JULY 2020

A PROJECT ON CAT - DOG CLASSIFICATION USING NEURAL NETWORKS

TRAINING IN DATA ANALYTICS, MACHINE LEARNING AND AI USING PYTHON

DiginiqueTechlabs
Ideal for Geeks

Submitted by:

JAYASUDHAN I
CENTRAL UNIVERSITY OF TAMIL NADU

Under the Guindance of :

MR.BIPULSHAHI
CORPORATE TRAINER

ACKNOWLEDGEMENT

- My sincere gratitude and thanks towards my project paper guide BipulShahi, Corporate Trainer, Developer, Traveller IOT, Artificial Intelligence, Robotics, Cloud Computing, Android Apps
- It was only with his backing and support that I could complete the report.
- He provided me all sorts of help and corrected me if ever seemed to make mistakes. I have no such words to express my gratitude.

ABSTRACT

- Image classification is a complex process that may be affected by many factors. Because classification results are the basis for many environmental and socioeconomic applications, scientists and practitioners have made great efforts in developing advanced classification approaches and techniques for improving classification accuracy.
- Image classification is used in a lot in basic fields like medicine, education and security. Correct classification has vital importance, especially in medicine. Therefore, improved methods are needed in this field. The proposed deep CNNs are an often-used architecture for deep learning and have been widely used in computer vision and audio recognition.
- To effectively manage all this data, we need to have some idea about its contents. Automated processing of image contents is useful for a wide variety of image-related tasks.
- For computer systems, this means crossing the so-called semantic gap between the pixel level information stored in the image files and the human understanding of the same images. Computer vision attempts to bridge this cap

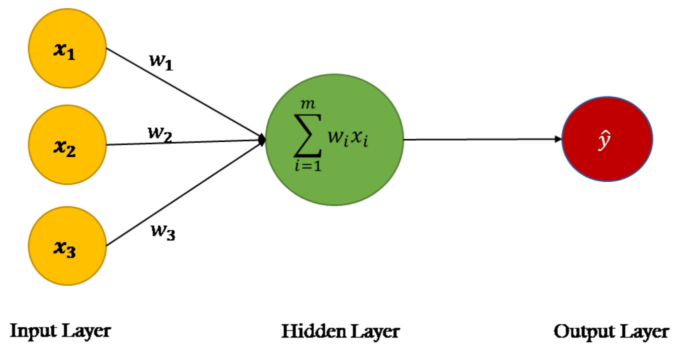
ABSTRACT

- This cat-dog classifier model using neural networks is a kind of image classification model which will be able to differentiate between cat and dog images very effectively.This model converts the non- readable data into machine readable data and uses these data to predict whether the given image is cat or dog.
- With the help of this image classification model one can distinguish between various objects like flowers, persons, Automobiles, Medical uses etc..

INTRODUCTION TO NEURAL NETWORKS

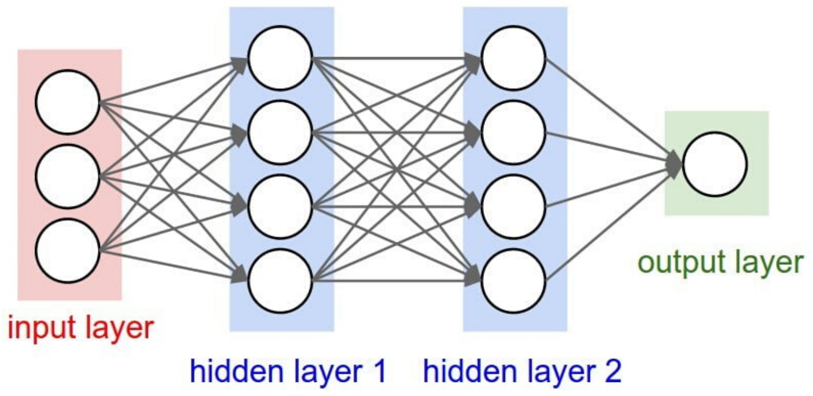
- A neural network refers to a series of algorithms that endeavors to recognize underlying relationships in datasets through a process that mimics the human brain. Neural networks are a computing system with interconnected nodes that work more like the neurons in a human brain.
- We use neural networks to recognize correlations and hidden patterns in raw data and also to cluster and classify raw data and to continuously learn and improve over time. Neural networks are used to solve complex problems people face in real-world situations.
- These networks can learn and model the relationships between inputs and outputs that are complex and nonlinear. These are also used to reveal hidden relationships and predictions, and model variances needed to predict rare events (fraud detection) and model highly volatile data like financial time-series data.
- We use a neural network to improve decision processes in areas such as Medicare fraud detection and credit card.Process and quality control.Targeted Marketing.Robotic Control System.Character and Voice recognition.Optimization of logistics for transportation networks.Computer vision for interpreting raw photos and videos.

