University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project
7PAM2002-0509-2022
Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:

Star-Galaxy Image Classification and Models Comparison

Student Name and SRN:

Prachi Diwan and 21060794

Supervisor: Gulay Gurkan Uygun

Date Submitted: 31/08/2023

Word Count: 9699

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science **in Data Science** at the University of Hertfordshire.

I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Prachi Diwan

Student Name signature: [illegible]

Student SRN number: 21060794

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

The prominent aim of this project was to carry out the classification of astronomical objects which are stars and galaxies by the most preferred model that is CNN and thereafter compare it with another model that is Random Forest Classifier. Along with this, in my project work I have tried to perform exploratory analysis as well in both given models that eventually benefited me in giving a much better understanding of working of both in detail and concluding that my CNN model got an accuracy of 92% model and 75% in my Random Forest Classifier model. Therefore, it is evitable from the results that CNN is giving far better accuracy than Random Forest Classifier proof and reasons of which I will be including in further discussions. Moreover, I was also able understand different methodologies involved in the image classification which I have discussed in my report in further sections and was able to critically select CNN and RF classifier model as my basis for performing this project work and star-galaxy image classification in detail.

ACKNOWLEDGEMENT

Hereby, as I am approaching towards the end of my project work and master’s as well, so I would like to convey my deep gratitude to my supervisor for the project work, Dr Gulay Gurkan Uygun for constantly supporting me and guiding me wherever I needed help and clearance in any topic.

Along with this, I would also like to thank all professors who have helped me throughout my entire study in university and have imparted with knowledge and skills which will be beneficial in shaping my future career.

Lastly, I would also like to express my gratitude and appreciation towards my family, friends, and all those people who believed and supported me in this great journey of learning.

CONTENTS

Table of Contents:

CHAPTER-1: INTRODUCTION

As we all know that space is a huge place which consists of uncountable objects which are mainly classified as stars, galaxies, meteors, asteroids, quasars and so on that has always been an interesting and mind snatching topic for researchers to classify these objects from such a distance as seen from the earth. Nevertheless, with the advancement in technologies and introduction of machine learning models now it has been easier to perform such classification algorithm on these objects as these models are quite reliable and gives adequate amount of accuracy that has made the process easier and efficient.

The topic of classification of stars and galaxies dates to at least as far as era of Messier (1781) since then there have been some basic techniques to carry out this task which included visually inspection where astronomers would look at each photo of the stars or galaxies taken by telescope and then carefully inspect them based on their brightness, shape, any patterns and then classified them in either of the one category. Furthermore, some of the other methods were stellar evolution, morphological classification, spectroscopy, photometric analysis and so on. Starting from stellar evolution, in which the astronomers used to observe and monitor the life cycle of stars over a long period of time which helped them in understanding the phases and generating a decision at last.

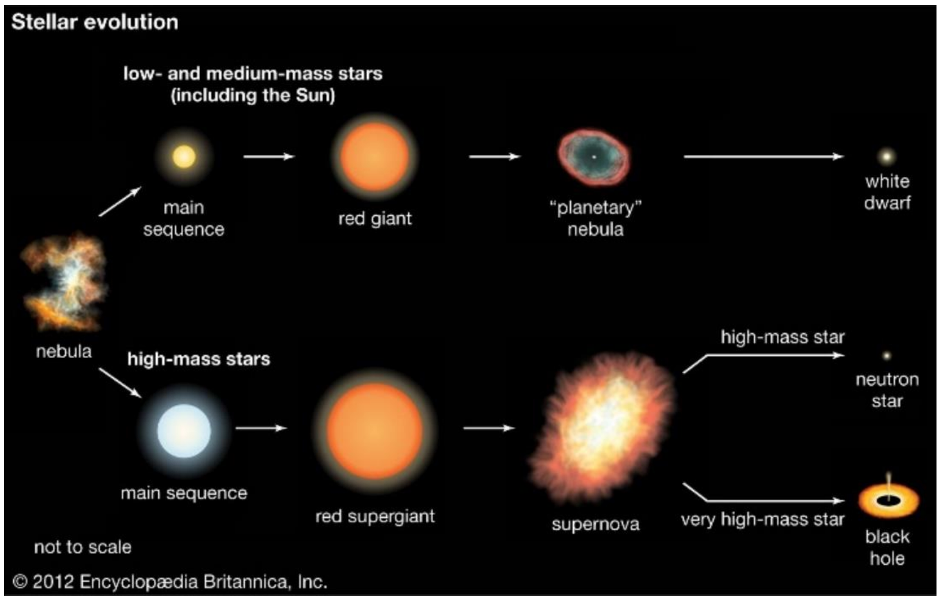


Figure 1: Evolution phases of stars from nebula to dwarf, star, or black hole (Source: https://www.britannica.com/science/star-astronomy/Star-formation-and-evolution).

On the other side, morphological classification solely focuses on galaxies in which an astronomer named Edwin Hubble (1963) developed system of classification for classification of the same based on visual appearance of galaxies such as elliptical, spiral, barred spiral and lenticular. He studied and performed augmentation techniques to fine tune the images and have a deep understanding of them which elevated the results and benefited in giving out better results.

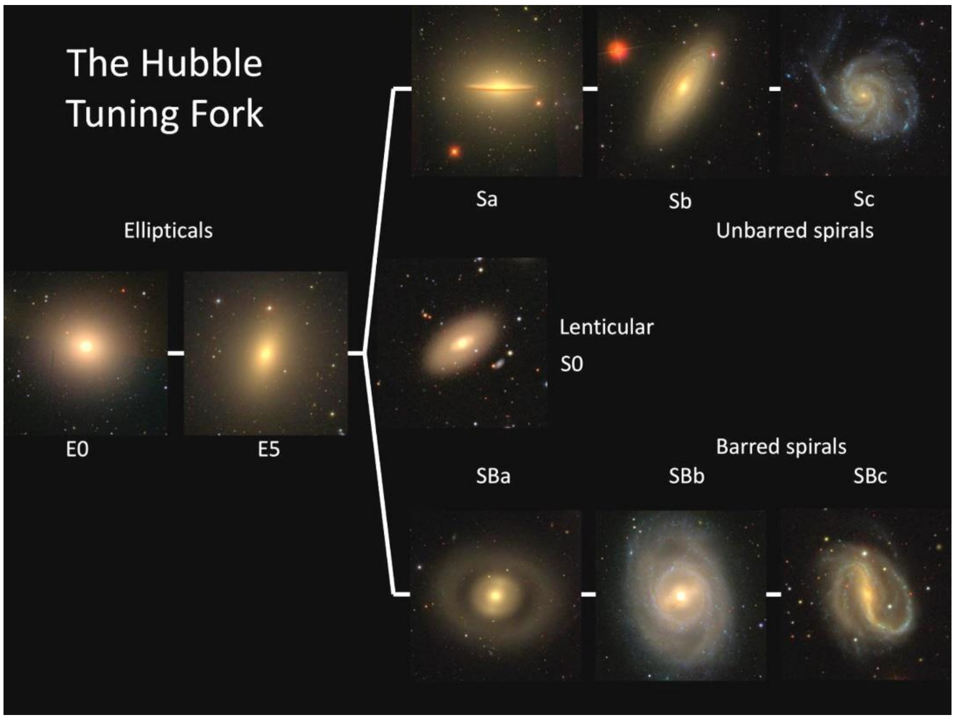


Figure 2: Evolution phases of galaxies into different shapes (Source: https://www.sci.news/astronomy/hubble-tuning-fork-07280.html).

