Weather Image Recognition

**Ajay Shrikrishna Naik, Umutbek Abdimanan Uulu, Serkan Hiziroglu,**

**Hema Deepika Thanigai Arasu, Vu Pham**   
    
DS620 –Machine Learning & Deep Learning, MSCS, City University of Seattle

[naikajayshrikrishna@cityuniversity.edu](mailto:naikajayshrikrishna@cityuniversity.edu), [abdimananuuluumutbe@cityuniversity.edu,](mailto:abdimananuuluumutbe@cityuniversity.edu,) [hizirogluserkan@cityuniversity.edu](mailto:hizirogluserkan@cityuniversity.edu), [thanigaiarasuhemade@cityuniversity.edu](mailto:thanigaiarasuhemade@cityuniversity.edu), phamvu@cityuniversity.edu

**Abstract**

Artificial Intelligence has been growing a lot in recent years. Machine Learning and Deep Learning is a major reason for its success. As DS620 students, we have decided to work on image classification to deepen our knowledge about classification and create a project related to this subject.

Extracting information related to weather and visual conditions at a given time and space is indispensable for scene awareness, which strongly impacts our behaviors, from simply walking in a city to riding a bike, driving a car, or autonomous drive assistance. Despite the significance of this subject, it is still not been fully addressed by machine intelligence relying on deep learning and computer vision to detect the multi-labels of weather and visual conditions with a unified method that can be easily used for practice.

In this project, we design to train three model to predict weather is good or bad. Three models will be train, first model will be train using Support Vector Classifier (SVC), second model will be train using Decision Tree and third model will be train using Convolutional Neural Network (CNN).

**Keywords**

Artificial intelligence, Machine learning, Deep learning, image detection, classification, weather image recognition

**Abbreviation:**

|  |  |
| --- | --- |
| SVC | Support Vector Classifier |
| CNN | Convolutional Neural Network |
| MSE | Mean-Squared Error |
| RMSE | Root-Mean-Square Deviation |

**Background and Problem Statement**

In their study, Lazo et al. (2009) found that United States adults looked at weather forecasts nearly 300 billion times a year. Reliable forecasts play a crucial role in the prediction of hazardous weather, such as flash floods, hurricanes, and blizzards, 9 to 10 days before their occurrence. The history of weather forecasting points to 1922 when Britain’s Lewis Fry Richardson constructed a systematic process based on math to help predict weather (Schultz, et al., 2021). A century later, the modern weather forecasting practice is based on the kind of complex computations initially imagined by Richardson. Scientists have shifted to the use of artificial intelligence and machine learning tools to predict the weather.

Although the weather and climate research community is aware of the potential of model deep learning technologies to solve various data analysis, numerical model, and post-processing problems in the area of numerical weather prediction (NWP), there is still reservation about deep learning. While deep learning weather prediction (DLWP) models do not currently rival NWP, the machine learning approach shows promise. There is still debate about the most accurate and reliable machine learning model that could be used to predict whether. This project seeks to train three models using Support Vector Classifier (SVC), Decision Tree, and Convolutional Neural Network (CNN) to determine the most accurate in predicting whether weather is good or bad.

**Data preparation for SVC and Decision Tree:**

To design this process, we needed to find a dataset that can contain a variety of weather images. We wanted various and diverse images to make our model work in different situations. After researching datasets in Kaggle, we have found one that we can work with. Our dataset has 6862 weather images in 11 different weather categories. However, since we would have only 2 labels, we decided that 400 images were enough. For preparation, we created two separate groups. One is for good weather, and one is for bad weather. The good weather group included 200 images which have dew and rainbow pictures. For the bad weather group, we have included 200 images from snow, rain, fog, and lightning weather. We paid attention to keeping image sizes the same in both groups to prevent overfitting. As we have decided to train two models SVC and CNN, per the model we have prepared the datasets.

**SVC Dataset**

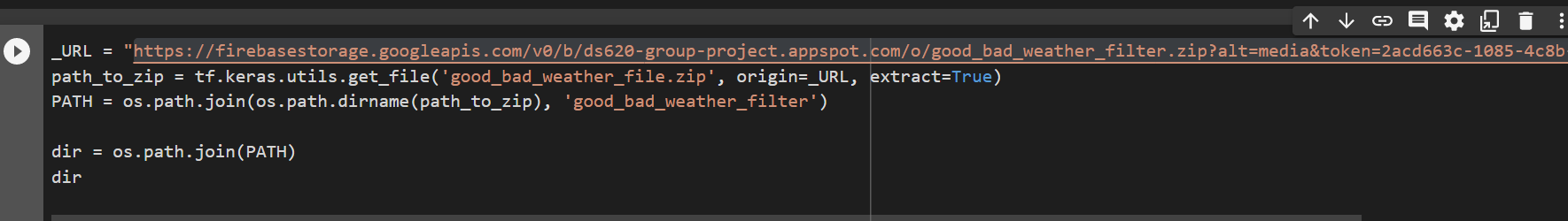
To prepare for the SVC dataset we have currently we have gathered some random 200 good images of dew, sunshine, and rainbow and 200 bad images of snow, rain, and fog. We have uploaded all these images to Firebase Google Storage as a zip file.

Data Uploaded URL:

<https://firebasestorage.googleapis.com/v0/b/ds620-group-project.appspot.com/o/good_bad_weather_filter.zip?alt=media&token=2acd663c-1085-4c8b-8d3d-2438890d7aba>

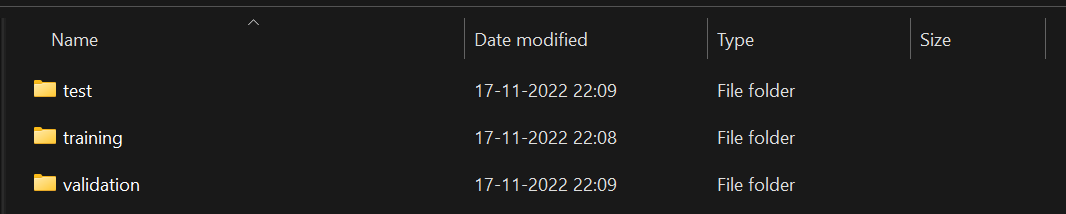
Our data initialization code will fetch the images from this URL and unzip it in *project/local/temp* directory.

**Code Snippet**



**CNN Dataset**

For CNN we are using the same dataset. The data initialization part will be the same but we have distributed the dataset into three folders training, validation, and testing. All these folders will be having two classes: good and bad.