ARTIFICIAL NEURAL NETWORKS



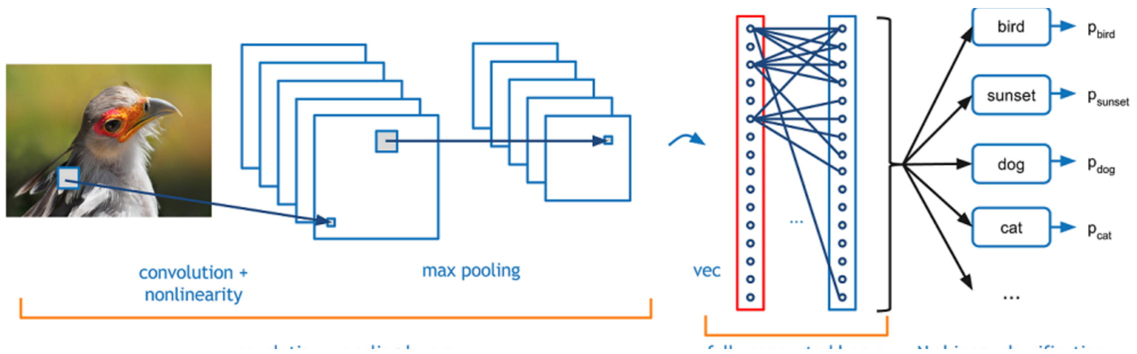
An artificial neural network (ANN) is the piece of a computing system designed to simulate the way the human brain analyzes and processes information. It is the foundation of artificial intelligence (AI) and solves problems that would prove impossible or difficult by human or statistical standards. ANNs have self-learning capabilities that enable them to produce better results as more data becomes available.

MULTI -LAYER NETWORKS



A Multi layer typically includes three types of layers: an input layer, one or more hidden layers and an output layer. The input layer usually merely passes data along without modifying it. Most of the computation happens in the hidden layers. The output layer converts the hidden layer activations to an output, such as a classification. A multilayer feed-forward network with atleast one hidden layer can function as a universal approximator.

CONVOLUTION NEURAL NETWORKS



- A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which 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.
- The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.
- The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

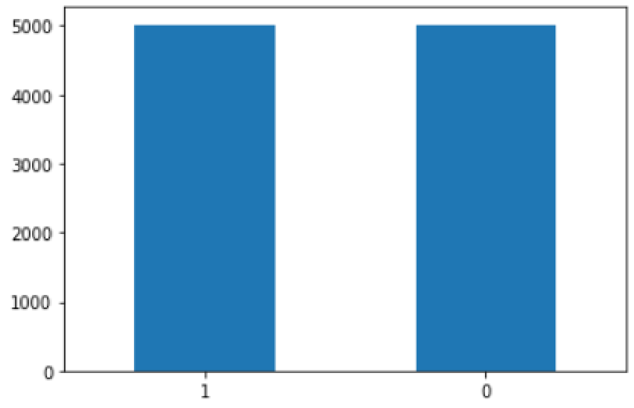
DESCRIPTION OF THE DATASET

- The dataset for this project has been download from the kaggle website and it is a open datasource anyone can download it from the below mentioned link
- https://www.kaggle.com/tongpython/cat-and-dog
- This dataset consist of 10,028 images of cats and dogs and each category is having equal number of images.
- These non- readable images are getting converted into machine readable data and these data are used to build the model using various types of methods like Artificial Neural Networks , Convolutional Neural Networks, Decision Tree classification and Guassian Naive Bayes.
- Once the training has been done, the test dataset is used to test the accuracy of the various models in order to find the good working model with highest accuracy.

STEP 1: CONVERTING RAW DATA INTO READABLE FORMAT

```
import pandas as pd
import matplotlib.pyplot as plt
label = pd.DataFrame(label)
label[0].value_counts().plot.bar(rot=0)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f39c672aa20>
```



```
[ ]  from tqdm import tqdm
     features = []
     for i in tqdm(os.listdir(loc)):
       path = os.path.join(loc,i)
       f = cv2.imread(path)
       fr = cv2.resize(f,(70,70))
       features.append(fr)
```

```
100%|██████████| 10028/10028 [01:20<00:00,
```

```
[ ]  import numpy as np
     X = np.array(features)
     Y = np.array(label)
     y = np.array(label)
```

```
[ ]  X.shape
```

```
(10028, 70, 70, 3)
```

```
[ ]  Xt = X.reshape(10028,14700)
```

- With the help of the Matplot library and by using the count plot, we are able to see the distribution of data between each category.
- Here the cat category is given a label as "0" whereas the dog category is given a label as "1".

- After the visualization, we are converting this non-readable data into a readable format with the help of libraries like os.listdir and cv2.imread.
- This os.listdir library reads all the images which is stored inthe location (loc)
- This cv2 library converts all the images into pixel format into order to convert into readable format.
- Once the data has been read these data are stored in the list called features and target has been stored in the list called label.
- Then these list data are converted into numpy array with the help of the numpy library by importing it in to the notebook.