Another method was spectroscopy which can be defined as splitting up of light into its constituent wavelengths forming a spectrum. This occurs because in any object such as stars or galaxies there are particles such as electrons whose energy levels are quantised where the emission and absorption of light will be occurring at specific wavelengths only giving the uneven colours of rainbow as shown in below given figure 3. Along with this, while performing same technique for galaxies, as seen in below figure 4 which is of a spiral and elliptical galaxy where it is observed that they both emit different colours of wavelengths like blue wavelength for spiral and red for elliptical that is dependent on the brightness the galaxies are emitting in the regions.

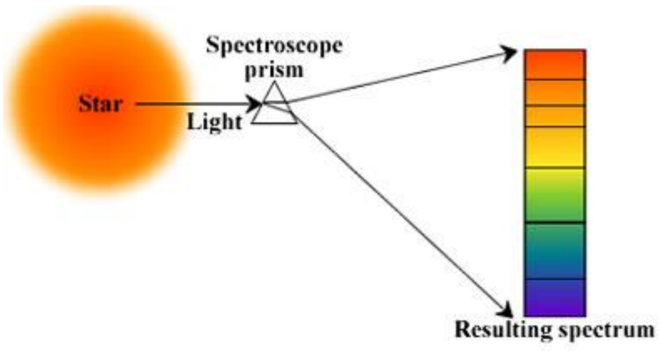


Figure 3: Light wavelength of star turning into a rainbow (Source: https://astronomy.swin.edu.au/cosmos/s/Spectroscopy)

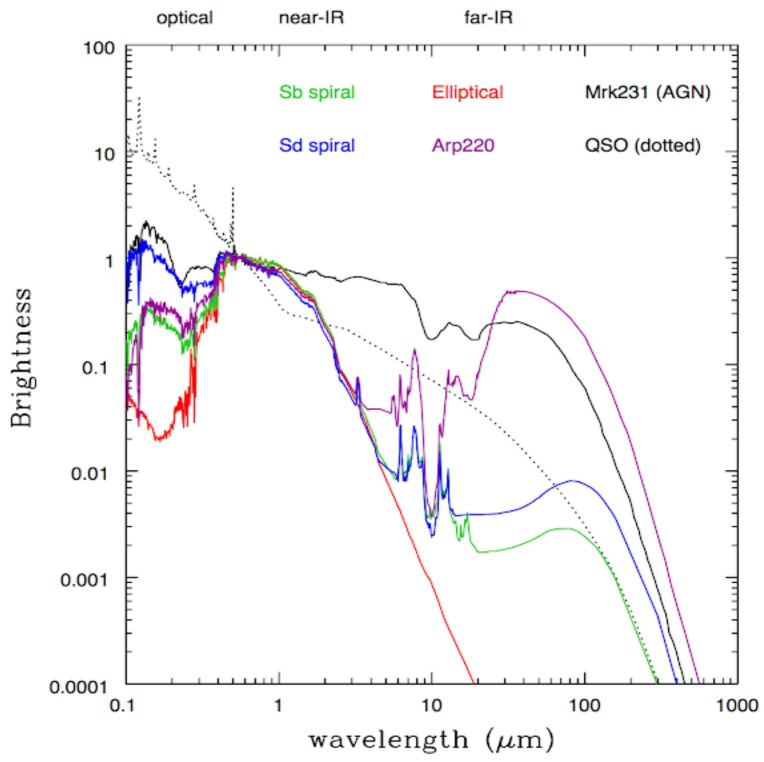


Figure 4: Wavelengths of spiral and elliptical galaxies (Source: https://candels-collaboration.blogspot.com/2012/08/the-multi-wavelength-shapes-of-galaxies.html)

Lastly, the term photometry can be defined as the measurement of brightness of astronomical objects by adding all its light emissions in total which helps in inferring the information of objects such as size, distance, temperature and so on.

Nevertheless, these spectroscopic techniques require high quality spectra to be generated from telescopes which in turn results in getting quite expensive, this is when the motivation to use images for classification of stars and galaxies comes into play as the machines can extract relevant information from features available in the images and train the model accordingly which reduces the human efforts and costs as well. Along with this, there are numerous scenarios where the likelihood between stars and galaxies images has been surreal and difficult to classify with naked human eyes and this problem has also been solved by the ML techniques for image classification which made me choose this topic for my project work as it helps in understanding of the space objects and methods to classify them correctly. An instance of the similarity of a star and galaxy when put together is shown below:

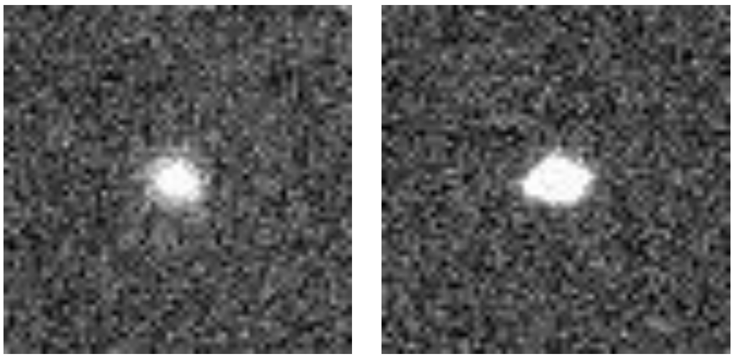


Figure 5: Image of a galaxy and star observed from a telescope and put together (Source: https://www.kaggle.com/datasets/divyansh22/dummy-astronomy-data)

Now, coming to the most popular and preferred techniques for classification of stars and galaxies in modern time by using machine learning models which have made this work easier as well as gives maximum accuracy also. These models are distinguished as Random Forest Classifier (RF), k-Nearest Neighbours (kNN), Support Vector Machines (SVM), Convolutional Neural Networks (CNN), Naïve Bayes, Logistic Regression (LR), Decision Trees and so on. To focus on the problem of categorising the images of stars and galaxies, I will be using CNN model and then further I will be comparing it with Random Forest Classifier one to analyse which model is carrying out better classification and giving better accuracy. Here, after performing classifications it is now evitable that Convolutional Neural Networks is one of the best models out of all other ones in giving highest amount of accuracy as shown in figure below.

