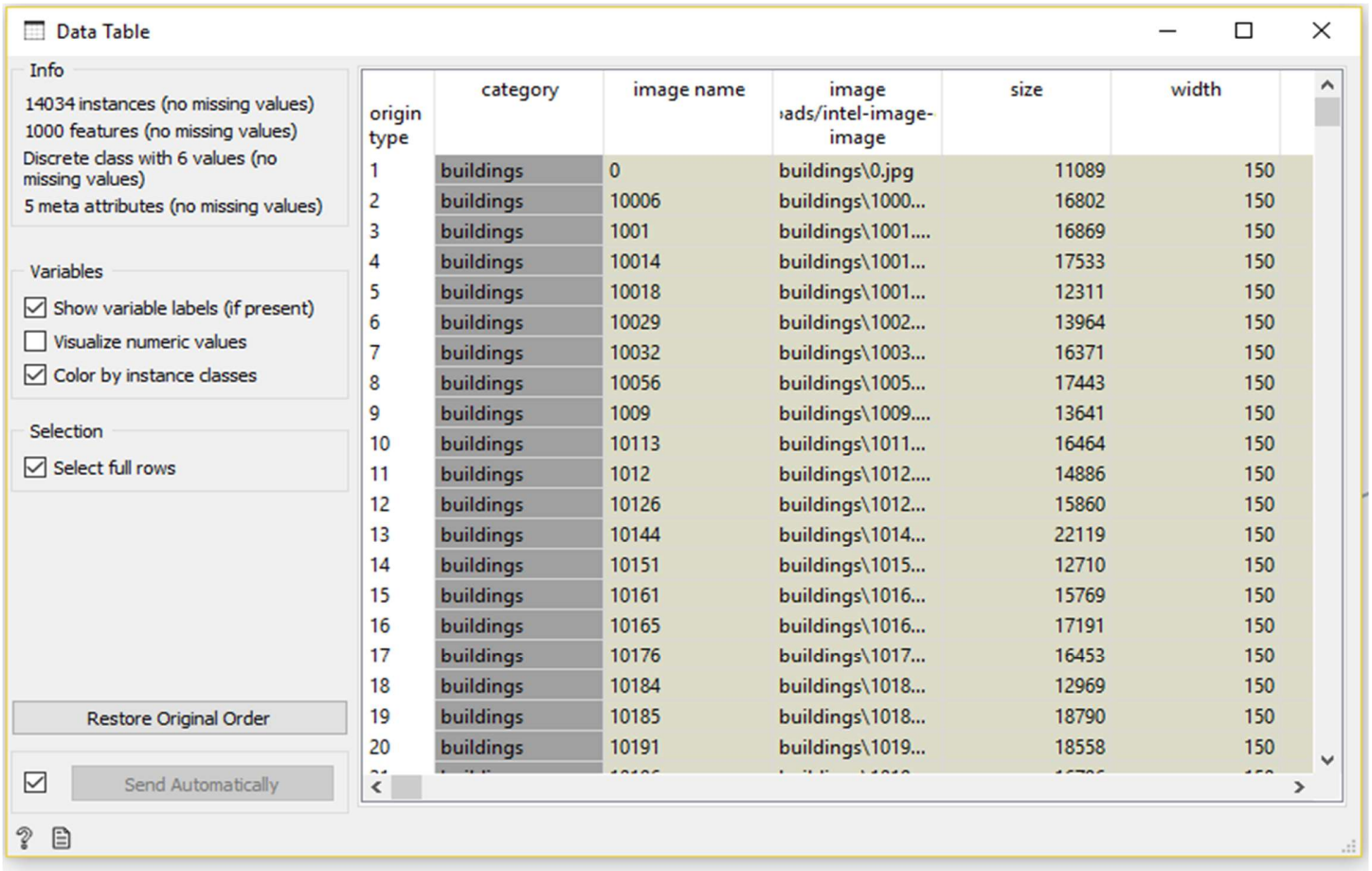
origin type	category	image name	image ads/intel-image-image	size	width
1	buildings	0	buildings\0.jpg	11089	150
2	buildings	10006	buildings\1000...	16802	150
3	buildings	1001	buildings\1001....	16869	150
4	buildings	10014	buildings\1001...	17533	150
5	buildings	10018	buildings\1001...	12311	150
6	buildings	10029	buildings\1002...	13964	150
7	buildings	10032	buildings\1003...	16371	150
8	buildings	10056	buildings\1005...	17443	150
9	buildings	1009	buildings\1009....	13641	150
10	buildings	10113	buildings\1011...	16464	150
11	buildings	1012	buildings\1012....	14886	150
12	buildings	10126	buildings\1012...	15860	150
13	buildings	10144	buildings\1014...	22119	150
14	buildings	10151	buildings\1015...	12710	150
15	buildings	10161	buildings\1016...	15769	150
16	buildings	10165	buildings\1016...	17191	150
17	buildings	10176	buildings\1017...	16453	150
18	buildings	10184	buildings\1018...	12969	150
19	buildings	10185	buildings\1018...	18790	150
20	buildings	10191	buildings\1019...	18558	150