STEP 2: ARTIFICIAL NEURAL NETWORK MODEL PREPARATION AND PREDICTION

```
import keras
from keras import layers
from keras.utils import to_categorical
```

```
model = keras.Sequential()
model.add(layers.Dense(200, activation = 'relu' , input_dim = Xt.shape[1]))
model.add(layers.Dense(100 , activation = 'relu'))
model.add(layers.Dense(2, activation = 'softmax'))
```

- Artificial neural network model can be created by importing keras library. Here in this model we have used two hidden layers with Rectified linear unit as an activation function and one output layer with softmax as an activation function

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 200)               2940200   
_________________________________________________________________
dense_2 (Dense)              (None, 100)               20100     
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 202       
=================================================================
Total params: 2,960,502
Trainable params: 2,960,502
Non-trainable params: 0
_________________________________________________________________
```

- The command model.summary gives us the details of parameters included in that model.

```
Yt = to_categorical(Y)
```

```
Yt
```

```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

- The command to_categorical will change the target column into one hot coding format.

```
f= model.fit(xtrain,ytrain,epochs=15)
```

```
Epoch 1/15
7521/7521 [==============================] - 2s 207us/step - loss: 0.0417 - accuracy: 0.9900
Epoch 2/15
7521/7521 [==============================] - 2s 208us/step - loss: 0.0594 - accuracy: 0.9824
Epoch 3/15
7521/7521 [==============================] - 2s 205us/step - loss: 0.0413 - accuracy: 0.9903
Epoch 4/15
7521/7521 [==============================] - 2s 209us/step - loss: 0.0251 - accuracy: 0.9948
Epoch 5/15
7521/7521 [==============================] - 2s 207us/step - loss: 0.0235 - accuracy: 0.9947
Epoch 6/15
7521/7521 [==============================] - 2s 213us/step - loss: 0.0166 - accuracy: 0.9980
Epoch 7/15
7521/7521 [==============================] - 2s 226us/step - loss: 0.0593 - accuracy: 0.9874
Epoch 8/15
7521/7521 [==============================] - 2s 225us/step - loss: 0.0188 - accuracy: 0.9977
Epoch 9/15
7521/7521 [==============================] - 2s 226us/step - loss: 0.0171 - accuracy: 0.9979
Epoch 10/15
7521/7521 [==============================] - 2s 230us/step - loss: 0.0111 - accuracy: 0.9997
Epoch 11/15
7521/7521 [==============================] - 2s 224us/step - loss: 0.0093 - accuracy: 0.9997
Epoch 12/15
7521/7521 [==============================] - 2s 220us/step - loss: 0.0088 - accuracy: 0.9997
Epoch 13/15
7521/7521 [==============================] - 2s 208us/step - loss: 0.0077 - accuracy: 0.9997
Epoch 14/15
7521/7521 [==============================] - 2s 204us/step - loss: 0.0077 - accuracy: 0.9999
Epoch 15/15
7521/7521 [==============================] - 2s 207us/step - loss: 0.0064 - accuracy: 1.0000
```

- The training accuracy for the artificial neural networks model is 100%

```
model.evaluate(xtest,ytest)
```

```
2507/2507 [==============================] - 0s 140us/step
[0.35472556473156064, 0.8775428533554077]
```

- The test accuracy for the artificial neural networks model is 88%

PREDICTION ON A UNKNOWN DATASET USING ARTIFICIAL NEURAL NETWORKS MODEL

```
p = ['cat','dog']
```

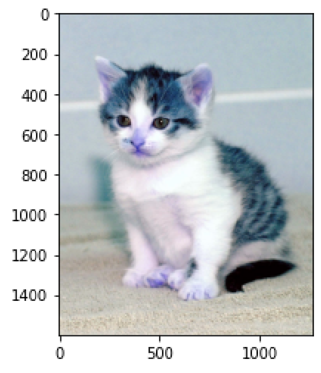
```
k = cv2.imread('cat4.jpg')
kt = cv2.resize(k,(70,70))
model.predict(kt.reshape(1,14700))
```

```
array([[1., 0.]], dtype=float32)
```

```
p[np.argmax(model.predict(kt.reshape(1,14700)))]
```

```
'cat'
```

```
plt.imshow(k)
plt.show()
```



```
k1 = cv2.imread('dog.jpg')
kt1 = cv2.resize(k1,(70,70))
model.predict(kt1.reshape(1,14700))
```

```
array([[0., 1.]], dtype=float32)
```

