**CS-770: Machine Learning – Final Project Report**

**Name:** Godi Lakshmi Vasanthi

**WSU ID:** V567R326

**Email:** lxgodi@shockers.wichita.edu

**Objective:**

Train a deep learning classifier for any binary image classification task of your choice. The deep learning classifier should consist of a set of convolutional and fully connected layers followed by the final output layer for image classification. Motivate the choice of the model. Use appropriate metrics to evaluate the model on the test set. Compare the performance of your deep learning model with three traditional machine learning models of your choice evaluated on the same test set.

**Step 1: Importing libraries**

Importing necessary libraries

|  |
| --- |
| #Importing dependencies  Import keras  from keras.datasets import cifar10  import matplotlib.pyplot as plt  from keras.models import Sequential  from keras.layers import Dense  from keras.layers import Dropout  from keras.layers import Flatten  from keras.constraints import maxnorm  from keras import datasets, layers, models  from keras.layers.convolutional import Conv2D  from keras.layers.convolutional import MaxPooling2D  from keras.utils import np\_utils  import numpy as np  import os as library  import numpy as npy\_arrays  from sklearn import datasets  from sklearn.metrics import roc\_curve, auc  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import label\_binarize  from sklearn import metrics as sk\_metrics  from sklearn import linear\_model as lin\_mdl  from sklearn.pipeline import Pipeline as Pipe\_Ln  import matplotlib.pyplot  from sklearn.ensemble import RandomForestClassifier  from sklearn.neighbors import KNeighborsClassifier |

**Step 2: Data Pre-processing**

To train a deep learning classifier considering the dataset from Kaggle. It has two categories of data one of which contains images of grass and non-grass.

This dataset has 434 images of grass and 130 images with non-grass.

|  |
| --- |
| # Directory with our training grass pictures  train\_grass\_dir = os.path.join('/tmp/train/grass')  # Directory with our validation grass pictures  valid\_grass\_dir = os.path.join('/tmp/valid/grass') |

|  |
| --- |
| img\_ht = 256 #consider image height to be 225 pixels  img\_wt = 256 #consider image width to be 225 pixels  grass\_true\_path = '/tmp/train/grass' #path to the images with brain tumo  grass\_false\_path = '/tmp/valid/grass' #path to the images without brain  grass\_train\_image\_array = [] #initialize array to store train dataset  grass\_test\_image\_array = [] #initialize array to store test dataset  grass\_train\_lbl\_array = [] #initialize array to store binary labels for train array  grass\_test\_lbl\_array = [] #initialize array to store binary labels for test array  epoch\_count = 50 #number of epoches to use while training the model |

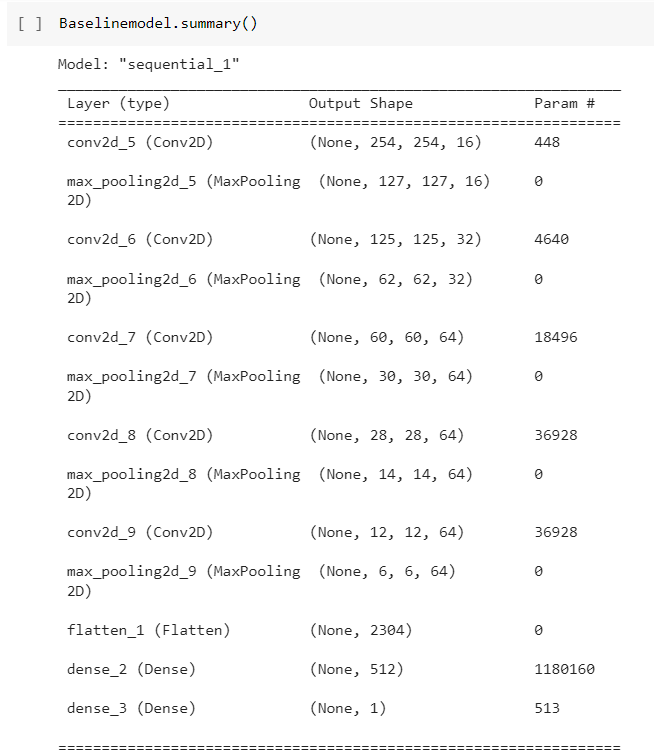
|  |
| --- |
| grass\_train\_image\_array = npy\_arrays.array(grass\_train\_image\_array)  grass\_train\_lbl\_array = npy\_arrays.asarray(grass\_train\_lbl\_array).astype('float32').reshape(len(grass\_train\_image\_array),-1)  grass\_test\_image\_array = npy\_arrays.array(grass\_test\_image\_array)  grass\_test\_lbl\_array = npy\_arrays.asarray(grass\_test\_lbl\_array).astype('float32').reshape(len(grass\_test\_image\_array),-1) |

