

Anomaly Detector

(USING ISOLATION FOREST FROM SCIKIT LEARN)

Tarun Sharma | B180574EP | 30 April'21

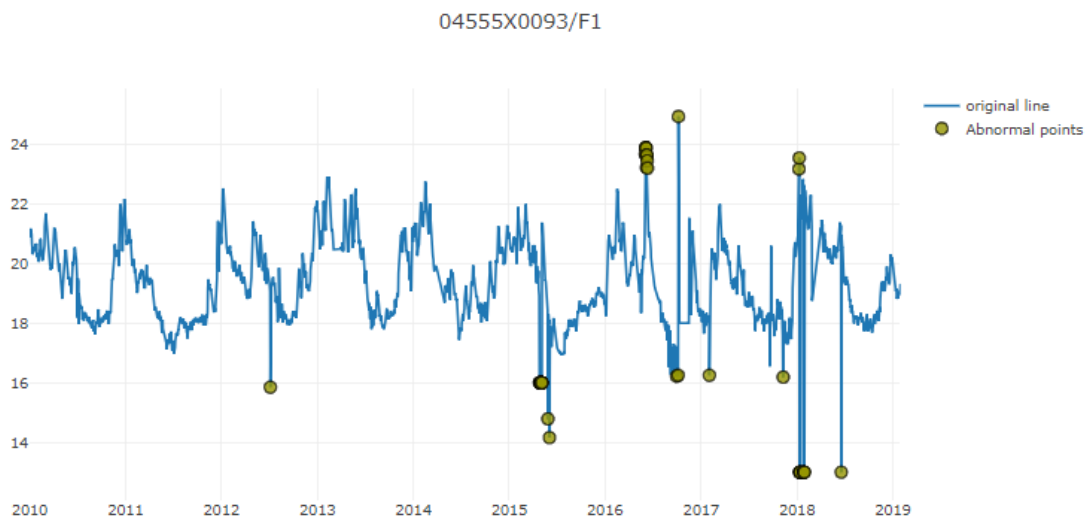
Abstract

This report presents an Anomaly Detector for Forest dataset using Isolation Forest which comes under the package Scikit Learn. Anomalies are events that deviate highly from the pattern or trend. Isolation Forest can be used to find anomaly or outlier data points in any dataset. We have used a subset of 8 scenes dataset which was used first in a [paper by Oliva](#) (click for more info). There are two folders namely Forest and Test which are used as Training and testing datasets. There are (256 x 256 pixels) images of forest in the Forest folder which serves as the training dataset. Test folder consists of scenes ranging from coast and mountains to street and tall buildings. This Anomaly Detector uses color histogram of images as training and testing data.

Introduction

Today's era is the era of Data Science and Data is power. We can train an AI for doing almost any work provided that we have enough training data. But while creating such gigantic datasets, mistakes are bound to happen. We often encounter events which are not ordinary and thus don't follow the pattern. These outliers or anomalies can damage our observations and hence need to be segregated. That's where the **Anomaly Detector** comes into play. Let's take an example of Interferometers which are used to detect Gravitational waves (example: LIGO). It works on principle of path difference, the change in path difference is caused due to the [gravitational waves](#). But there are some anomalies in this data caused due to the geological vibrations from natural disasters or human activities. This data can be and should be removed from the data set.

Unsupervised learning algorithms like K means clustering are being used for clustering the data around centres of high density, from a very long time now. But K means can't establish long term relation and use them for anomaly detection. Thus we need an algorithm which not only can do the clustering but also have memory equivalent to one class LSTM. Isolation forest algorithm is just the algorithm we need for anomaly detection more effectively.



Forest Isolation is generally used for the tabular data but we will be using it on image data sets. We will utilise OpenCV tools to Create a colour Histogram of the image. This colour histogram tells us about the colour distribution throughout the image. We will be using this colour histogram as the input data for our isolation forest. The same colour histogram will be used to quantify the test images.

We will be using the argument parser package for Pointing the location of input data set, output model location and the address of the testing image. Argument parser is used to avoid the changing of code whenever we want to provide a new address for training and testing data sets.

Isolation Forest

Isolation forest is a function which comes under the Scikit learn package. Isolation forest is used for finding the outliers. It is one of the unsupervised learning algorithms. When fit on a dataset, it returns an anomaly score for each of the samples.