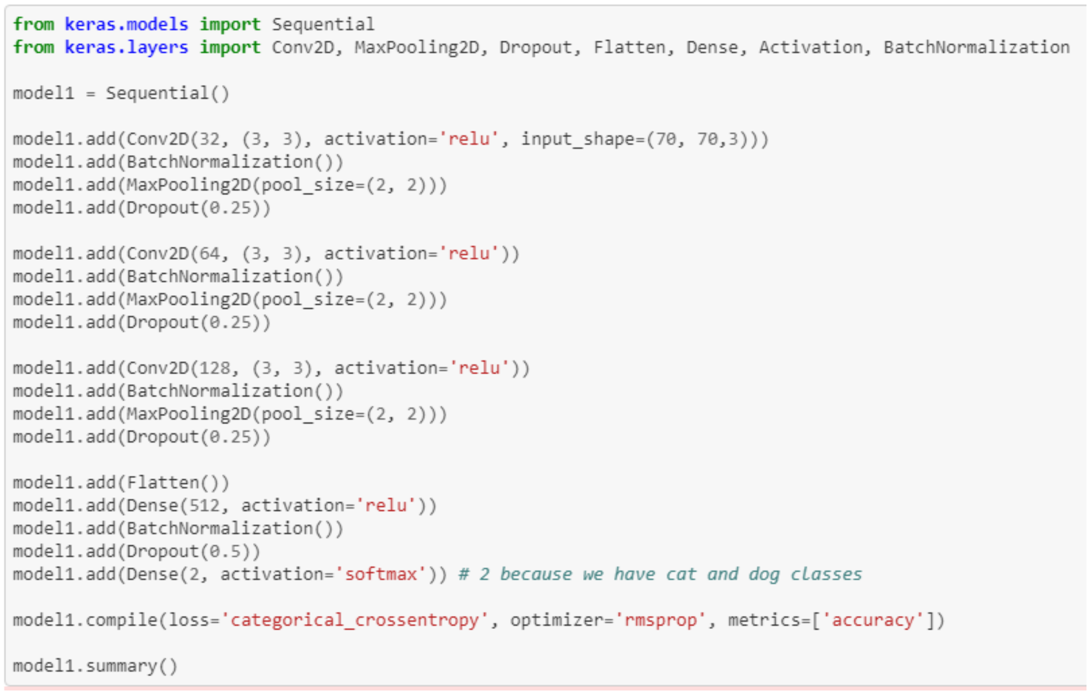
```
p[np.argmax(model.predict(kt1.reshape(1,14700)))]
```

```
'dog'
```

```
plt.imshow(k1)
plt.show()
```

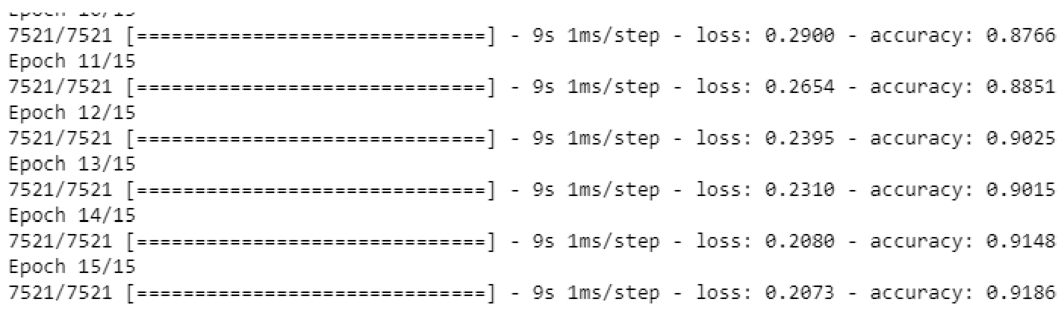
- In both these unknown datasets the artificial neural network model predicted exactly as it is with the highest accuracy. Hence this model is good to classify images.

STEP 2: CONVOLUTION NEURAL NETWORK MODEL PREPARATION AND PREDICTION

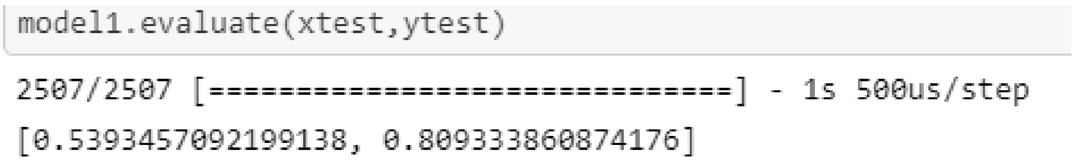


```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation, BatchNormalization

model1 = Sequential()

model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(70, 70,3)))
model1.add(BatchNormalization())
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.25))

model1.add(Conv2D(64, (3, 3), activation='relu'))
model1.add(BatchNormalization())
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.25))

model1.add(Conv2D(128, (3, 3), activation='relu'))
model1.add(BatchNormalization())
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.25))

model1.add(Flatten())
model1.add(Dense(512, activation='relu'))
model1.add(BatchNormalization())
model1.add(Dropout(0.5))
model1.add(Dense(2, activation='softmax')) # 2 because we have cat and dog classes

model1.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

model1.summary()
```

- Convolution neural network model can be created by importing keras library. Here in this model we have used three convolution layer, three pooling layer, three Batch Normalisation layer with Rectified linear unit as an activation function and one flatten layer with softmax as an activation function