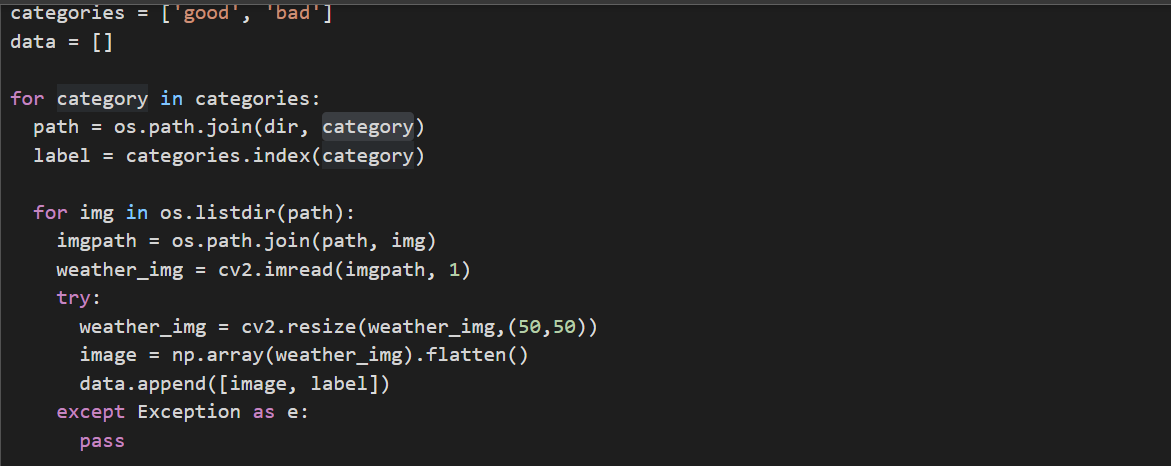
We have distributed the data 70% as training, 15 % as validation, and 15% as a test.

**Preparing the model**

To access the dataset, we used Firebase Cloud Storage and Google Cloud API. Our datasets contain two folders, one folder is good which contains all good weather images, and bad which contains all bad weather images.

First, we initialized the array which will contain an image array with labels, and reshuffled the data.

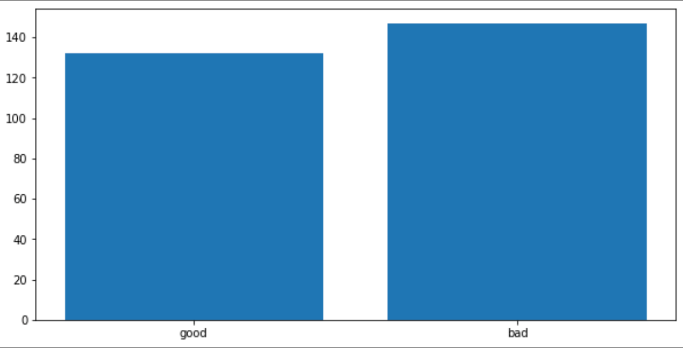
Code Snippet:



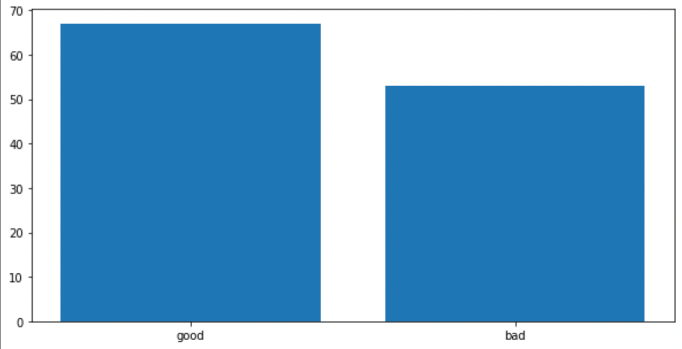
Using *train\_test\_split* function, we have split the data into training and test sets. We have reserved 70% of the images for training and 30% of them for test.

*Some Visualization of training and test data:*

**Training Data**



**Test Data**

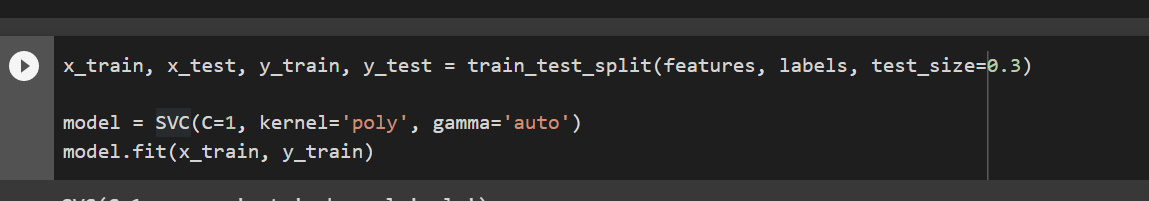


**SVC Model Initialization**

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers’ detection.

SVC works by mapping data points to a high-dimensional space and then finding the optimal hyperplane that divides the data into two classes.

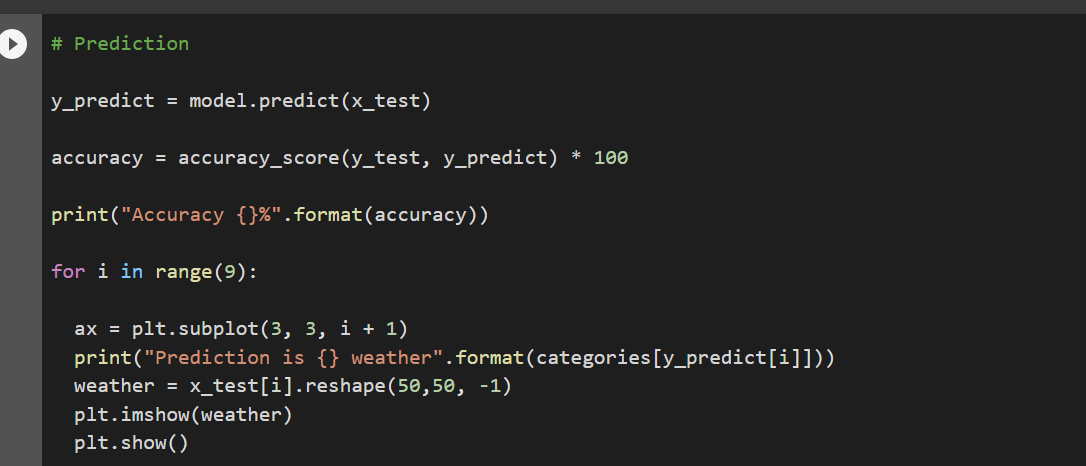
We have initialized the SVC model with polynomial kernel with 1 regularization parameter and gamma value as *auto*. After initialization of SVC model, we have trained the model using the fit method.



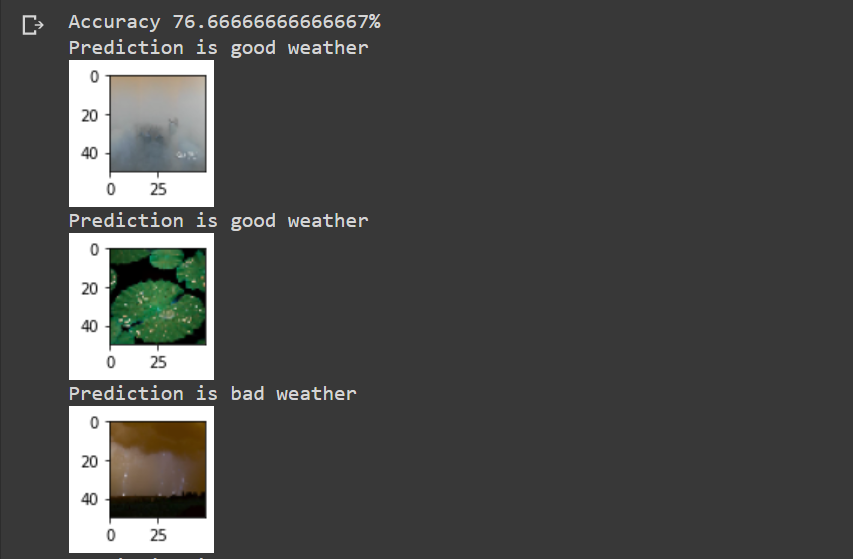
**Evaluating the model**

After training our model, we needed to test the mode’s accuracy. To test the model, we have used test dataset, called the predict method and stored the output. We also printed same output vs the actual output. We got 76% of model accuracy.