Syntax:

```
sklearn.ensemble.IsolationForest(*, n_estimators=100,
max_samples='auto', contamination='auto', max_features=1.0,
bootstrap=False, n_jobs=None, random_state=None, verbose=0,
warm_start=False)
```

The IsolationForest ‘isolates’ the observations by randomly selecting a feature first after that it randomly selects a split value between the maximum and minimum values of the selected feature. Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node. This path length is averaged over a forest of such random trees, which is a measure of normality and our decision function. This random partitioning results in a shorter path for the Anomaly so we can say that Forest of random trees with shorter path are anomalies.

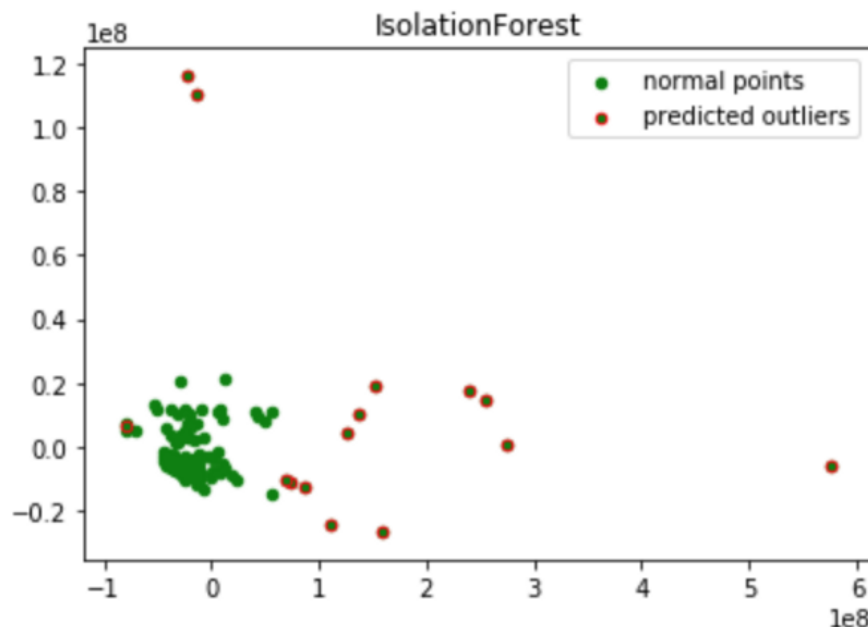
Parameters:

n_estimators: number of base estimators

contamination: the proportion of anomalies with respect to normal data points

random_state: randomness in selection of feature and splitting values across each branch and tree.

It can be implemented on tabular data very easily and the output can be visualised using matplotlib plots. given below is an example of the isolation forest algorithm used on the Revenue Matrix data set.



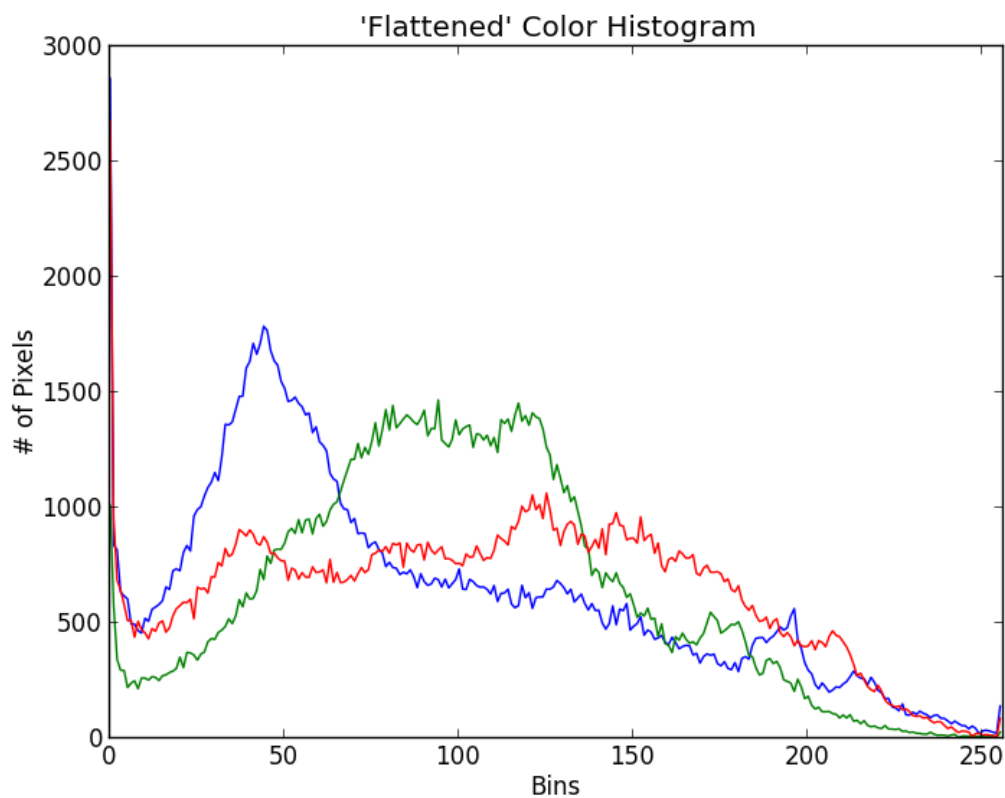
Packages and Algorithms:

OpenCV:

It is an open source software library which is mostly used for computer vision and machine learning application. We utilised it for generating the color histogram of input and testing images.

Color Histogram:

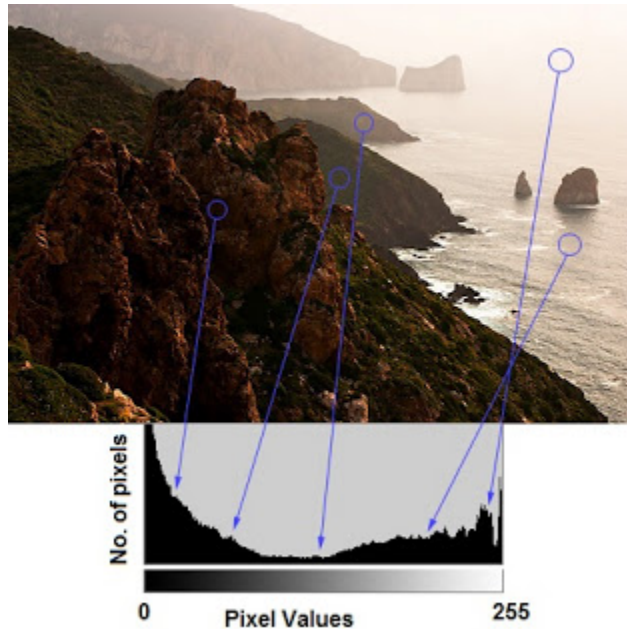
Color histogram quantifies the color distribution throughout the image. This data is later used for training our model using Isolation Forest Algorithm.



We convert the image from RGB space to HSV space and then capture its color histogram using OpenCV function `calcHist`.

Syntax:

```
cv2.calcHist(images, channels, mask, histSize, ranges[,  
hist[, accumulate]])
```

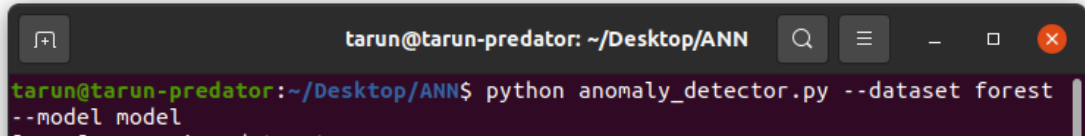


Argument Parser:

Argument parser is a package which enables us to supply command line arguments. This helps us to change the values of certain parameters without actually editing the code. Here we used it to feed the dataset location of input images, output location where our model is to be created and test image address.