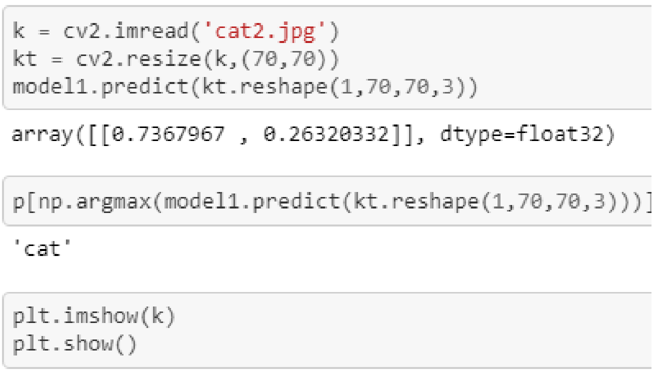
```
Epoch 10/15
7521/7521 [==============================] - 9s 1ms/step - loss: 0.2900 - accuracy: 0.8766
Epoch 11/15
7521/7521 [==============================] - 9s 1ms/step - loss: 0.2654 - accuracy: 0.8851
Epoch 12/15
7521/7521 [==============================] - 9s 1ms/step - loss: 0.2395 - accuracy: 0.9025
Epoch 13/15
7521/7521 [==============================] - 9s 1ms/step - loss: 0.2310 - accuracy: 0.9015
Epoch 14/15
7521/7521 [==============================] - 9s 1ms/step - loss: 0.2080 - accuracy: 0.9148
Epoch 15/15
7521/7521 [==============================] - 9s 1ms/step - loss: 0.2073 - accuracy: 0.9186
```



```
model1.evaluate(xtest,ytest)

2507/2507 [==============================] - 1s 500us/step
[0.5393457092199138, 0.809333860874176]
```

- In this convolutional neural networks model we have achieved a model with a training accuracy of 92% and test accuracy of 81%

PREDICTION ON A UNKNOWN DATASET USING CONVOLUTIONAL NEURAL NETWORKS MODEL



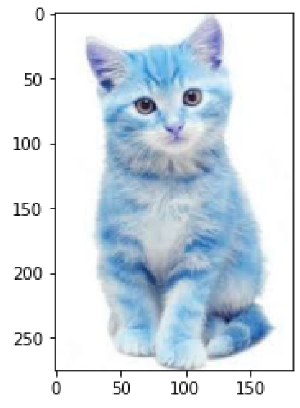
```
k = cv2.imread('cat2.jpg')
kt = cv2.resize(k,(70,70))
model1.predict(kt.reshape(1,70,70,3))
```

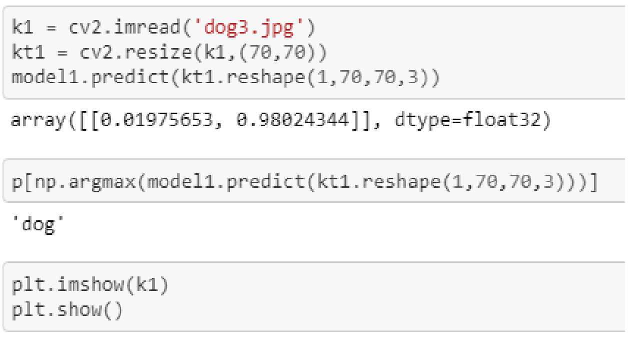
```
array([[0.7367967 , 0.26320332]], dtype=float32)
```

```
p[np.argmax(model1.predict(kt.reshape(1,70,70,3)))]
```

```
'cat'
```

```
plt.imshow(k)
plt.show()
```





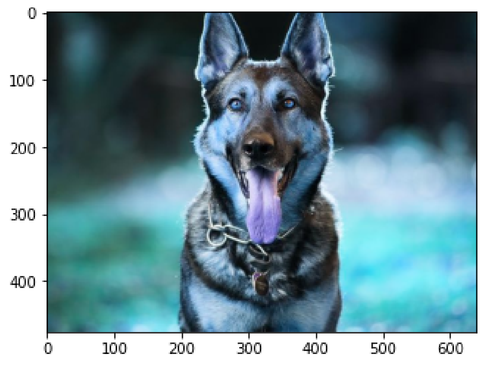
```
k1 = cv2.imread('dog3.jpg')
kt1 = cv2.resize(k1,(70,70))
model1.predict(kt1.reshape(1,70,70,3))
```

```
array([[0.01975653, 0.98024344]], dtype=float32)
```

```
p[np.argmax(model1.predict(kt1.reshape(1,70,70,3)))]
```

```
'dog'
```