**Step 3: Building model using CNN**

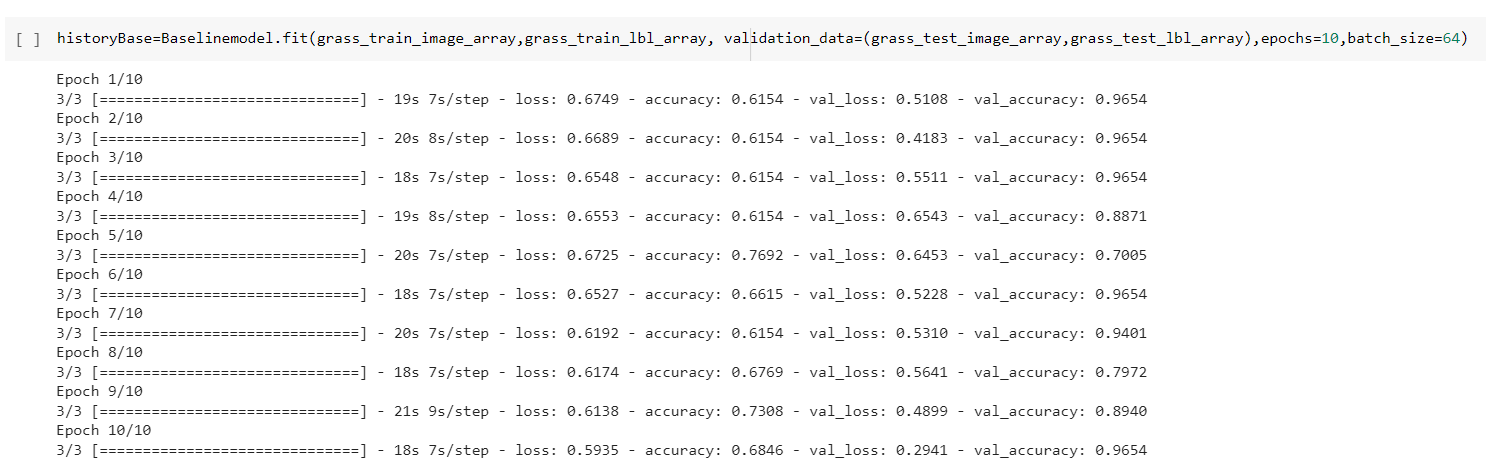
* Model is build using ‘adam’ as optimizer. Loss is determined using Binary Cross entropy for 10 epochs.
* Creating a neural classifier and adding layers to it

|  |
| --- |
| #Conv2D(16, (3,3), activation='relu', input\_shape=(200, 200, 3))  Baselinemodel = Sequential()  Baselinemodel.add(Conv2D(16, (3,3), activation='relu', input\_shape=(256, 256, 3)))  Baselinemodel.add(MaxPooling2D((2, 2)))  #Second convolution  Baselinemodel.add(Conv2D(32, (3,3), activation='relu'))  Baselinemodel.add(MaxPooling2D((2, 2)))  Baselinemodel.add(Flatten())  Baselinemodel.add(Dense(512, activation='relu'))  Baselinemodel.add(Dense(1, activation='sigmoid'))  # compile model  Baselinemodel.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy']) |

This gives the summary of layers added to the classifier

****

* compiling the classifier
* Adaptive Moment Estimation (ADAM) adjusts the learning rate for each parameter during training which can result in better performance

****

* calculating the test accuracy and loss
* predicting the trained model
* calculating the metrics for the trained model

****

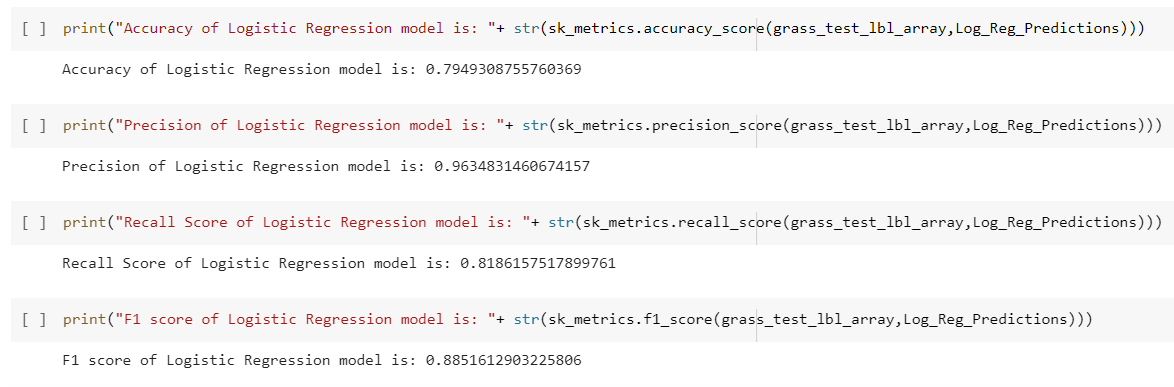
**Building Traditional Machine Learning models**