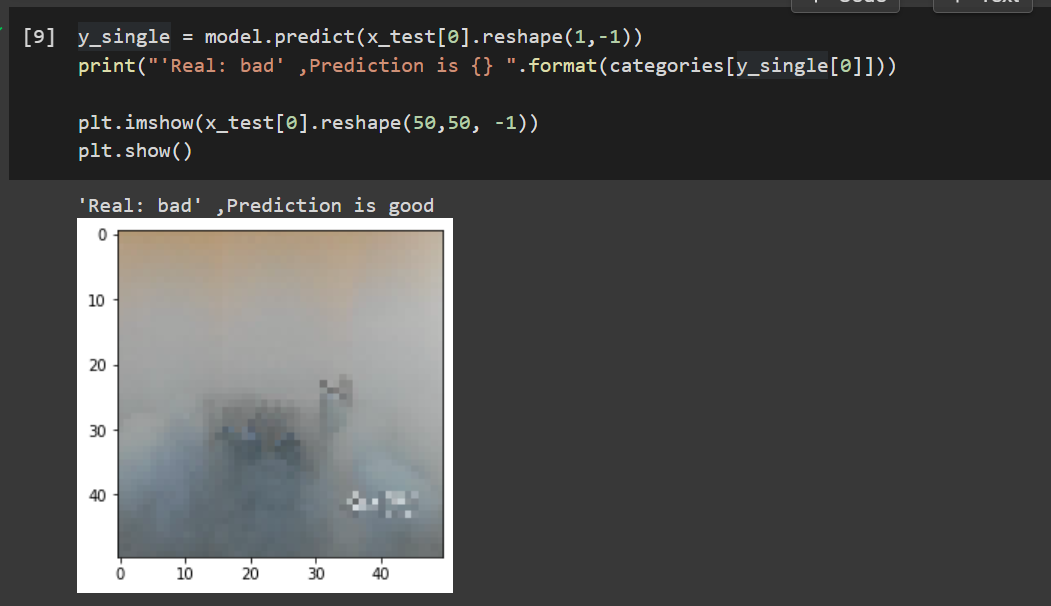
**Code Snippet**



**Outputs**

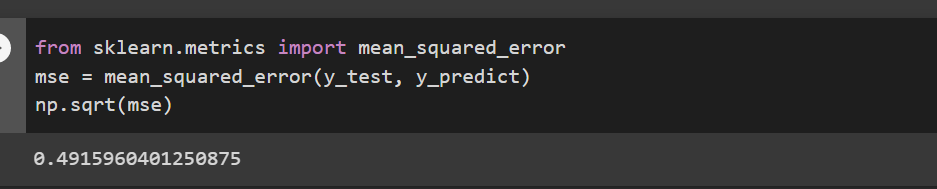
****

**Predicted vs actual output results**





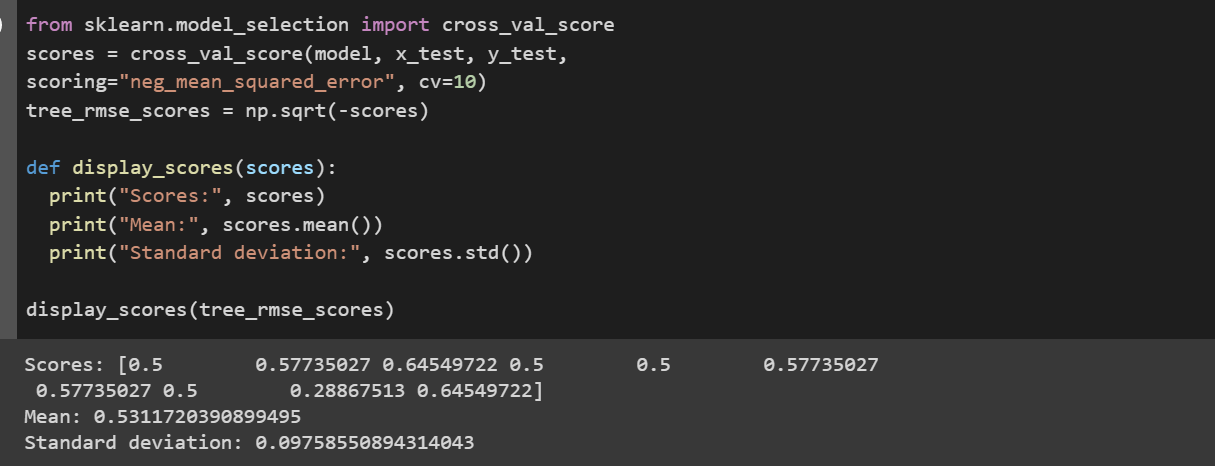
In Statistics, Mean Squared Error (MSE) is defined as Mean or Average of the square of the difference between actual and estimated values.



The MSE is quite high with a 49% score. We need to improve the accuracy of the model.

Another alternative is to use Scikit-Learn’s K-Fold Cross-Validation feature. The following code randomly splits the training set into 10 distinct subsets called folds, then it trains and evaluates the model 10 times, picking a different fold for evaluation every time and training on the other 9 folds. The result is an array containing

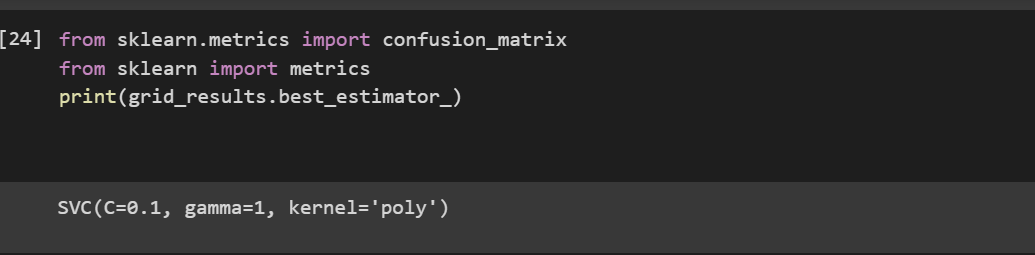
the 10 evaluation scores:



**Finding the best parameters for the model**

GridSearch is an exhaustive search over specified parameter values for an estimator.