Syntax:

```
argparse.ArgumentParser().add_argument("-d",  
"--dataset", required=True, help="path to  
the training dataset images")
```



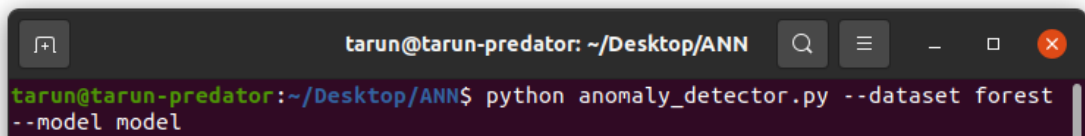
```
tarun@tarun-predator: ~/Desktop/ANN
tarun@tarun-predator:~/Desktop/ANN$ python anomaly_detector.py --dataset forest
--model model
```

The parameters are supplied in the command line like mentioned in the previous image. The second address is of the output model.

Project Setup:

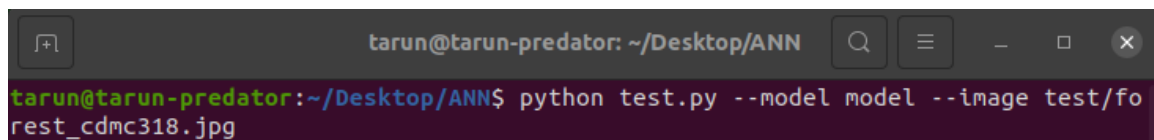
The project is divided into 5 components namely

1. helper_fcn.py - This is a python file that contains two important helper functions named as **quantify_image** and **load_dataset**, which are used in the training and testing scripts.
2. anomaly_detector.py - This is a python file which is model training script. It should be run through the command line with location of training dataset and output model location as the arguments.



```
tarun@tarun-predator: ~/Desktop/ANN
tarun@tarun-predator:~/Desktop/ANN$ python anomaly_detector.py --dataset forest
--model model
```

3. test.py - This is a python file which is the testing script of our project. Run this using the command line along with the address of the image you want to test as the argument.



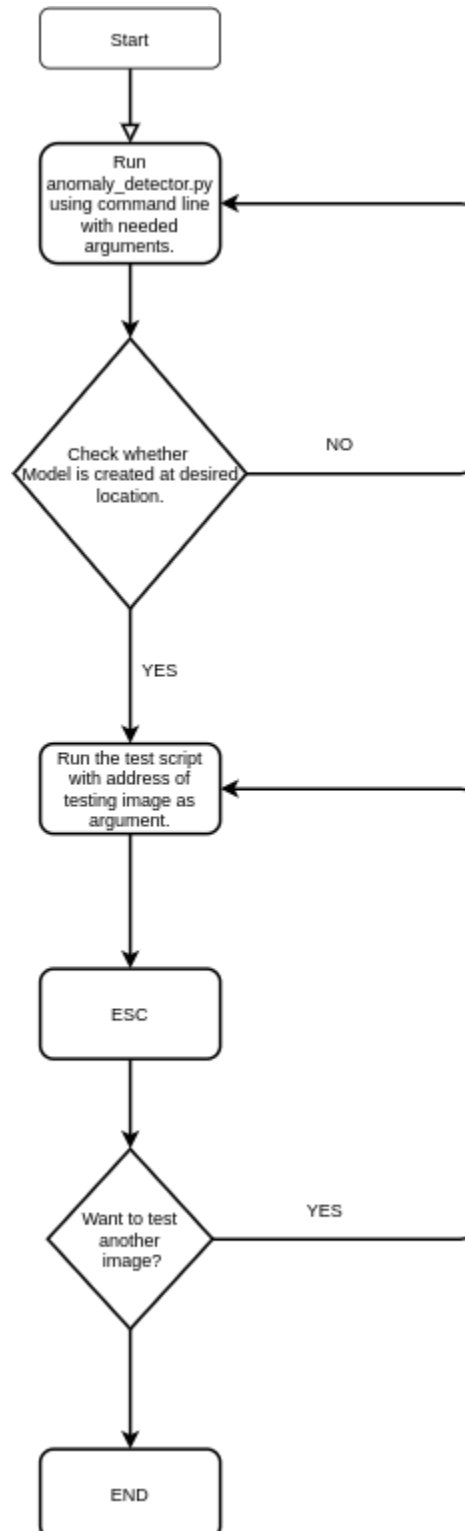
```
tarun@tarun-predator: ~/Desktop/ANN
tarun@tarun-predator:~/Desktop/ANN$ python test.py --model model --image test/forest_cdm318.jpg
```

4. Forest - It's a folder that contains (256 x 256 pixels) images of forest which serves as our training dataset. This is a subset of [8 scenes dataset](#).
5. Test - It's also a folder that contains images with the same resolution as the training dataset. It has pictures of different sceneries which are used for testing our model. It's also a subset of [8 scenes dataset](#).