**Step 4: Building model using Logistic Regression**

* Creating a logistic regression model
* Training the logistic regression model
* Predicting the classification for test dataset using Logistic regression

|  |
| --- |
| Log\_Reg\_Model = lin\_mdl.LogisticRegression(max\_iter=100)  Log\_Reg\_Model.fit(Log\_Reg\_Train, grass\_train\_lbl\_array.ravel())  Log\_Reg\_Predictions = Log\_Reg\_Model.predict(Log\_Reg\_Test) |

calculating the metrics for the trained model

****

**Step 5: Building model using Random Forest**

* Creating a Random Forest model
* Training the Random Forest model
* Predicting the classification for test dataset using Random Forest

|  |
| --- |
| RF\_model=RandomForestClassifier()  RF\_model=RF\_model.fit(RF\_Train,grass\_train\_lbl\_array.ravel())  RF\_Model\_Prediction = RF\_model.predict(RF\_Test) |

calculating the metrics for the trained model

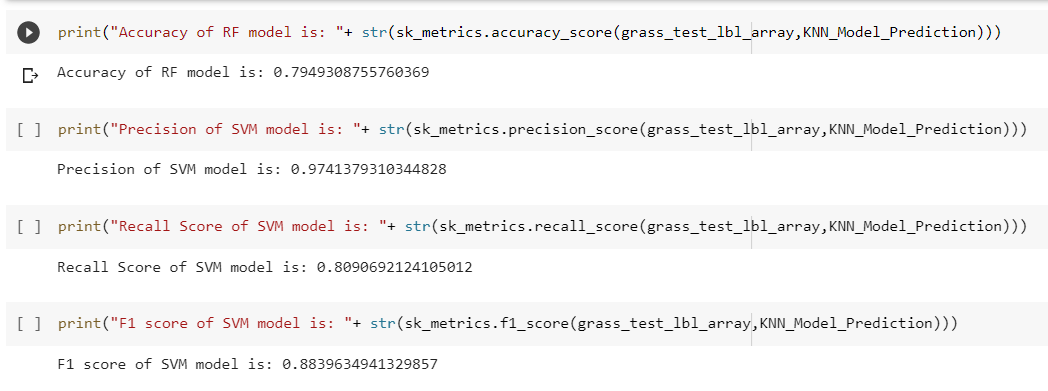
****

**Step 6: Building model using KNN Classifier**

* Creating a KNN classifier model
* Training the KNN classifier model
* Predicting the classification for test dataset using KNN classifier

|  |
| --- |
| knn=KNeighborsClassifier(n\_neighbors=2)  knn.fit(KNN\_Train,grass\_train\_lbl\_array.ravel())  KNN\_model=RF\_model.fit(KNN\_Train,grass\_train\_lbl\_array.ravel())  KNN\_Model\_Prediction = KNN\_model.predict(KNN\_Test) |

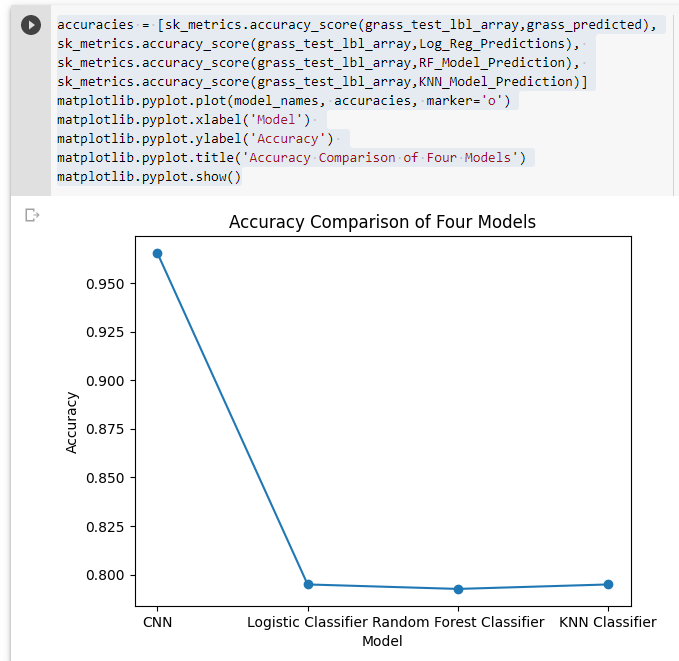
calculating the metrics for the trained model

****

**Step 7: Model Comparison**

Comparing the CNN and the traditional ML models

|  |
| --- |
| model\_names = ['CNN','Logistic Classifier', 'Random Forest Classifier', 'KNN Classifier'] |

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metrices** | **CNN** | **Logistic Classifier** | **Random forest Classifier** | **KNN Classifier** |
| **Accuracy** | **0.96** | **0.77** | **0.79** | **0.77** |
| **Precision** | **0.96** | **0.96** | **0.97** | **0.95** |
| **Recall** | **0.87** | **0.81** | **0.80** | **0.80** |
| **F1 score** | **0.92** | **0.88** | **0.88** | **0.87** |

On Comparison, it is evident that the accuracy obtained from the CNN model is high when compared to all other models. Because the CNN model uses multiple layers to learn the features of the images and learns them in hierarchical manner and uses backward propagation to predict the values which makes it more accurate.