Code Snippet:



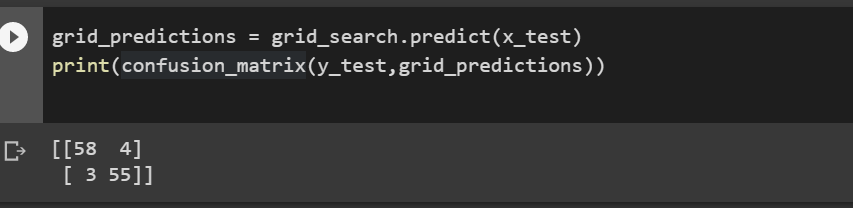
From GridSearch, we found the best parameters for SVC: Regularization parameter is *0.1*, gamma is *1* and kernel is *poly*.

We also find out some metrics from Grid Search:

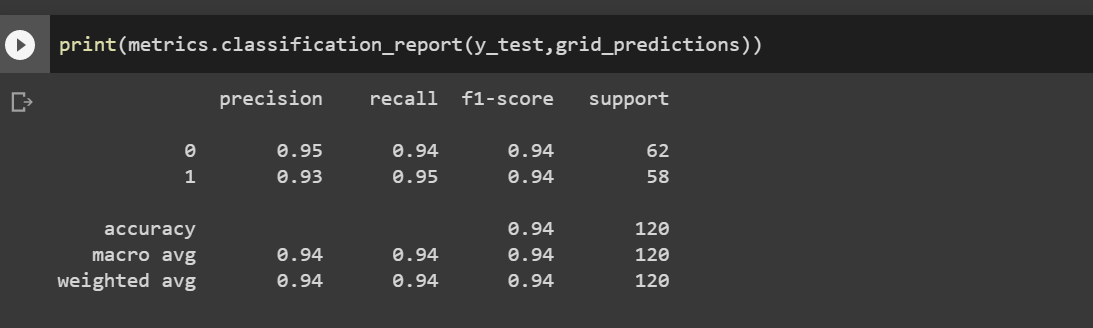
**Confusion Matrix**

Confusion Matrix is a method to evaluate the accuracy of a classification.

Results:

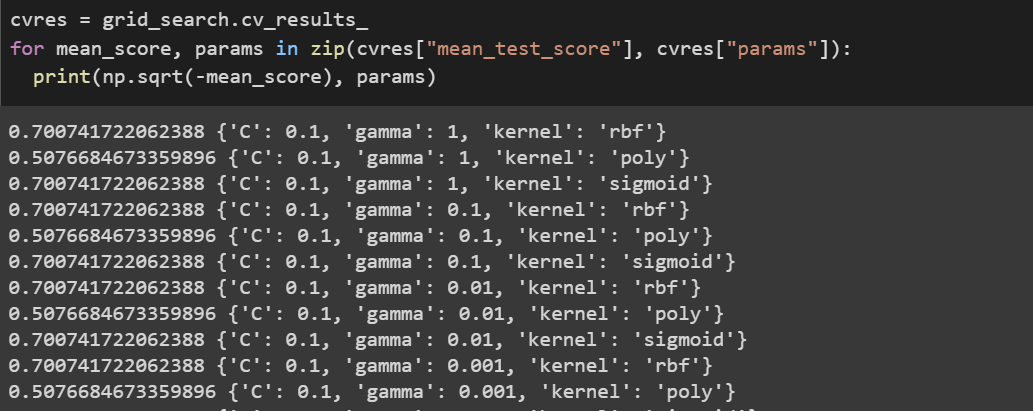


**Classification Report Results**



As per the classification\_report metrics, the precision and recall are close to 94% for both good and bad classes with an improved accuracy score, which is 78%.

The GridSearch also gave us RMSE for all the metrics which we have given as a parameter.



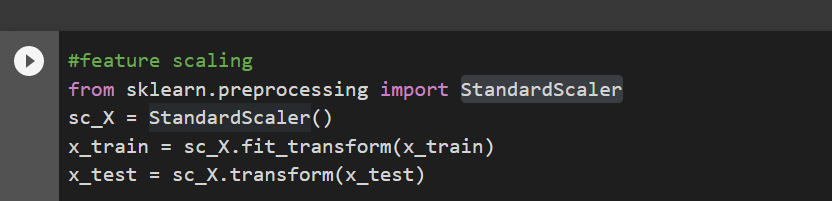
From the given results the RMSE is 0.5076684673359896 which is quite less compared to previous model.

**Decision Tree Model**

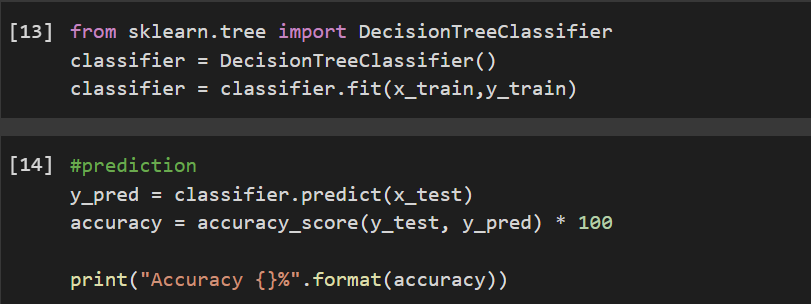
A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Data preparation and splitting the code is similar to SVC.

To Standardize the data, we have used StandardScaler to standardize features by removing the mean and scaling to unit variance.



Training and testing the decision tree is similar with SVC.



Without doing fine-tune of the model, accuracy is 68.33 %.

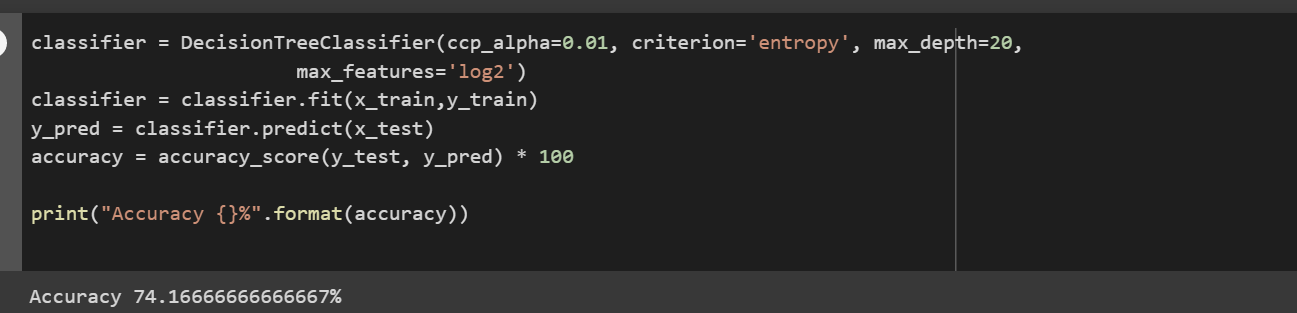
**Fine Tuning Decision Tree**

Like SVC model tuning we are using GridSearchCV to find the best params.