Data Pre-processing:

As the dataset has been given in form of .jpeg format they are first pre-processed into categories and added features using the image embedding widget which is used to pre-process the data as follows :

1. Information on the number of embedded images and images skipped.
2. Settings:
    - *Image attribute*: attribute containing images you wish to embed
    - *Embedder*:
        - SqueezeNet: Small and fast model for image recognition trained on ImageNet.
        - Inception v3: Google's Inception v3 model trained on ImageNet.
        - VGG-16: 16-layer image recognition model trained on ImageNet.
        - VGG-19: 19-layer image recognition model trained on ImageNet.
        - Painters: A model trained to predict painters from artwork images.
        - DeepLoc: A model trained to analyze yeast cell images.
3. Tick the box on the left to start the embedding automatically. Alternatively, click *Apply*. To cancel the embedding, click *Cancel*.
4. Access help.





origin type	category	image name	image ads/intel-image-image	size	width
1	buildings	0	buildings\0.jpg	11089	150
2	buildings	10006	buildings\1000...	16802	150
3	buildings	1001	buildings\1001....	16869	150
4	buildings	10014	buildings\1001...	17533	150
5	buildings	10018	buildings\1001...	12311	150
6	buildings	10029	buildings\1002...	13964	150
7	buildings	10032	buildings\1003...	16371	150
8	buildings	10056	buildings\1005...	17443	150
9	buildings	1009	buildings\1009....	13641	150
10	buildings	10113	buildings\1011...	16464	150
11	buildings	1012	buildings\1012....	14886	150
12	buildings	10126	buildings\1012...	15860	150
13	buildings	10144	buildings\1014...	22119	150
14	buildings	10151	buildings\1015...	12710	150
15	buildings	10161	buildings\1016...	15769	150
16	buildings	10165	buildings\1016...	17191	150
17	buildings	10176	buildings\1017...	16453	150
18	buildings	10184	buildings\1018...	12969	150
19	buildings	10185	buildings\1018...	18790	150
20	buildings	10191	buildings\1019...	18558	150

Model :

We will use Orange to build the simple Artificial Neural Network and random forest.

Neural Network

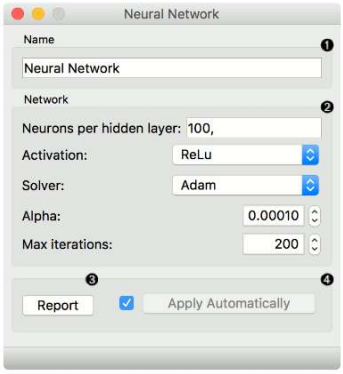
A multi-layer perceptron (MLP) algorithm with backpropagation.