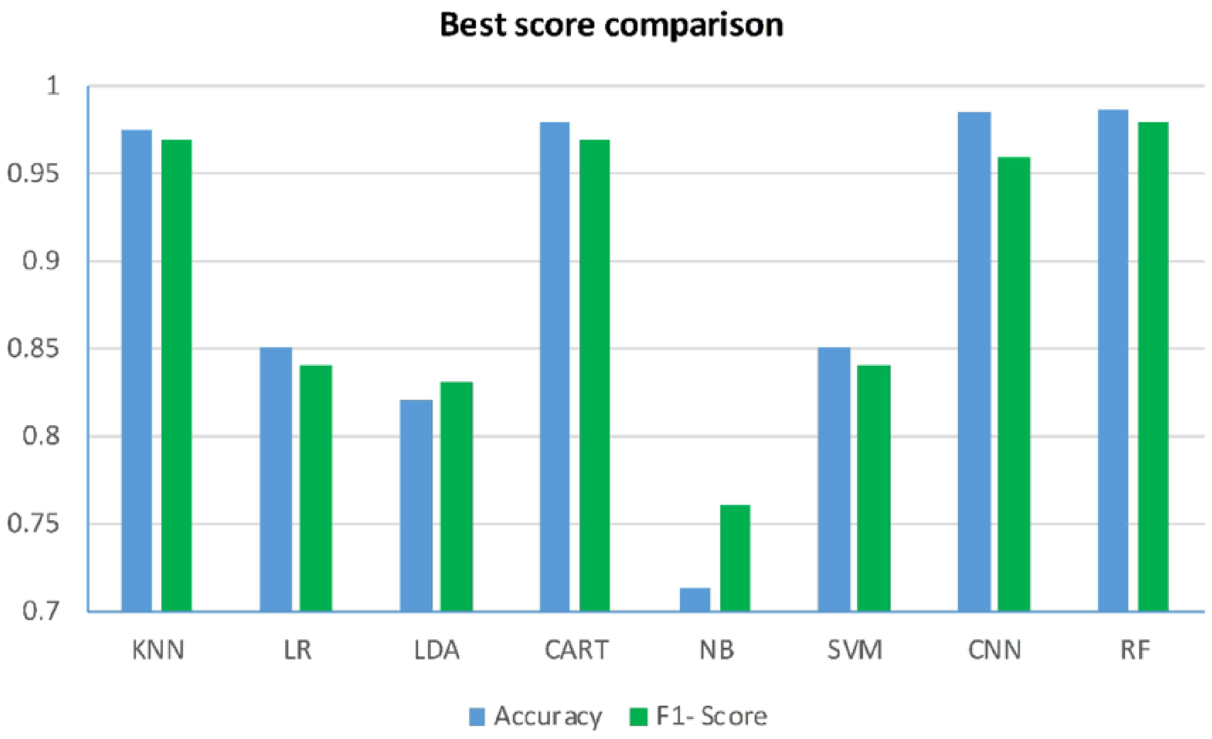


Figure 6: Graph showing accuracy for different ML models (Source: https://www.researchgate.net/publication/358468685_A_Sustainable_Model_for_Emergency_Medical_Services_in_Developing_Countries_A_Novel_Approach_Using_Partial_Outsourcing_and_Machine_Learning/figures?lo=1)

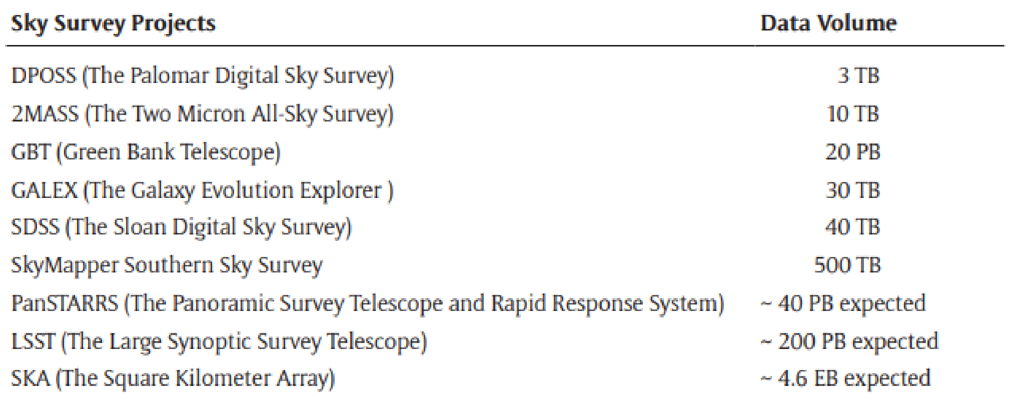
In my project work, I have used dataset provided from Kaggle website which consists of cutouts of images of stars and galaxies in different directories which makes this a supervised classification as the images are already categorised and have labels on them as well, further information for it will be given in dataset discussion part of my report. Thereafter, I have built my CNN model from scratch, compiled, and evaluated it in using different performance matrices like confusion matrix, precision, recall and ROC Curve. Moreover, in a similar manner I have built RF model in which I have trained some number of images on my given dataset and tested the model on remaining images as well as carrying out same kinds of performance matrices as CNN and then comparing both in detail. As a result, in my CNN model I have got an accuracy of 93% while my RF model has 73% accuracy, concluding that CNN is far better and convenient option than any other model.

CHAPTER-2: LITERATURE REVIEW

To commence with, while I was researching for the initial information of my project topic, I came across number articles that helped me in getting a deep and proper understanding of the methods, background, different surveys involved, results, drawbacks and so on. Some of them are as follows:

1. Understanding Astronomical Data –

In modern era, with increasing use of advanced technologies and tools we can gather loads and loads amount of data of space to use it for research purpose. Moreover, many surveys have come across since a long time now who are helping in collecting samples of images and data as well for performing numerous tasks on them for personal and public purpose. Here, below given is a list of surveys that have been established and are contributing towards gathering of data along with the size of data that they surveyed till date **(Zhang and Zhao, 2015)**. A noticeable thing is the enormous amount of data in TB, PB and EB that the surveys can extract which are widely used in different areas for different purposes. Furthermore, the list also includes some famous surveys that are SDSS, SkyMapper Southern Sky Survey, LSST which has their data collection in TB whereas SKA holds the highest amount equivalent to around 4.6 EB which is gigantic.



Sky Survey Projects	Data Volume
DPOSS (The Palomar Digital Sky Survey)	3 TB
2MASS (The Two Micron All-Sky Survey)	10 TB
GBT (Green Bank Telescope)	20 PB
GALEX (The Galaxy Evolution Explorer)	30 TB
SDSS (The Sloan Digital Sky Survey)	40 TB
SkyMapper Southern Sky Survey	500 TB
PanSTARRS (The Panoramic Survey Telescope and Rapid Response System)	~ 40 PB expected
LSST (The Large Synoptic Survey Telescope)	~ 200 PB expected
SKA (The Square Kilometer Array)	~ 4.6 EB expected

Figure 7: Different surveys with their dataset size (Source: Zhang and Zhao, 2015).

After collection of data, these huge sets of data are stored in databases (e.g.: Oracle, SciDB, MS SQLserver, MySQL, MongoDB, SQLite) which are accessed by researchers and observers for variety of purposes to carry out different operations on them. In same respect, a concept was put forward by Szalay and Gray (2001) named as Virtual Observatory (Zhang and Zhao, 2015) which they formed a collection of datasets and all kinds of software tools to make a scientific environment in which scientists can gather to access these data collections and discover as well as analyse them in a properly manner. Thereafter, comes the data mining tasks to be performed on these datasets such as classification, regression, clustering, anomaly detection and time series analysis along with two followed up methods that are feature selection and feature extraction to make data clearer and avoid any noises in it.