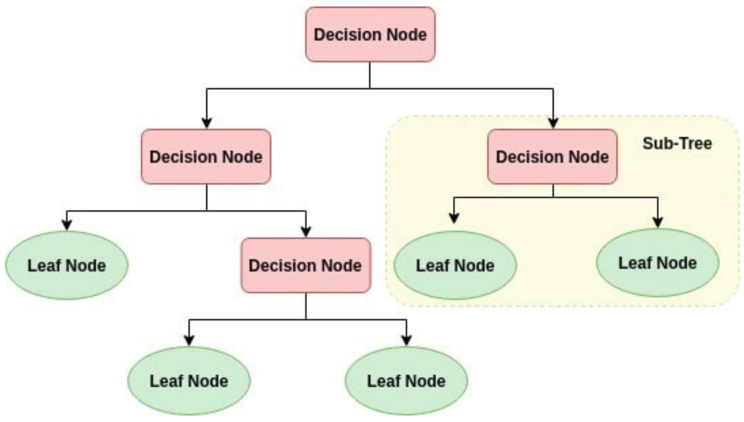
```
plt.imshow(k1)
plt.show()
```



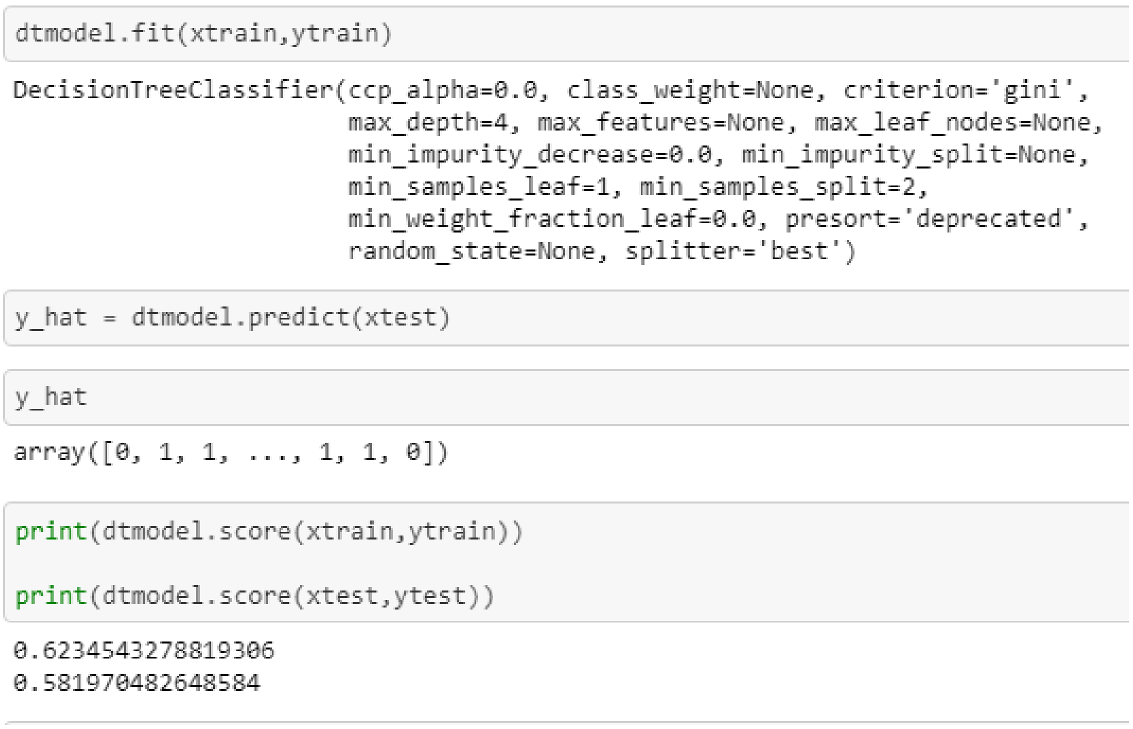
- In both these unknown datasets the convolutional neural network model predicted exactly as it is with the highest possible accuracy. Hence this model is good to classify images.

STEP 3: DECISION TREE CLASSIFICATION MODEL PREPARATION AND PREDICTION

- A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.
- The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning.
- This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.



- Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network.
- Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data.
- The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy



```
dtmodel.fit(xtrain,ytrain)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=4, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
y_hat = dtmodel.predict(xtest)
```

```
y_hat
```

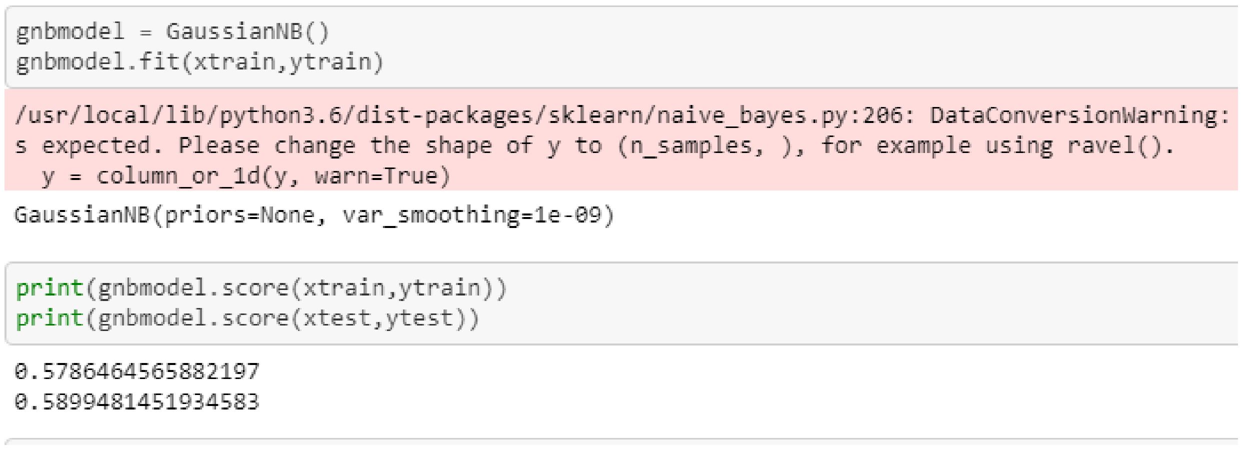
```
array([0, 1, 1, ..., 1, 1, 0])
```

```
print(dtmodel.score(xtrain,ytrain))

print(dtmodel.score(xtest,ytest))
```

```
0.6234543278819306
0.581970482648584
```