Inputs :

Data: input dataset
Preprocessor: preprocessing method(s)

Outputs :

Learner: multi-layer perceptron learning algorithm
Model: trained model
The Neural Network widget uses sklearn's Multi-layer Perceptron algorithm that can learn non-linear models as well as linear.



1. **A name under which it will appear in other widgets. The default name is "Neural Network".**

2. **Set model parameters:**

Neurons per hidden layer: defined as the ith element represents the number of neurons in the ith hidden layer.

E.g. a neural network with 3 layers can be defined as 2, 3, 2.
Activation function for the hidden layer:

Identity: no-op activation, useful to implement linear bottleneck

Logistic: the logistic sigmoid function

tanh: the hyperbolic tan function

ReLu: the rectified linear unit function

Solver for weight optimization:

L-BFGS-B: an optimizer in the family of quasi-Newton methods

SGD: stochastic gradient descent

Adam: stochastic gradient-based optimizer

Alpha: L2 penalty (regularization term) parameter

Max iterations: maximum number of iterations

Other parameters are set to sklearn's defaults.

3. **Produce a report.**

4. **When the box is ticked (Apply Automatically), the widget will communicate changes automatically. Alternatively, click Apply.**

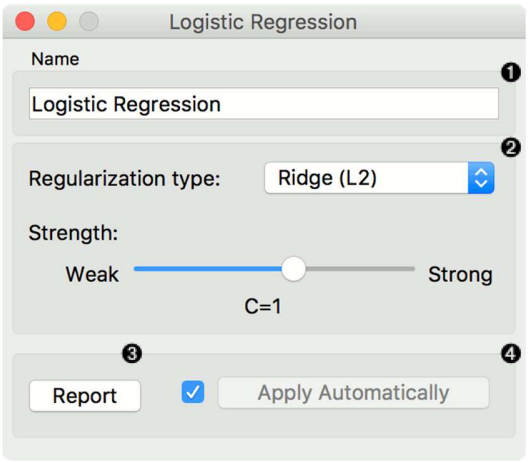
Logistic Regression :

The logistic regression classification algorithm with LASSO (L1) or ridge (L2) regularization.

Inputs

Data: input dataset
Preprocessor: preprocessing method(s)
Outputs

Learner: logistic regression learning algorithm
Model: trained model
Coefficients: logistic regression coefficients
Logistic Regression learns a Logistic Regression model from the data. It only works for classification tasks



1. **A name under which the learner appears in other widgets. The default name is "Logistic Regression".**
2. **Regularization type (either L1 or L2). Set the cost strength (default is C=1).**
3. **Press Apply to commit changes. If Apply Automatically is ticked, changes will be communicated automatically.**

RESULTS

For the testing purpose Image Scene Classification of Multiclass dataset have separate test data with 3000 cases. The following testing accuracy is tested on this data.

Results of Logistic Regression:

Training Accuracy:

Using Logistic Regression, we are able to achieve the 98+/-1 % Train accuracy.

Testing Accuracy:

Using Logistic Regression, we are able to achieve the 97+/-1 % Test accuracy.

Results of Neural Network:

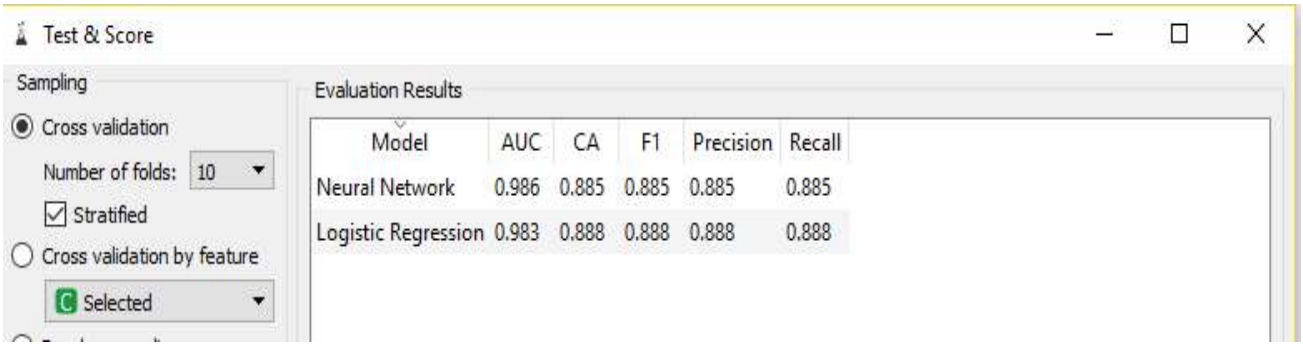
Training Accuracy:

Using Neural Network, we are able to achieve the 98+/-1 % Train accuracy.

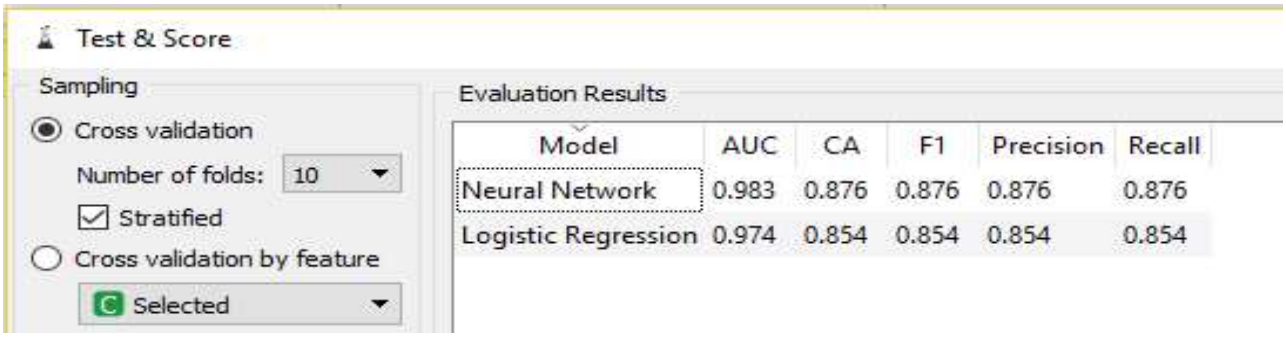
Testing Accuracy:

Using Neural Network, we are able to achieve the 98+/-1 % Test accuracy.

Train Data:



Test Data:

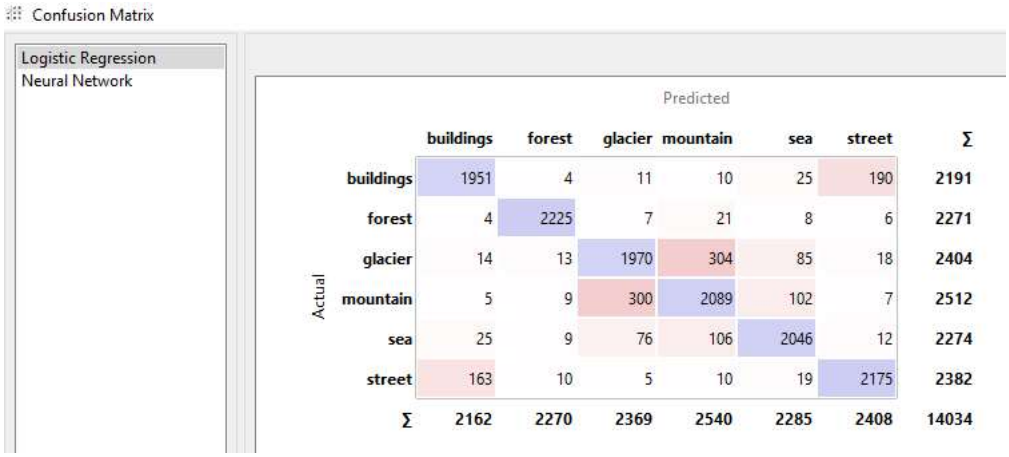


Confusion Matrix:

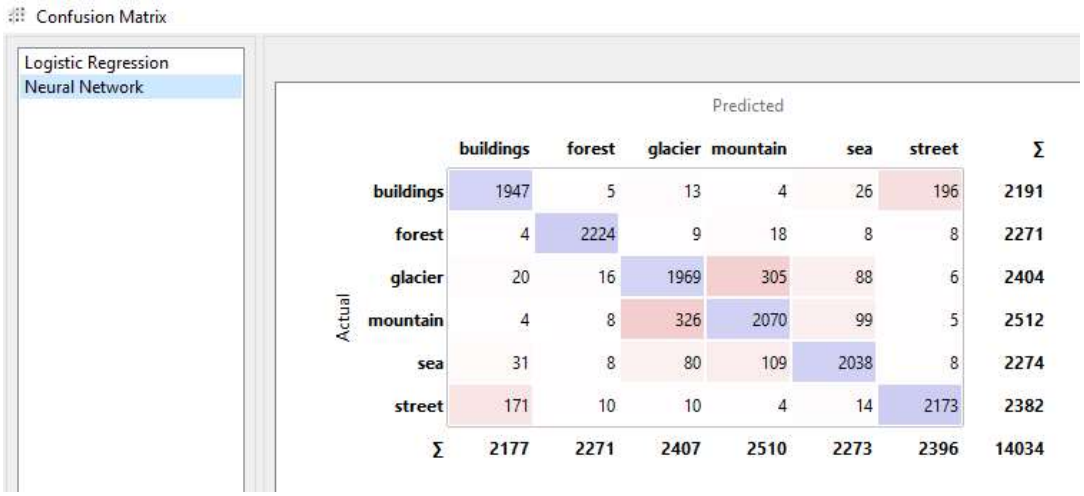
A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. It allows easy identification of confusion between classes e.g. one class is commonly mislabelled as the other. Most performance measures are computed from the confusion matrix.

- **Positive (P): Observation is positive.**
- **Negative (N): Observation is not positive.**
- **True Positive (TP): Observation is positive, and is predicted to be positive.**
- **False Negative (FN): Observation is positive, but is predicted negative.**
- **True Negative (TN): Observation is negative, and is predicted to be negative.**
- **False Positive (FP): Observation is negative, but is predicted positive.**

Train Data:



Actual \ Predicted	buildings	forest	glacier	mountain	sea	street	Σ
buildings	1951	4	11	10	25	190	2191
forest	4	2225	7	21	8	6	2271
glacier	14	13	1970	304	85	18	2404
mountain	5	9	300	2089	102	7	2512
sea	25	9	76	106	2046	12	2274
street	163	10	5	10	19	2175	2382
Σ	2162	2270	2369	2540	2285	2408	14034



Actual \ Predicted	buildings	forest	glacier	mountain	sea	street	Σ
buildings	1947	5	13	4	26	196	2191
forest	4	2224	9	18	8	8	2271
glacier	20	16	1969	305	88	6	2404
mountain	4	8	326	2070	99	5	2512
sea	31	8	80	109	2038	8	2274
street	171	10	10	4	14	2173	2382
Σ	2177	2271	2407	2510	2273	2396	14034

Test Data:

Confusion Matrix

Logistic Regression
Neural Network

Predicted

Actual	buildings	forest	glacier	mountain	sea	street	Σ
buildings	361	1	5	5	10	55	437
forest	0	464	3	3	3	1	474
glacier	4	2	438	83	25	1	553
mountain	3	2	90	412	17	1	525
sea	7	4	25	25	447	2	510
street	46	2	2	3	8	440	501
Σ	421	475	563	531	510	500	3000