Overall, this paper made me look closely to how the data is gathered, organizations involved in it, methods to carry out on data and so on which I have implemented in my coding part as I chose classification as my prominent topic of research in astronomical objects that is stars and galaxies. It also helped me in choosing adequate feature selection and extraction techniques that I performed while doing data augmentation part to train my images properly.

2. **Different ML algorithms for performing classification –**

In modern world era, with widespread usage of image capturing techniques and latest technologies it has become easier to classify stars and galaxies. These algorithms train themselves from the features of datasets automatically and in some of them we can even manually perform this task to increase accuracy which helps them in training adequately and giving results as per requirement. This paper has discussed in total five algorithms that includes k-Nearest Neighbours (kNN), Random Forest Classifier (RF), Naïve Bayes (NB), Support Vector Machines (SVM), Logistic Regression (LR), Decision Trees (DT) and have chosen Convolutional Neural Networks (CNN) as prominent model for performing the classification task between stars and galaxies (Savyanavar, 2023) which gave highest accuracy of 92.44% while other models gave an accuracy of around 70%. The details for these models are as below:

k-Nearest Neighbour (kNN): It is one of the most basic models out of all given, as it utilizes distance as a factor for classification. In this, the distance of the object that needs to classified is calculated along with calculation between training data objects, result of which gives the classification of object's class. This distance can be calculated by using Euclidean measure (Savyanavar, 2023) along with this the size of computation is directly proportional to the size of dataset as the degree of complexity increases in the similar manner.

Random Forest Classifier (RF): This model is quite preferred when doing image classification as it uses the algorithm of forming numerous decision trees and looking all of them simultaneously for getting to the results which helps in taking every parameter into context and that minimises the ratio of loss as well as increases accuracy.

Naïve Bayes (NB): It is probabilistic technique for categorizing the objects which uses a mathematical approach named as Bayes Theorem (Savyanavar, 2023). In this algorithm, it is applicable on supervised learning which means that the objects are having labels and have classes in initially. Moreover, it calculates the posterior probability P(X|Y) that is occurrence of an event X when an event Y has already occurred where each object is contributing equally and independently towards the outcome. So, this algorithm calculates this posterior probability for each class and when an object occurs as an input variable, it will be classified to the class having highest probability of it. Nonetheless, this algorithm requires rigorous knowledge of probabilities beforehand resulting in high computation costs as well (Savyanavar, 2023).

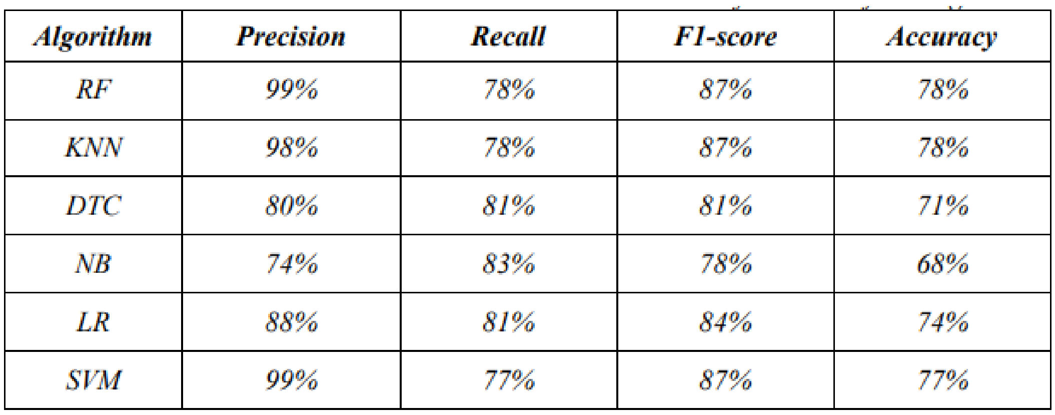
__Support Vector Machines (SVM):__ This algorithm comes forward as compared to others by researchers while doing classification, regression and so on. In SVM algorithm, a hyper-lane is drawn based on objects it has classified from the training dataset according to their features. Also, this hyper-lane is goes through the best possible middle part of the classification and on the either side of it will be the objects that is classified. Thereafter, when a new object is to be classified it will look for features or "support vectors" which are close to the hyper-lane for deciding a class and then it will decide as to which class the object belongs. It uses kernel-based approach for doing the separation which can be used in several real-life scenarios like text mining, fraud detection, insurance, chemistry, and bioinformatics (Savyanavar, 2023).

__Logistic Regression (LR):__ It is an algorithm for binary class where the result will be either "yes/no" or "0/1" and so on which uses logistic function also named as sigmoid function for translating predictions into probabilities (Savyanavar, 2023). Here, the model is initially trained with some parameters that makes the class of "0" and "1" after which the model is tested to a new value where it searches for a feature that can be used as a distinguishing point and then it classifies it to one of the classes.

__Decision Trees (DT):__ This algorithm makes tree-like structure from the dataset provided for training purpose where at each split, a feature is selected for performing the classification because of which when performing the splitting part there are chances that any given feature can be selected more than one time which sometimes results in giving biased decision. Thus, it is preferrable but has its own drawbacks as well.

__Convolutional Neural Networks (CNN):__ This algorithm provides highest level of accuracy amongst all the other ones as it trains from the data itself giving the desired result. Along with this, the model is made using layers and is compiled using specific types of optimizers which helps in classification purposes. Thereafter, it is tested on the dataset using "epochs" factor which determines the number of times model is being tested again, finally coming to the results providing best accuracy.

As a result, the paper suggests that to avoid overfitting problem in the model, numerous data augmentation techniques can be considered such as rotation, reflection and so on which can help in getting better accuracy. Moreover, the article even proposed several ML techniques results which made my vision clear regarding which model is performing under different performance metrics and that ultimately helped me in getting better prospects of those models from the table provided below:



Algorithm	Precision	Recall	F1-score	Accuracy
RF	99%	78%	87%	78%
KNN	98%	78%	87%	78%
DTC	80%	81%	81%	71%
NB	74%	83%	78%	68%
LR	88%	81%	84%	74%
SVM	99%	77%	87%	77%

Figure 8: Different ML techniques and their performance metrics score (Source: Savyanavar, 2023).

However, the paper was able to achieve the accuracy of 92.44% but was not able to go further as they used CNN for the classification of images which limited their accuracy till this point which can be improved in future by using other CNN models in the market (Savyanavar, 2023).

Thus, this article helped me in getting in-depth knowledge of different ML algorithms that can be used while doing classification of images out of which I chose CNN and RF as my prominent models for the same. Furthermore, it also provided key points that needs to consider when deciding which model to use according to different kinds of datasets.

3. Image Classification using CNN –

The article focuses on building a whole CNN model from scratch where they used dataset available from Sloan Digital Sky Survey (SDSS) and Canada-France-Hawaii Telescope Lensing Survey. Firstly, they have explained CNN in detail which helped me in getting familiarised with the convolutional and pooling layers which includes different types of activation function in them like Leaky ReLU, Softmax, Sigmoid and so on. Secondly, they also stated the data augmentation techniques they used along with detailed explanation of avoiding any overfitting or underfitting of data by using "Dropout" method (Kim and Brunner, 2016).