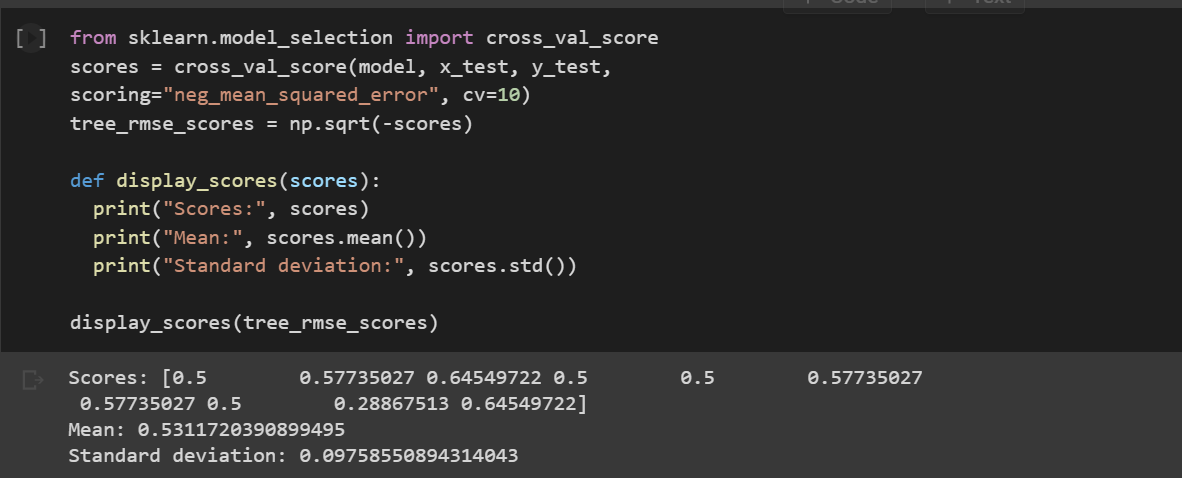


After finding the best params model accuracy is increased to 75%.

Mean Square error was 0.57%. After tuning it, is 0.5%.



Scores, mean and deviation has been reduced also.



**Convolutional Neural Network Model**

CNN or the convolutional neural network (CNN) is a class of deep learning neural networks. In short, think of CNN as a machine learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other.

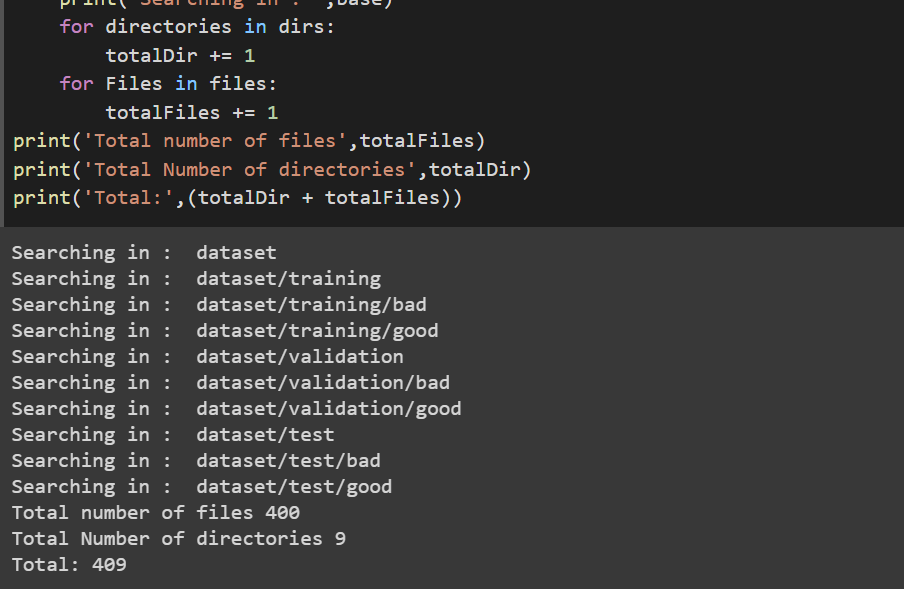
CNN works by extracting features from the images. Any CNN consists of the following:

* The input layer which is a grayscale or color image
* The Output layer which is a binary or multi-class labels
* Hidden layers consisting of convolution layers, ReLU (rectified linear unit) layers, the pooling layers, and a fully connected Neural Network.

As explained in the data preparation parts we will use uploaded data as datasets.

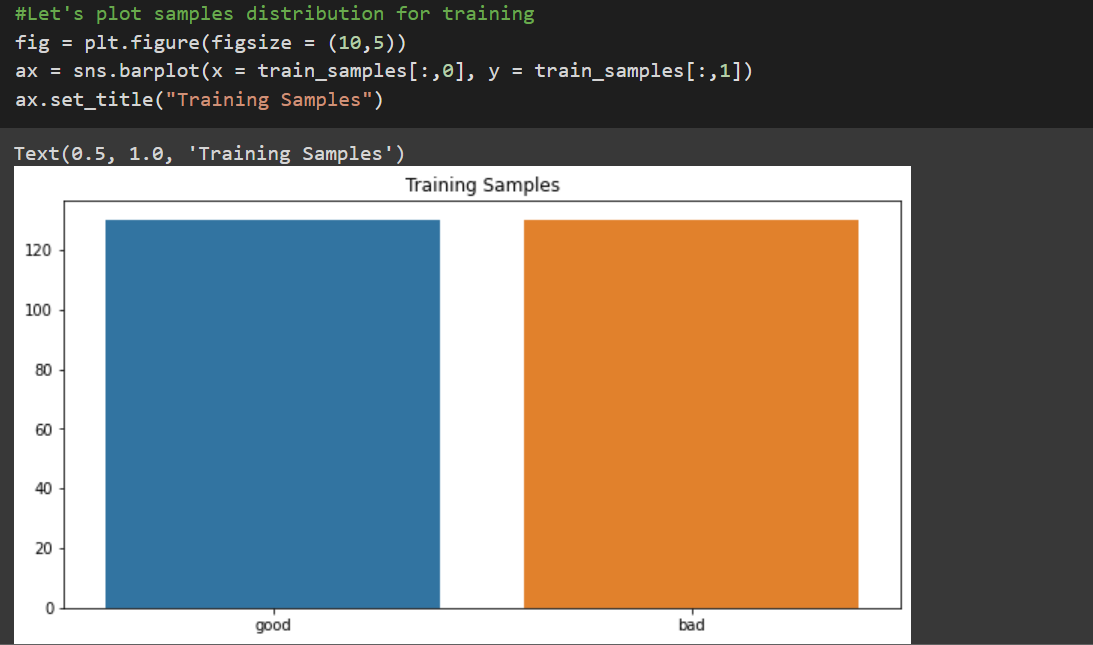
**Exploring the dataset (EDS)**

As a good practice, EDS is important to understand the data first and try to gather as many insights from it. EDA is all about making sense of the data in hand. So, let’s start an exploration of the available data. The dataset structure is composed as follows:

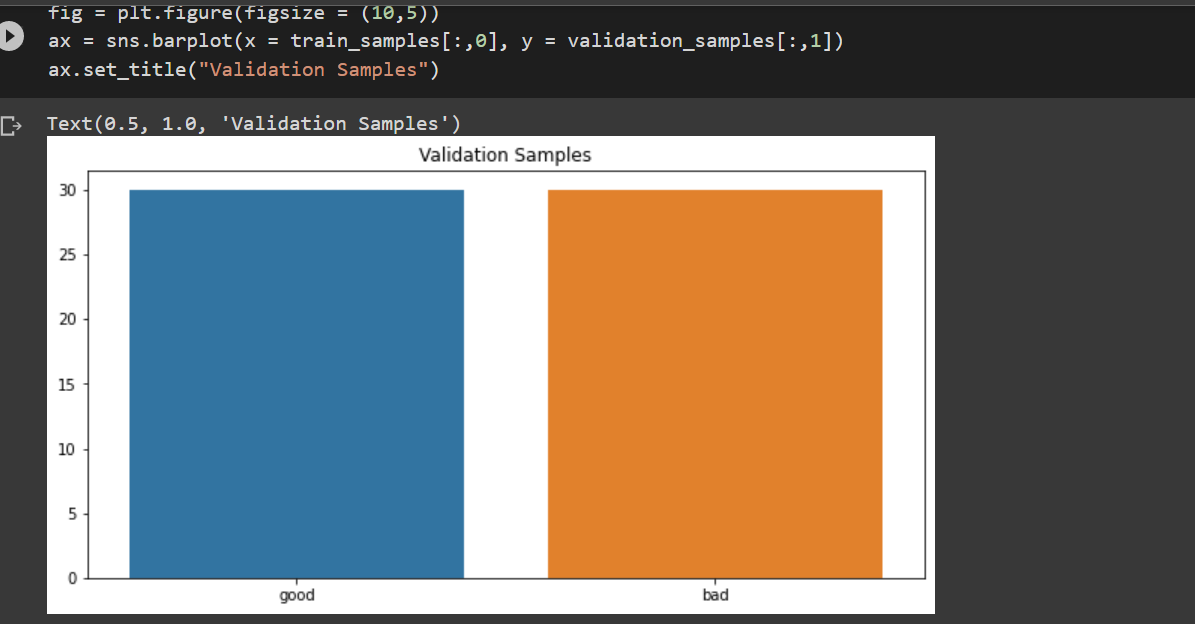
****

Let’s look at distribution of samples of the images by each category. In the following figure, we can see that for the training dataset, Validation, and Test dataset the category of good and bad number of samples.