Confusion Matrix

Logistic Regression
Neural Network

Predicted

Actual	buildings	forest	glacier	mountain	sea	street	Σ
buildings	381	2	2	0	8	44	437
forest	0	467	3	2	1	1	474
glacier	3	2	444	77	25	2	553
mountain	1	3	69	433	19	0	525
sea	7	3	19	23	455	3	510
street	50	0	0	1	3	447	501
Σ	442	477	537	536	511	497	3000

Finding the misclassified images:

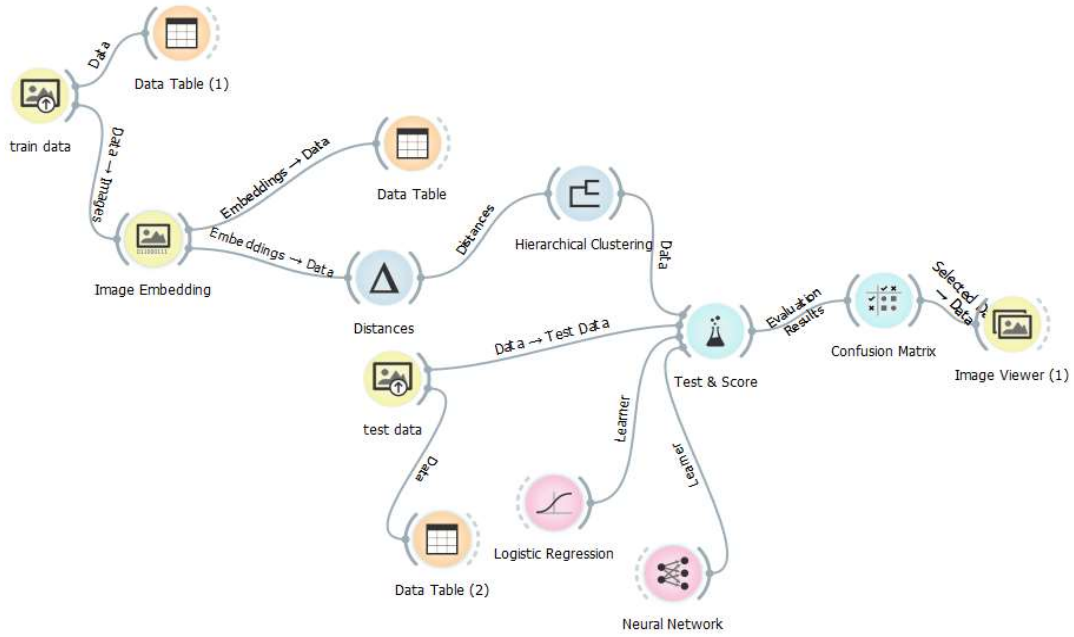
For finding the misclassified points we can use confusion matrix, Orange provide a nice tool to select the misclassified images.