The results that they got suggested that ConvNet was quite reliable model for having better performance in image classification section and by addition of certain hyperparameters, it was able to achieve higher accuracy. Nonetheless, the model has been seen to have a little step back due to the problem of overfitting which they have mentioned in the paper as a drawback of this method and has suggested ways to overcome the problem as well (Kim and Brunner, 2016).

Moreover, I also included one of the major augmentation techniques after referring this paper which was rotation that benefited my model to be trained on different kinds of orientations of images due to which I was able to achieve such high accuracy.

4. Image Classification using RF –

The paper aims on solving the problem of classification by combining both CNN and RF models as well as explaining each of them in detail separately which gave me the adequate differentiation points between the two and the factors affecting it. Nevertheless, the part that helped me most was the thorough explanation of RF algorithm that it consists of "k" number of classification trees where tree consists of different kinds of nodes in which the features of the data are divided, and the root node is represented as the training set as well as each leaf node is labelled as training or testing that helps in classification of data into several subsets (Xi, 2022). Here, a noticeable feature comes into action which is termed as "Gini index" which can be defined as the optimal cut to determine that how well a split is made in a tree, and it also quantifies the probability of a new given input point to being wrongly classified with a given node. The formula for Gini index is given below:

$$G_{\text{gini}}(D) = 1 - \sum_{n=1}^{N} \left(\frac{|C_n|}{D} \right)^2,$$

Figure 9: Formula for Gini index (Source: Xi, 2022)

Here, "C*n*" stands for subset of samples in a dataset "D" that belongs to a particular class "*n*th" class. Furthermore, it has another prominent parameter of "n-estimators" which determines the number of splits the tree is going to make that means more the splits the better model will perform. As a result of which, in my model I chose 100 n-estimators as referred from paper itself along with implementation of method as discussed below and I was able to better my accuracy particularly.

Moreover, they have added another prominent part in building the model which is to combine CNN and RF classifier to create a hybrid model which had structure of layers as shown below:

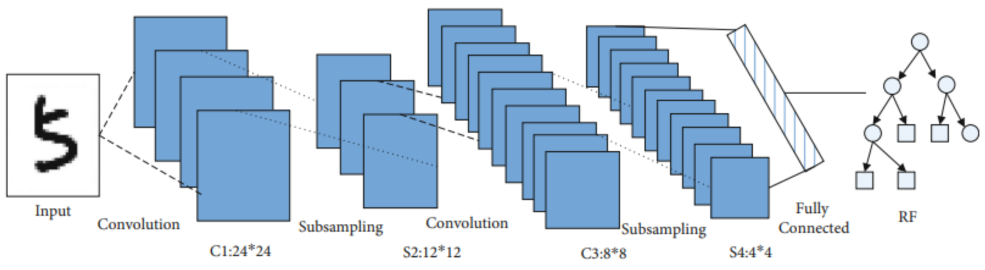


Figure 10: Hybrid CNN and RF classifier model combined (Source: Xi, 2022).

The model made the error rate bring down to 1.97% and due to this hybrid model building, they were able to solve the major issue of CNN model's taking less time in training it and solves the issue of manually selecting features for RF as well (Xi, 2022).

CHATPER-3: METHODOLOGY

3.1 OVERVIEW –

The prominent part of my project work is coding for which I decided to use CNN and RF models for comparing them and going in-depth for understanding working of them adequately. Also, the dataset that I have chosen is available on Kaggle website which consists of images of stars and galaxies in ratio of 76-26. Here, the CNN model includes methods of data collection, image pre-processing, model training and evaluation and interpretation & analysis part whereas RF model includes collecting dataset and pre-processing on that, splitting of data into training and testing parts, training the model and evaluation of it. Moreover, after performing these steps in both models I was able to achieve an accuracy of 93% in my CNN model and 72% in RF model.

3.2 DATASET DISCUSSION –

The dataset that I have chosen for performing my coding part has been taken from a research institute named as Aryabhatta Research Institute of Observational Sciences (ARIES) located in state Nainital, India. The cutout of images is labelled as stars and galaxies which makes it a supervised learning problem and were captured by the in-house telescope of size 1.3m from an observatory situated in Devasthal part of Nainital state, India. Initially the images were captured in size of 2k*x*2k format which was then reduced to size of 64*x*64 so that the images take minimal memory in the processing part. Moreover, image segmentation part was used for labelling the images in correct form as it was the further used for identifying the source of images and lastly, Sloan Digital Sky Survey (SDSS) was used as a query search tool for labelling the images by using centre point coordinates of those. Lastly, these cutout images were stored in different directories. The instance of stars and galaxy images is as below:

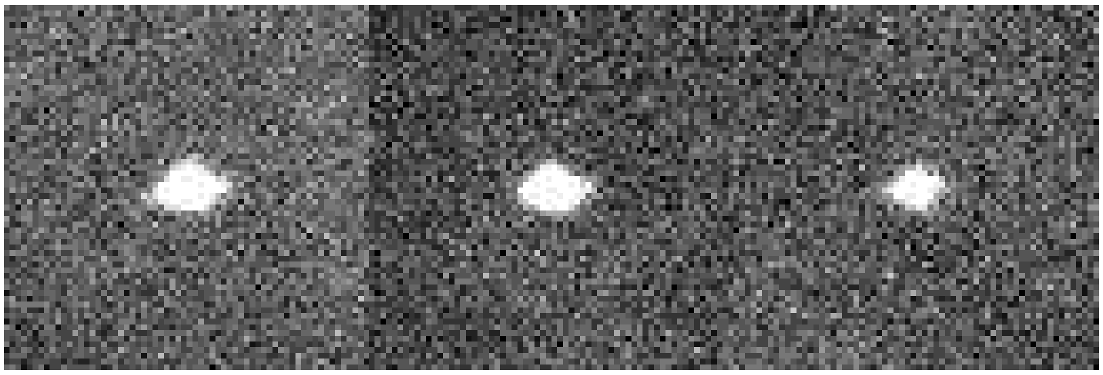


Figure 11: Images of Galaxies from dataset (Source: https://www.kaggle.com/datasets/divyansh22/dummy-astronomy-data)

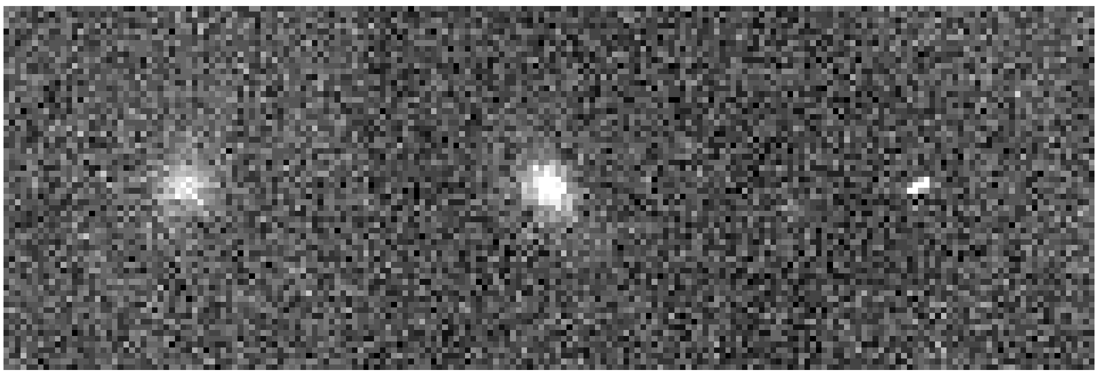


Figure 12: Images of stars from dataset (Source: https://www.kaggle.com/datasets/divyansh22/dummy-astronomy-data)

Moreover, there are in total 3986 images combined of which 942 of them are of galaxies and remaining 3044 are stars that means the images of stars are in majority which helps model to learn better and give the required output.