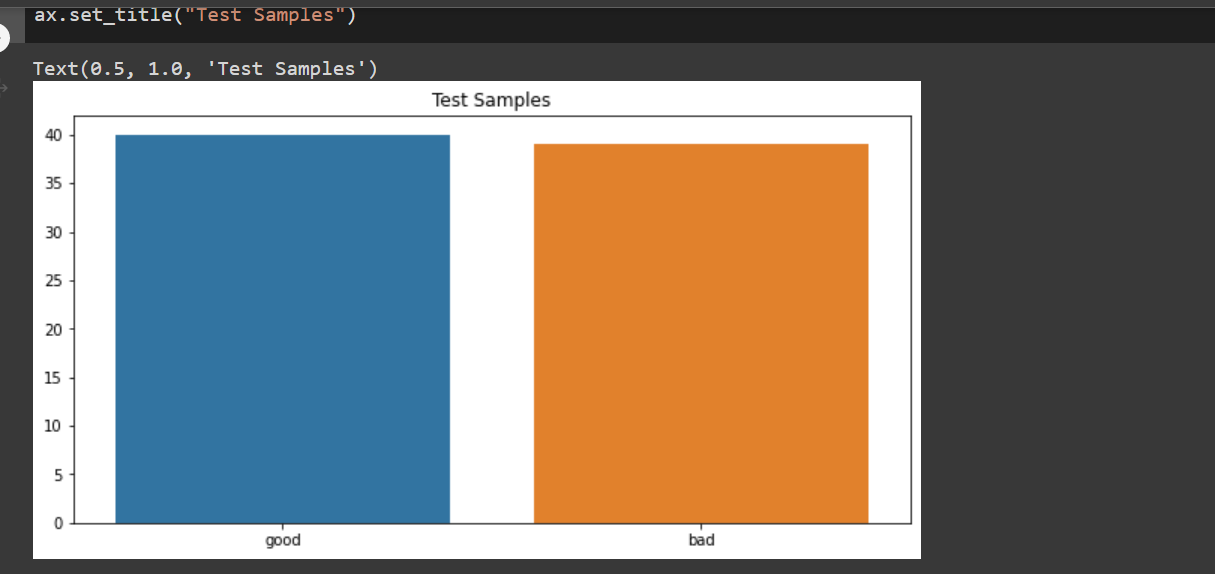
**Training Data**



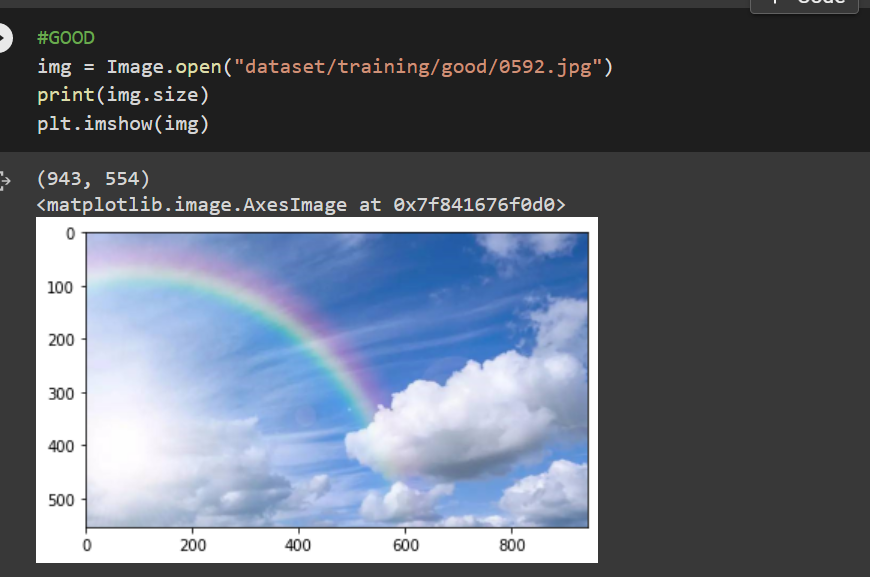
**Validation Data**

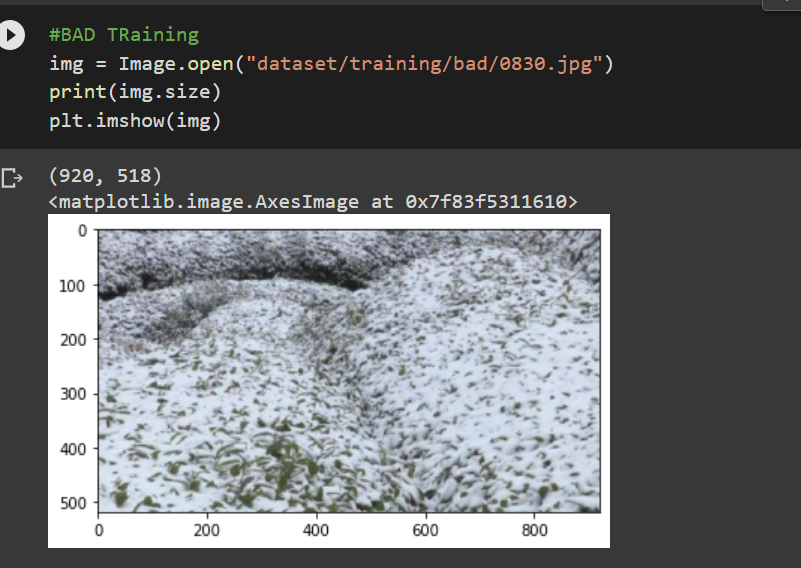


**Test Data**



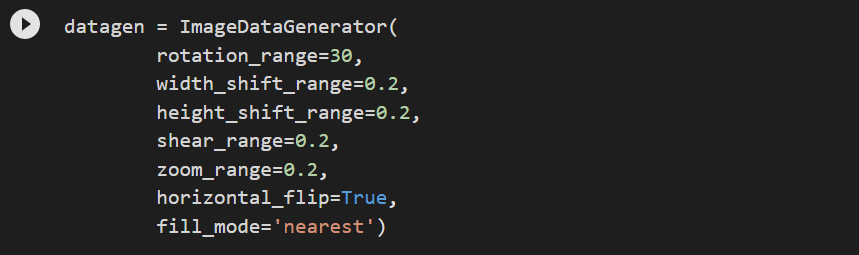
The images shown have different sizes so they need to be corrected and resized in the processing image stage.



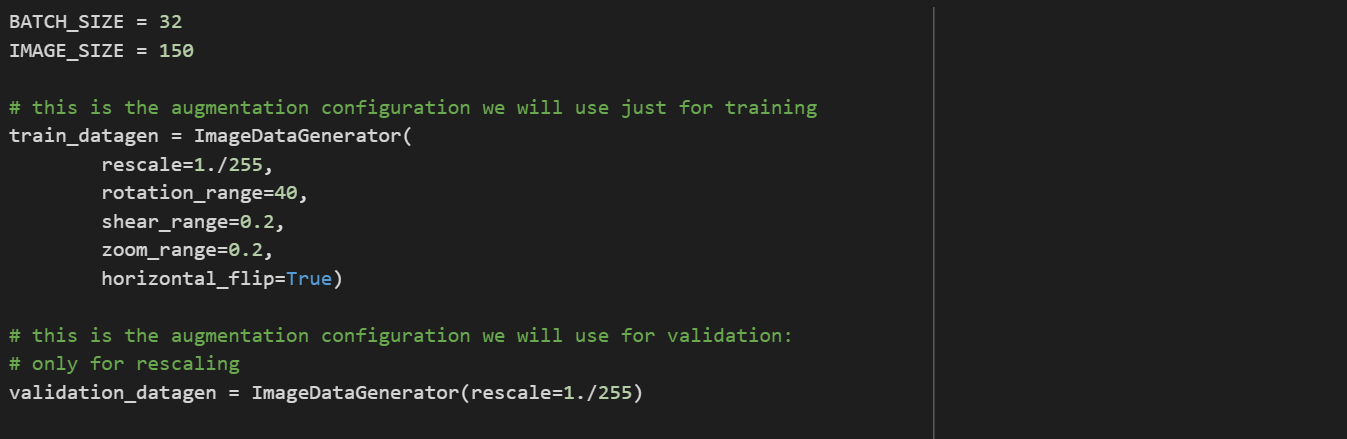
****

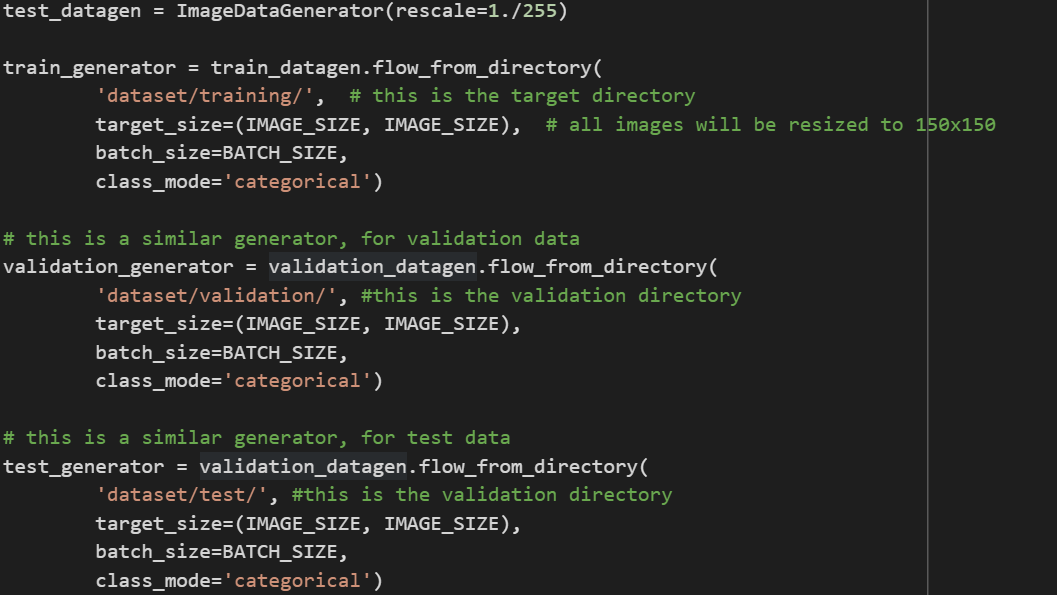
**Image Processing and Data Augmentation**

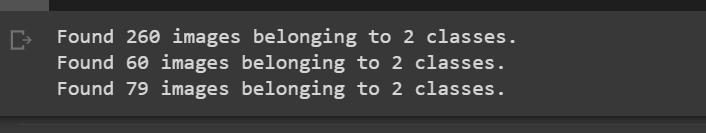
In order to make the most of our training examples, we will “augment” them via a number of random transformations, so that our model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalize better. In Keras, this can be done via the ImageDataGenerator class. This class allows you to configure random transformations and normalization operations to be done on your image data during training. It also instantiates generators of augmented image batches (and their labels) via *.flow*(data, labels) or .*flow\_from\_directory*(directory). These generators can then be used with the Keras model methods that accept data generators as inputs, *fit\_generator, evaluate\_generator* and *predict\_generator*.



Loading the dataset with the image generator from Keras. We will use the following configurations: rescaling, rotation, shear range, zoom and horizontal flip.







**Model Implementation**

The CNN model works in two steps: feature extraction and Classification

Feature Extraction is a phase where various filters and layers are applied to the images to extract the information and features out of it and once it’s done it is passed on to the next phase (i.e. Classification) where they are classified based on the target variable of the problem.

A CNN model looks like this:

* Input layer
* Convolution layer + Activation function
* Pooling layer
* Fully Connected Layer

The First-block model has two convolutional layers with 32, 32 filters followed by a max pooling layer. The Second Block model has two convolutional layers with 64,64 filters followed by a max pooling layer. Then, we defined our fully connected layers for classification with one flatten, one dense, and one dropout layer, and at last the output layer with “*softmax*” activation function.