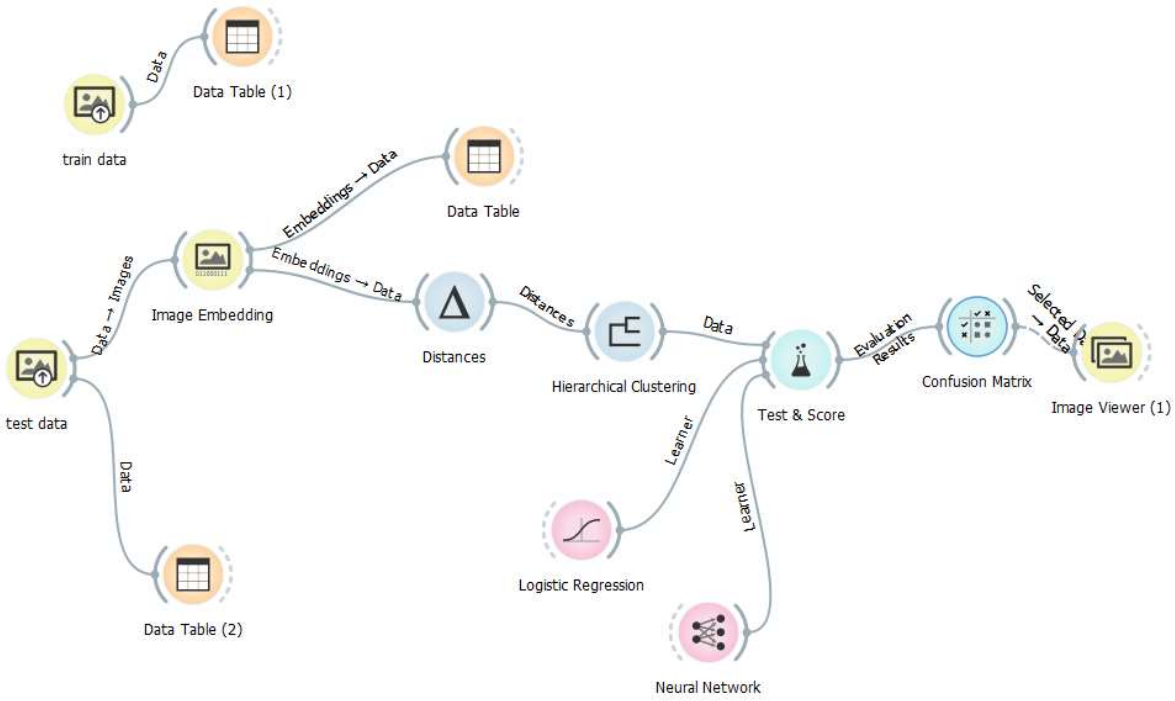
		Predicted					
		buildings	forest	glacier	mountain	sea	street	Σ
Actual	buildings	1947	5	13	4	26	196	2191
	forest	4	2224	9	18	8	8	2271
	glacier	20	16	1969	305	88	6	2404
	mountain	4	8	326	2070	99	5	2512
	sea	31	8	80	109	2038	8	2274
	street	171	10	10	4	14	2173	2382
	Σ	2177	2271	2407	2510	2273	2396	14034

Select the misclassified section in confusion matrix and show them in image viewer as given below:

The above images were wrongly classified as glaciers.

Snapshot


Data
Data Table (1)
train data
Data → Images
Embeddings → Data
Data Table
Embeddings → Data
Image Embedding
Distances
Distances
Hierarchical Clustering
Data
Data → Test Data
Evaluation Results
Confusion Matrix
Selected Data
Image Viewer (1)
Test & Score
test data
Learner
Learner
Data
Logistic Regression
Data Table (2)
Neural Network


Data
Data Table (1)
train data
Embeddings → Data
Data Table
Data → Images
Embeddings → Data
Image Embedding
Distances
Distances
Hierarchical Clustering
Data
Evaluation Results
Confusion Matrix
Selected Data
Image Viewer (1)
Test & Score
test data
Learner
Learner
Data
Logistic Regression
Data Table (2)
Neural Network

CONCLUSION

We conclude that Logistic Regression and Neural Networks can be used as classification algorithms for Image Scene Classification of Multiclass. However, Neural Network has performed better in this case with testing accuracy of 98+/-1 %. We were able to achieve maximum accuracy of 97+/-1 % with Logistic Regression for Image Scene Classification of Multiclass dataset.

Future Work

Our Model's capability is to overcome the obstacles for classification of images according to given categorical folders.

In this project we created a model that could recognize the images that were arranged in a proper manner inside the folders. We aim to develop our model further so that in future it is useful in real world web applications. This work can be extended to recognize and predict those images that are not arranged in classified manner in folders and are instead present in a jumbled manner.

Our future aim is to make a working application of our model using OpenCV and Django.