Here, another noticeable thing is that the dataset that I have selected is ethically approved which means that it is available publicly under the Licence of "CC0" which states that no copyright has been applied to any part of the dataset and anyone can copy, edit, distribute, and perform the work without asking for permission.

<u>3.3 IMAGE PRE-PROCESSING –</u>

<u>3.3.1 FEATURES AND AUGMENTATION –</u>

The images of stars and galaxies are of appropriate size in the dataset itself which is 64*64 pixels which makes it compatible with the whole processing part. Also, some of the major advantages of using this smaller size of images is that it helps in reducing the usage of memory while building the model, data augmentation part gets easier, it reduces overfitting as well because the images are quite small which helps the model to learn from the more detailed patterns and so on.

In my model building, I have used specific kinds of data augmentation techniques which has helped in increasing my accuracy and training the model more effectively, these are as follows:

1. **Rescaling –**

   This is one of the crucial steps while performing pre-processing of images as this helps in scaling the pixels of images for the model. Here, as we know that each image has its own pixels associated with it which determines its intensity and clarity as well. Usually, the images are either in RGB colour tone or in greyscale but in both of those cases the value of pixels ranges from 0-255 where each pixel is representing the colour tone of the image, but these values are quite high for the model to train efficiently due to this reason they need to be normalised by certain factors to be efficient enough for model training part.

Thus, the image pre-processing part of “1./255” comes into scenario where each pixel value is divided by highest value of pixel to make the range from 0-1 rather than 0-255 which is far more compatible with the model.

2. **Rotation –**

   The prominent aim of using this augmentation technique is to train the model with certain angle to recognize different patterns or orientations of images of stars and galaxies. The space comprises of numerous kinds of these objects which are in different angles as a reason of which I needed to train my model with angle rotation part as well so that it is able learn from various positions of images for giving better classification results.

   Here, I have after the trial-and-error method of choosing the correct angle for performing rotation technique I finally concluded that using the angle of 45-degree has made my model optimal in getting higher accuracy whereas other angles such as 90, 180 and so on were adversely affecting this accuracy factor and were making my model have higher loss function.

3. **Zoom Range –**

   This factor is typically used to zoom in and out of any image in the dataset to increase the diversity of the images which will help the model to train under various conditions and getting used to the unseen images as well.

   In order to have appropriate usage of this technique, I have used the zoom range value of “0.2” which has been quite ideal for my model as it has made the images either zoom in or out by 20% depending upon the structure of the images of stars or galaxies which adjust automatically zoom in to make them clearer and focusing on the details of the images making or zoom out for getting the proper structure of the objects and quite larger view making the model robust and compatible to different structures of images.

4. **Horizontal Flip –**

   This data augmentation technique is beneficial to model as it helps in learning different orientations of images by flipping them horizontally exposing them to turn left or right depending on the position they are originally. Also, the value of this function is set to “False” by default which means that we need to turn it in to “True” in case we want to apply this technique.

   Here, by using this technique my model has been able to achieve the desired accuracy as it has been trained on various inclinations and patterns of images making it more adaptable to different sets of images.

5. **Validation Split –**

   This is the final augmentation part that I have included while training my model with different data augmentation techniques, which has made the split for number of images that I will be using for my validation part and remaining of them in my training part. Moreover, to perform this splitting I have set the value to "0.2" which means that 20% part from my total images will be used for validating purposes and rest 80% will be used for training my model to have adequate distribution of images for so that overfitting can be reduced as the model will be able to learn from the trained images and will be validated on different images separately.

3.4 MODEL BUILDING –

3.4.1 CNN MODEL –

Convolutional Neural Network also known as CNN model is one of the most successful models across different other models available which are Decision Trees, Random Forest Classifier, Support Vector Machines, k-Nearest Neighbour and so on. This model has become quite popular in all domains of classification from astronomy to medical by providing highest ratio of accuracy and better classified images as final output. Furthermore, the best quality of this model is that it automatically learns from the features of the dataset provided which makes it robust giving better results as compared to other models where we must provide features manually making it less accurate.

Basically, CNN model compiles of different building blocks such as convolutional, max pooling and fully connected layers and it learns the features of the dataset through the method called as backpropagation algorithm (Yamashita et al., 2018). Here, the first two layers that are convolutional and max pooling works for feature extraction part whereas the last fully connected layer helps in drawing these features to the output as classification part. The working of a simple CNN model has been shown below in the picture:

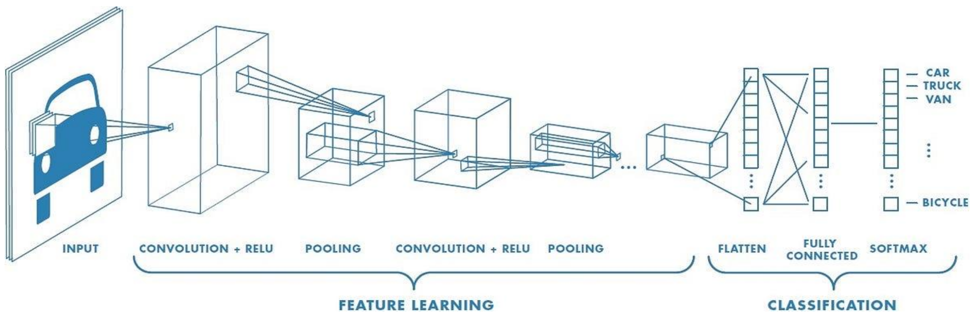


Figure 13: Working of CNN model with different layers attached together (Source: Convolutional Neural Networks, Explained. [online] Medium. Available at: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.)

In figure given above, there is a type of vehicle which is car that is used as an input for classification purposes which is used by convolutional and max pooling layers for extracting features like number of wheels, structure, size and so on based on which the classification happens at the where the output will be in form of either car, truck, van, bicycle and so on. Moreover, a CNN model layers consists of different types of blocks in them like activation functions and regularization parameter.

Now, looking the CNN model in more detailed manner starting with the Convolutional layer which is one of the basic components of the whole model which is used for extracting features from the dataset provided and it consists of two parts that are convolutional operation and activation function.