- This decision tree model is giving only 63 percent of training accuracy and only 59 percent of test accuracy since it is very much large dataset.
- Although the Decision tree classifier is much faster than neural networks it does not gives that much high accuracy when compared to neural networks.

STEP 4: NAIVE BAYES MODEL PREPARATION AND PREDICTION



```
gnbmodel = GaussianNB()
gnbmodel.fit(xtrain,ytrain)
```

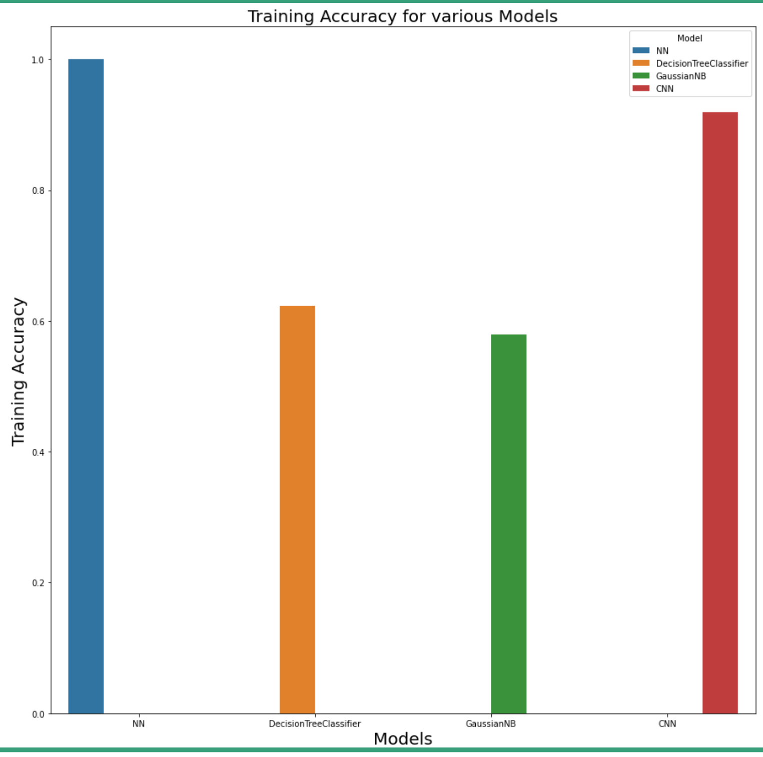
```
/usr/local/lib/python3.6/dist-packages/sklearn/naive_bayes.py:206: DataConversionWarning:
s expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
print(gnbmodel.score(xtrain,ytrain))
print(gnbmodel.score(xtest,ytest))
```

```
0.5786464565882197
0.5899481451934583
```

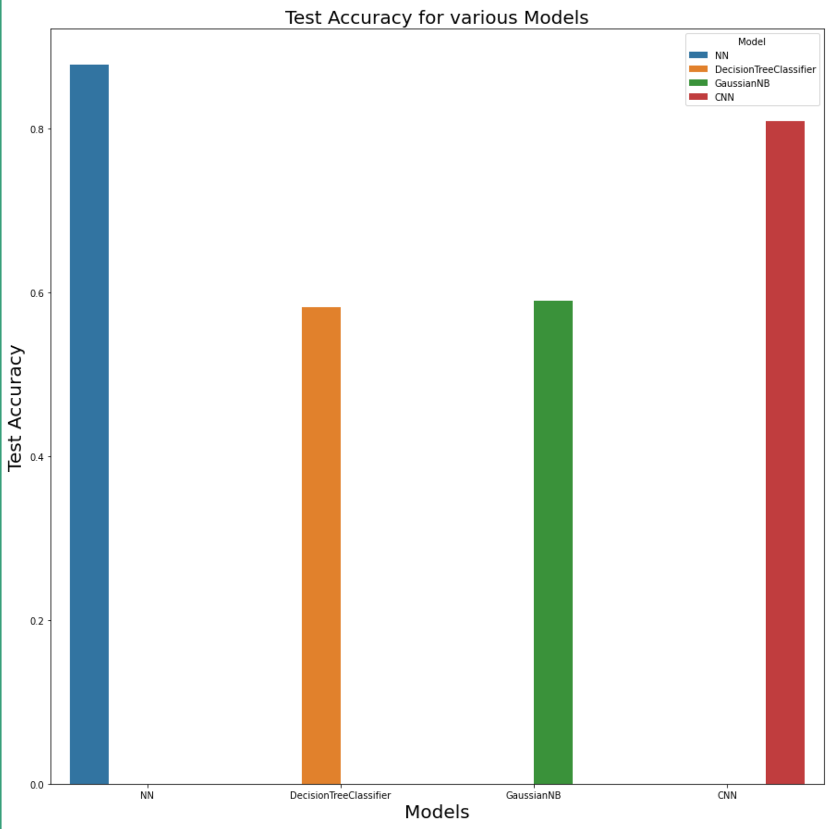
- This Guassian Naive Bayes model is giving only 57 percent of training accuracy and only 59 percent of test accuracy since it is very much large dataset.
- Hence this is not a good model for the classification purposes.