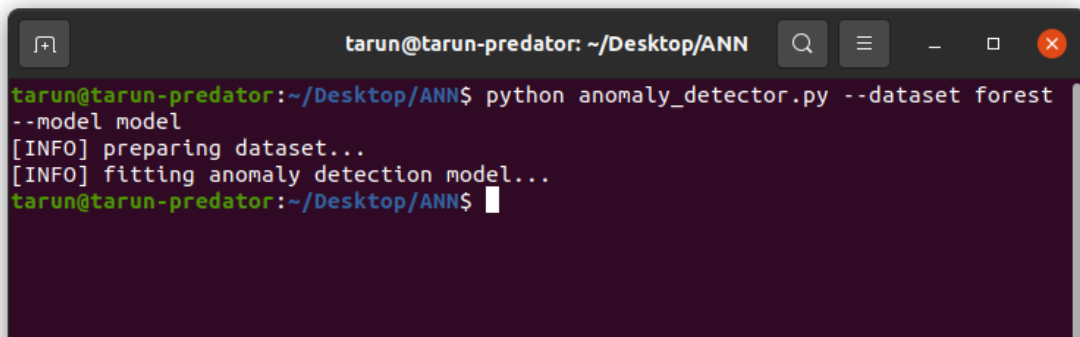
Once we run the anomaly_detector.py with needed arguments it creates a model in pickle format.

Working:

The anomaly detector works in the following steps.

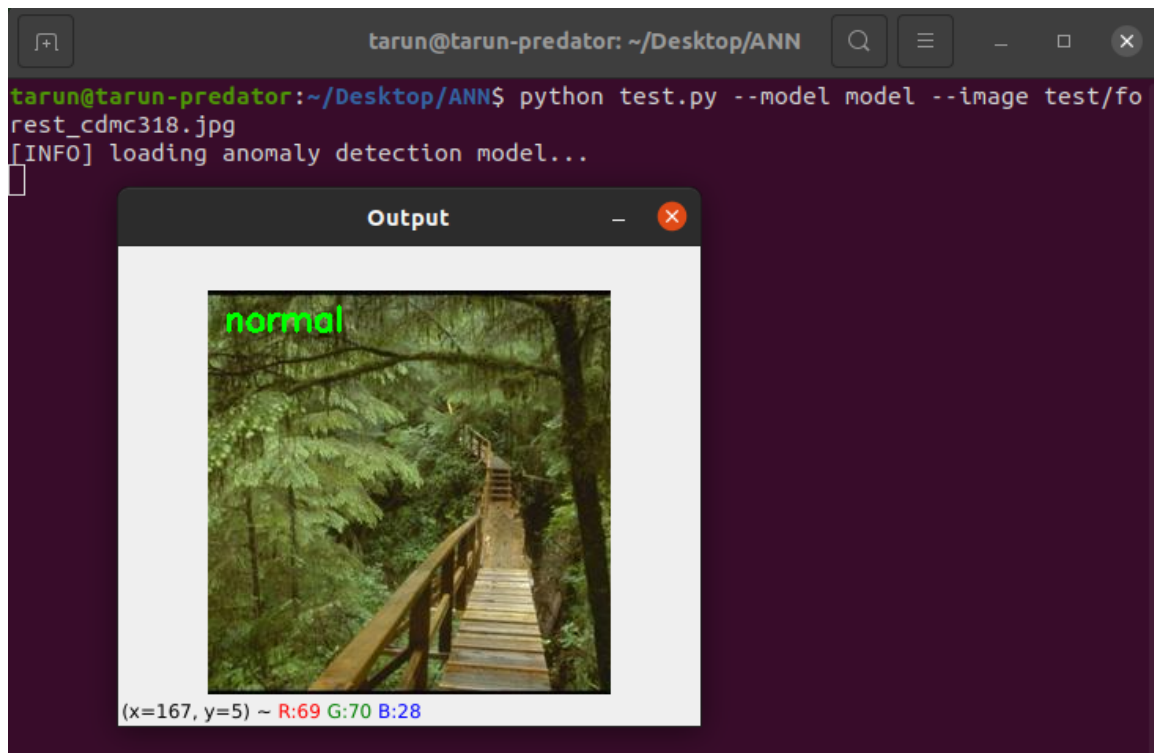


Once we run anomaly_detector.py we will get command line output like this.