Firstly, convolutional is a type of operation which consists of array of numbers called kernels which are multiplied across the input which are another set of arrays of numbers termed as tensors where each value of input tensor is being mathematically calculated to kernel as element-wise product and the summed output of those is corresponded to the specific position of feature map. This procedure is repeated several times to form the whole feature map in similar manner. The prominent two hyperparameters of convolutional operation are size and the number of kernels (Yamashita et al., 2018).

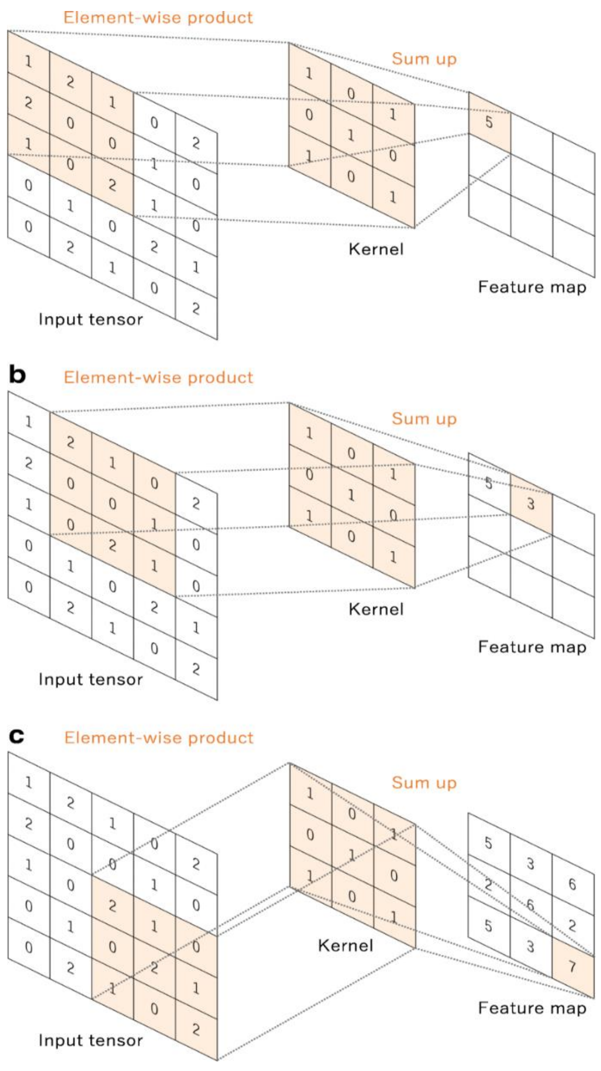


Figure 14: CNN model feature extraction process in various layers (Source: Yamashita et al., 2018).

Next, comes the non-linearity layers which consists of activation functions like sigmoid, tanh and ReLU in it which are crucial to CNN model as the first part which is convolution operation is a linear one which works in a definite manner only but the datasets might have some exceptions in them so in that scenario for the model to work appropriately, the concept of non-linearity came in existence which helps the model to learn the non-linear features from the dataset which are complex making model working smoothly under different conditions. The “sigmoid” function is used when the output is in probability meaning 0 or 1 as the curvature of it is in the same form as shown below:

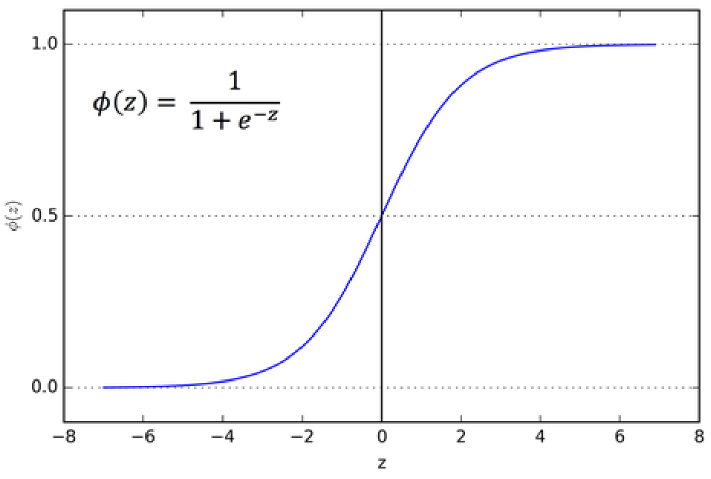


Figure 15: Curvature for sigmoid function (Source: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)

The “tanh” function has range from [-1, 1] making it much better than the sigmoid one as the negative inputs are mapped more strongly and the inputs which are near to zero will be mapped as zero which helps in better classification making it more differentiable (SHARMA, 2017). This function is majorly used for classification between two sets of classes and the curvature of it can be seen below:

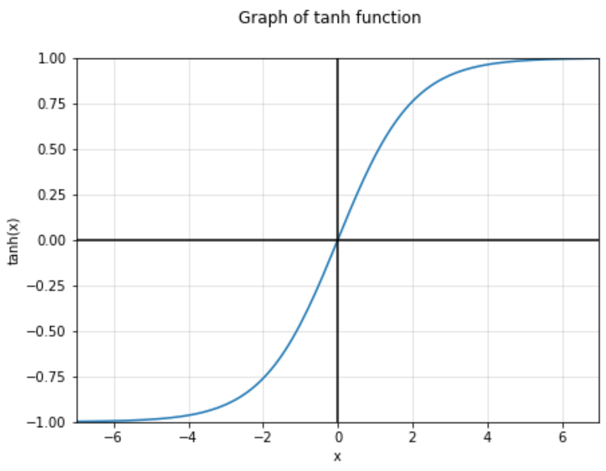


Figure 16: Curvature for tanh function (Source: https://www.researchgate.net/publication/340644173_A_Survey_on_Activation_Functions_and_their_relation_with_Xavier_and_He_Normal_Initialization/figures?lo=1)

Lastly there is "ReLU" function which is currently the most preferred one compared to sigmoid and tanh. It has range from [0, infinity) which makes it half rectified that means the value of the function becomes zero when it is below zero and gives the desired output when it is more than or equal to zero (SHARMA, 2017). The image given below shows that the function gives same kind of output when it is below the zero point and then increases relatively:

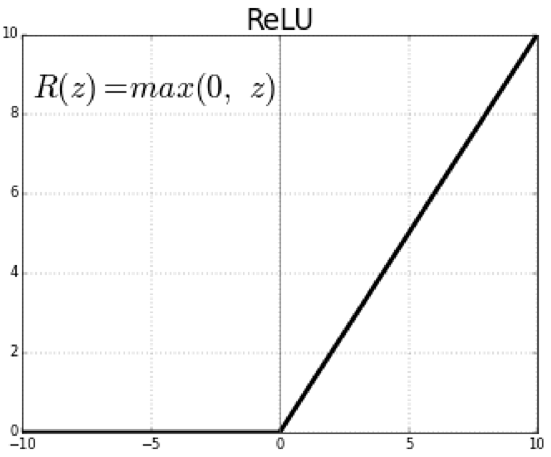


Figure 17: Curvature for ReLU function (Source: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)