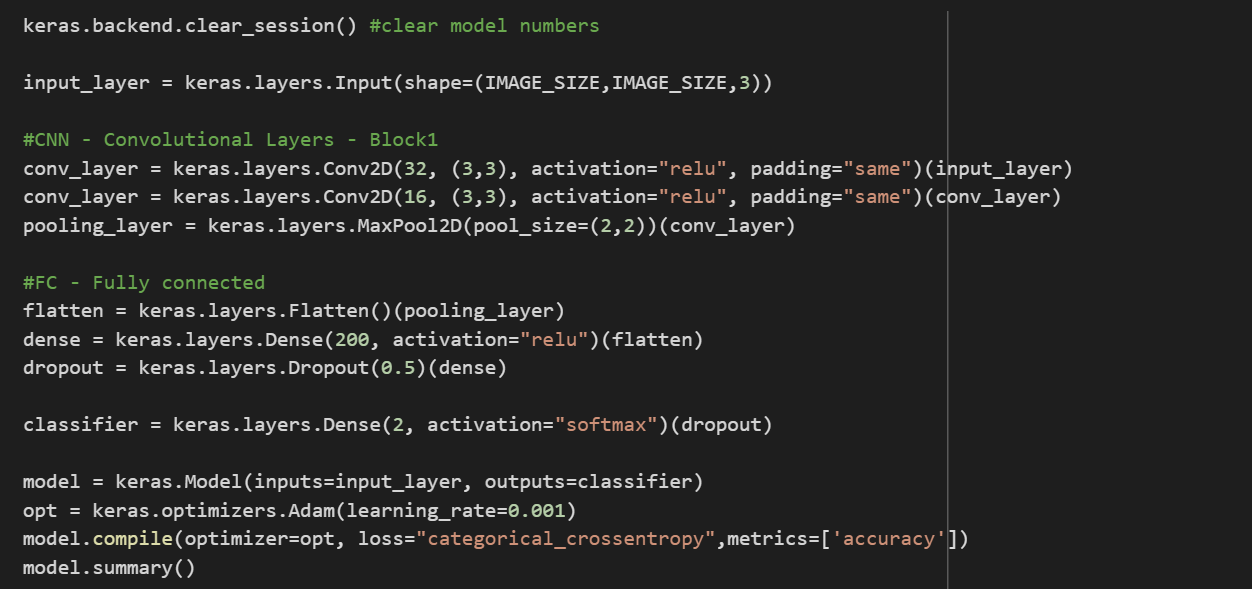
Next, we needed to compile our model. Compiling the model takes three parameters: optimizer, loss and metrics.

The optimizer controls the learning rate. We will be using ‘*adam’* as our optimizer. Adam is generally a good optimizer to use for many cases. The Adam optimizer adjusts the learning rate throughout training.

The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

We will use ‘*categorical\_crossentropy’* for our loss function. A lower score indicates that the model is performing better.

we used the ‘*accuracy’* metric to see the accuracy score on the validation set when we were training the model.



Below is the Summary of the model.

Text

Description automatically generated

**Training CNN Model:**

To train, we will use the ‘fit()’ function on our model with the following parameters: training data , validation data, and the number of epochs.

The number of epochs is the number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. For our models, we will set the number of epochs to 70.

The training model returns an accuracy of 92.31% for training and 86.67% for validation. And if we observe the last 5 epochs in validation, we get some results above 83%.

Text

Description automatically generated

Code Snippet to plot loss vs validation loss and accuracy vs validation accuracy.

Text

Description automatically generated

A picture containing chart

Description automatically generated

**Preliminary Scores:**

**Text

Description automatically generated**

We got Train accuracy as 94 %, Validation accuracy as 86% and Test accuracy as 86%.

Test Result:

Graphical user interface, website

Description automatically generated

We try to increase the epoch and we figure out that the model was overfitting, so we keep the epoch to 70.

**Comparing the Models**

Chart, bar chart

Description automatically generated

The most accurate classification is CNN, which is closely followed by SVC, but the Decision Tree is the least accurate.SVM accuracy is 78%,Decision tree accuracy is 75 % and CNN accuracy is 84%. Some of the research paper (Reference Added) also shows that CNN is better then SVM and Decision tree.

**Improving the Accuracy of CNN**

While our findings show that the accuracy of CNN is highest, its score in our study is lower than its accuracy in other studies. Even though the CNN is highest we need improve model accuracy by 7 to 8% by which it will close to 95 to 96%.

There are several ways through which the accuracy of CNN can be improved. First, increasing the training data improves the accuracy of CNN (Hasan et al., 2019). On the other hand, (Mana et al., 2021) found that the accuracy of CNN could be improved by optimizing the filter size of the convolution layer and max pooling of 512 by 512 images. This approach makes it possible to quickly obtain a highly accurate image classification model. Lastly, the use of bimodal image enhancement will have a significant impact on the accuracy of CNN. It is important to ensure that a good bimodal enhancement algorithm is used.

**6. CONCLUSIONS**

In this article, we have established a database of weather phenomena images under meteorological criterion. This database contains 400 images with 2 weather classes: good or bad. We proposed a three classification models using Keras and Tensorflow due to its simplicity and ease of code.

Three models have been implemented from scratch to find out their learning behavior with few samples in the dataset. Then a model has been implemented using SVC ,Decision Tree and on CNN we got better performance on a train and test model.

From the results obtained, these models can learn features of weather phenomena well, specially the one with CNN which is effective for weather classification. However, this model confuses some categories of weather phenomena, which may be due to the similarity and complexity of the images.

Overall, the classification accuracy of CNN model is 84% for test.

**7. ACKNOWLEDGEMENTS**

Our team would like to acknowledge and give our warmest thanks to our instructor Bukasa Tshilombo for his constant support and for guiding us in the right path. We would also like to give thanks to City University of Seattle for including this course in our curriculum.

**9.** **REFERENCES**

Bhathena, J. (2021, November 26). *Weather image recognition*. Kaggle. Retrieved October 23, 2022, from https://www.kaggle.com/datasets/jehanbhathena/weather-dataset

Google. (n.d.). Google. Retrieved November 19, 2022, from <https://firebase.google>.

Hasan, M., Ullah, S., Khan, M. J., & Khurshid, K. (2019). Comparative analysis of SVM, ANN and CNN for classifying vegetation species using hyperspectral thermal infrared data. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, (pp. 1861-1868).

Lazo, J. K., Morss, R. C., & Demuth, J. L. (2009). Sources, perceptions, uses, and values of weather forecasts. *Bulletin of the American Meteorological Society, 90*(6), 785–798. doi:https://doi.org/10.1175/2008BAMS2604.1

Mana, K., Asami, Y., Yamada, T., & Sugimori, H. (2021). Improvement in the convolutional neural network for computed tomography images. *Applied Sciences, 11*, 1-13.

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., . . . Stadtler, S. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions, 379*, 1-22. doi:https://doi.org/10.1098/rsta.2020.0097

Wang, P., Fan, E., & Wang, P. (2021). Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters, 141*, 61-67. doi:https://doi.org/10.1016/j.patrec.2020.07.042

DNB, Katalog der Deutschen Nationalbibliothek. (n.d.). Retrieved December 7, 2022, from https://d-nb.info/1188414305/34