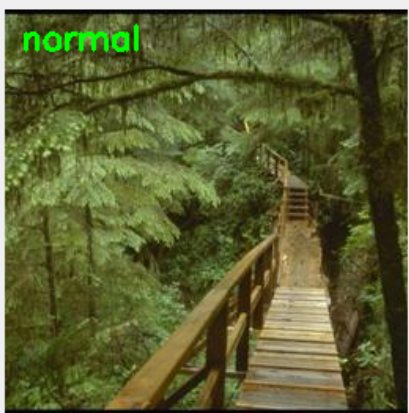
```
tarun@tarun-predator: ~/Desktop/ANN
tarun@tarun-predator:~/Desktop/ANN$ python anomaly_detector.py --dataset forest
--model model
[INFO] preparing dataset...
[INFO] fitting anomaly detection model...
tarun@tarun-predator:~/Desktop/ANN$
```

After this we run the testing script which tells us whether the given image is **normal** or **anomaly**.



```
tarun@tarun-predator:~/Desktop/ANN$ python test.py --model model --image test/forest_cdm318.jpg
[INFO] loading anomaly detection model...
```

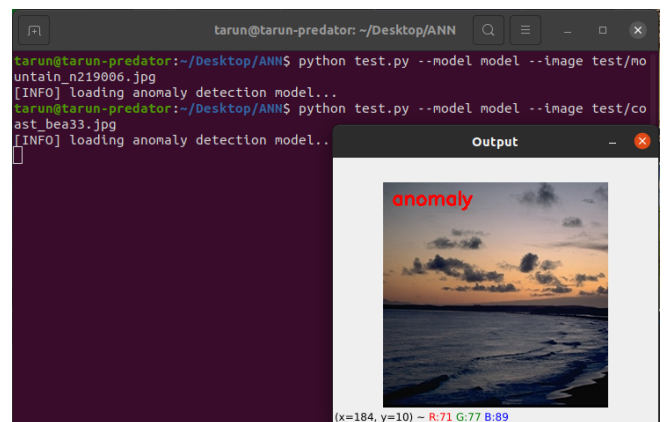
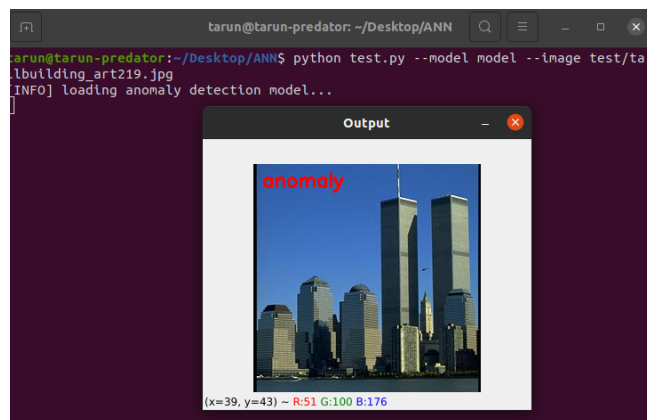
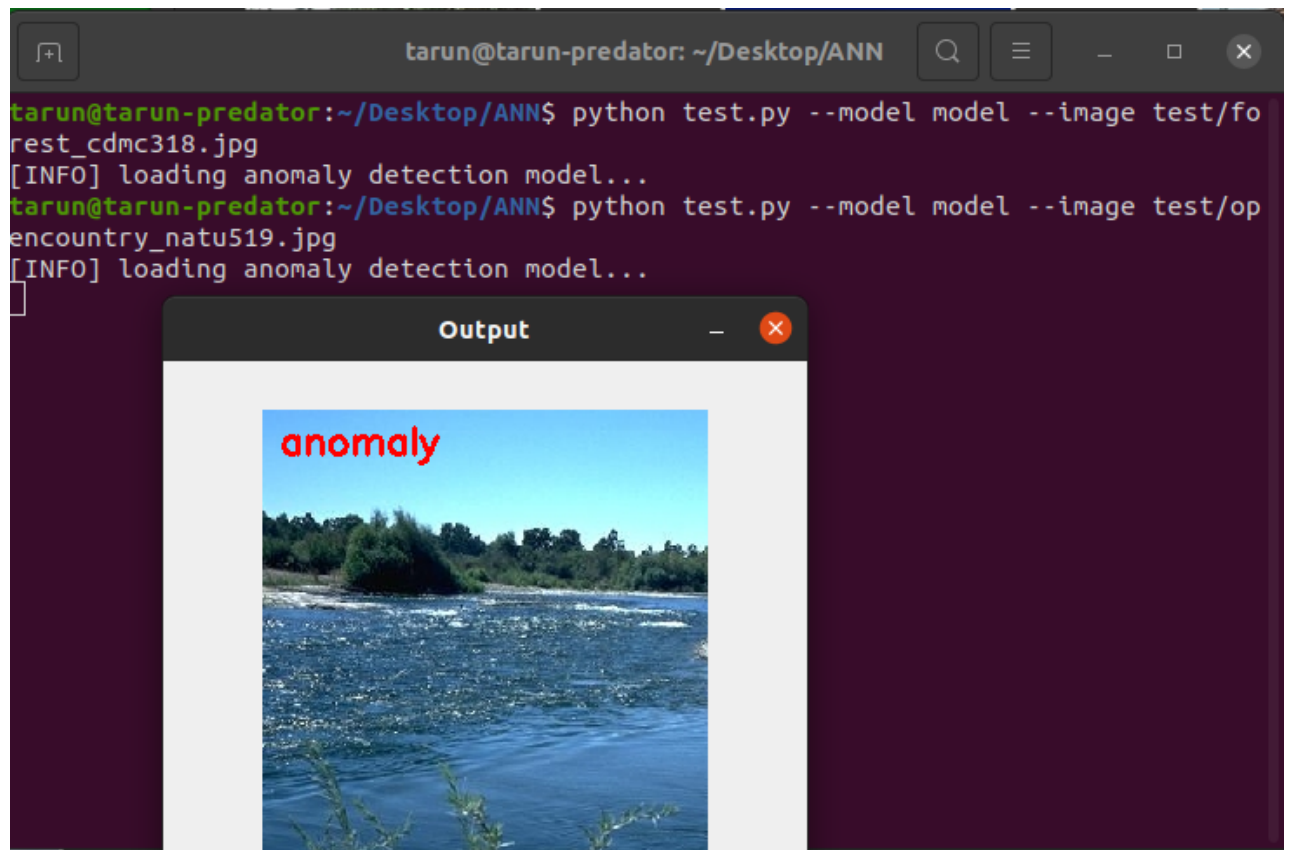
Output



(x=167, y=5) ~ R:69 G:70 B:28

As the testing image is of forest so we will get **normal** as output.

Now when we test it with any other scene except that of forest, it must give output as an **anomaly**.



Conclusions:

The Anomaly detector has a wide range of applications in this era of data science. Though we used it for anomaly detection in a simple forest dataset but it can be used to find outliers or anomalies in relatively large datasets like dataset of online trading applications like FOREX or Olymp trade and we can quantify the effect of events like CoVid pandemic on the nature of curves. As discussed earlier it also has applications in big projects like LIGO(Gravitational Wave detection) or CERN (Particle Accelerator) etc. It's up to us to find its other uses and keep updating such algorithms.

References:

1. Paper by oliva using 8 scenes dataset
<https://people.csail.mit.edu/torralba/code/spatialenvelope/>
2. Anomaly detection using Isolation forest algorithm
<https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2>
3. Isolation forest algorithm
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html>
4. OpenCV <https://docs.opencv.org/master/modules.html>
5. OpenCV DNN module
https://docs.opencv.org/master/d6/dof/group_dnn.html
6. Color Histogram
https://docs.opencv.org/master/d1/db7/tutorial_py_histogram_begins.html
7. Argument parser
<https://www.tutorialspoint.com/argument-parsing-in-python>