COMPARISON OF TRAINING ACCURACY BETWEEN VARIOUS MODELS



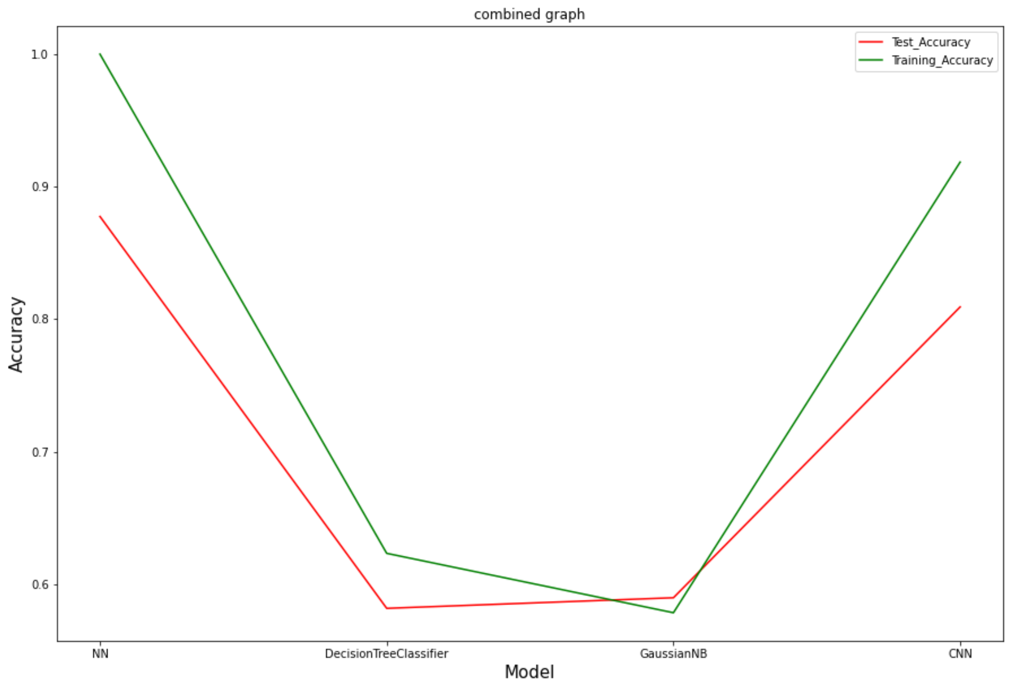
- This is a comparison of training accuracy between various models using barplot
- From this we can infer that Artificial neural networks has highest training accuracy of 100 percent
- Next to that CNN model has the second highest training accuracy of 90percent.

COMPARISON OF TESTING ACCURACY BETWEEN VARIOUS MODELS



- This is a comparison of testing accuracy between various models using barplot
- From this we can infer that Artificial neural networks has highest test accuracy of 88 percent
- Next to that CNN model has the second highest test accuracy of 81percent.

COMBINED LINE PLOT OF TRAIN AND TEST ACCURACY



CONCLUSION

- This project has practiced different machine learning technique and different models for data training attempting to discover a representation of good image classification model that allow their effective recognition and to achieve the highest accuracy of predicting images i.e whether the image is cat or dog.
- Thus, this study settled on classifying a given image using four different algorithms and consequently testing its accuracy.
- This project built handwritten cat dog classification model and evaluated their performances on dataset which is a open source from kaggle website and then find out the appropriate model for doing the image classification.
- From this project we can come to a conclusion that Neural Networks is giving more accuracy and good performance on predicting and classifying the image compared to the normal classifiers like knn classifier, Decision tree classifier, Naive Bayes etc.

GITHUB NOTEBOOK LINK

https://github.com/Jayasudhan1205/Capstone-project-IBM/blob/master/cat_dog_classifier1%20(1).ipynb