Here, the issue with this activation function can be seen clearly as it is not able to perform effectively when it is going below zero which decreases the accuracy of the whole model so to tackle that situation, the concept of "Leaky ReLU" comes into play which has the range from (-infinity, infinity) which helps in increasing the criteria for performing better. An instance of comparison of both "ReLU" and "Leaky ReLU" can be found in image given below:

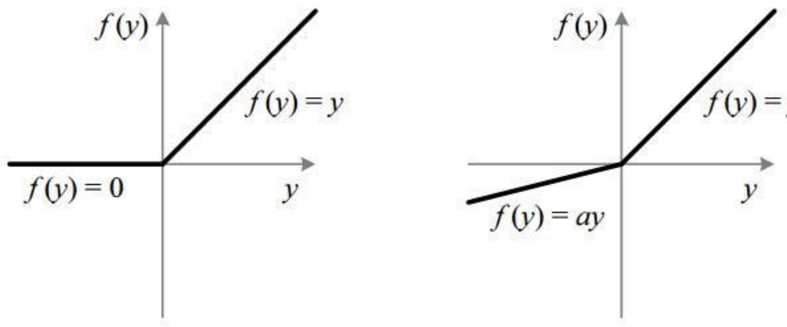


Figure 18: Curvature comparison for ReLU vs Leaky ReLU (Source: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)

Moving further in our CNN model, another prominent part that comes is of pooling layers which helps in replacing the output of the network at specific location by the summary statistic of nearby outputs that reduces the spatial size of representations and decreases the computation and weights of the code as well. In pooling layers parts, there comes number of variations available to perform that like rectangular neighbourhood, weighted average based on the distance from central pixel and L2 norm of the rectangular neighbourhood.

Nevertheless, most popular one is the max pooling one which takes the maximum value of the neighbourhood and draws it to the output matrix (Mishra, 2020). This process is shown in below picture:

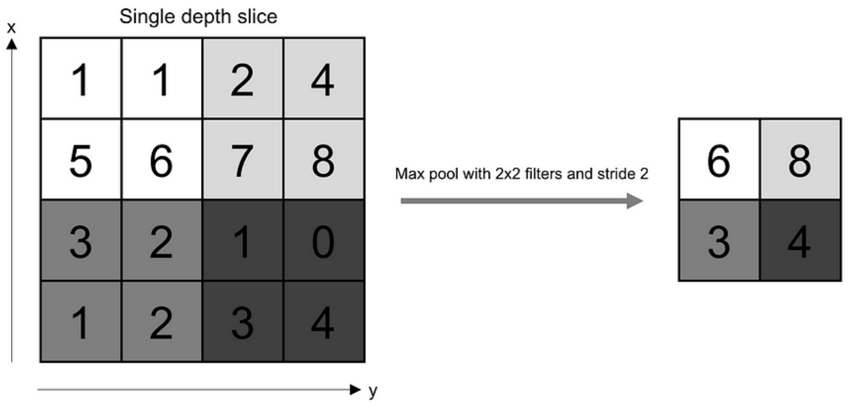
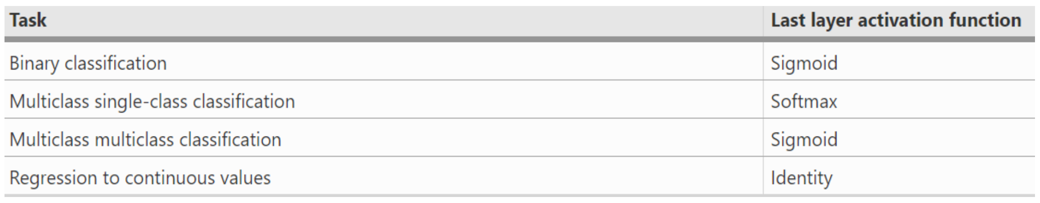


Figure 19: MaxPooling layer drawing towards final matrix (Source: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939)

Fully connected layers are the last part of whole layers topic, in this the output is typically flattened meaning it is in one-dimensional (1D) array of numbers and it is connected to one or more other fully connected layers, which is also known as dense layers where each input is connected to each output by some learnable weight (Yamashita et al., 2018). In this, the features that are extracted from the convolutional layers are down sampled by max pooling layers and then are projected to a subset of fully connected layers for the final output which is in probabilities of each class in classification part and the number of output nodes in this layer is same as the number of classes in the dataset (Yamashita et al., 2018). Moreover, a last layer activation function is also needs to be selected to perform conclusion part which is applied to the last fully connected layer, and it changes in accordance with every task. Basically, the "softmax" function is used in most cases where there is classification problem involved which normalizes the output values from the last fully connected layer and draws the conclusions towards the target class probabilities (Yamashita et al., 2018). The choices of function depending upon the problems are listed below:



Task	Last layer activation function
Binary classification	Sigmoid
Multiclass single-class classification	Softmax
Multiclass multiclass classification	Sigmoid
Regression to continuous values	Identity

Figure 20: Different classification tasks along with their suitable activation function (Source: Yamashita et al., 2018).

<u>**3.4.1.1 CNN LAYERS –**</u>

To commence with layers part in project work, it is evitable to have the input shape pre-defined so that we can directly include it in our layers. So, for this purpose, I chose my input shape to be “input_shape = (star_galaxy_image_size[0], star_galaxy_image_size[1], 3)” which specifies how many elements an array and tensor has in each dimension. Here “star_galaxy_image_size[0]” and “star_galaxy_image_size[1]” represents the dimensions of input images where [0] depicts the number of rows or height of the images and [1] represents the number of columns or width of the images which ultimately determine the number of pixels that are in each image. This is important because it is thereafter used for feature extraction part in following layers. Also, number “3” at last represents the colour coding involved which is RGB in my input shape as it has 3 channels which are stacked together forming a 3-dimensional (3D) image. The importance of this input shape is that it determines the dimensions like height, width and colour channels for model’s input layer while performing the processing of images. The layers that I have built for my model are shown in below image:

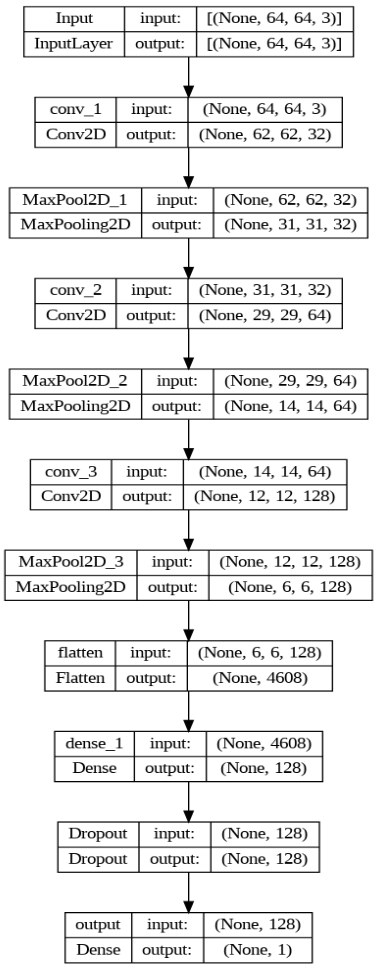
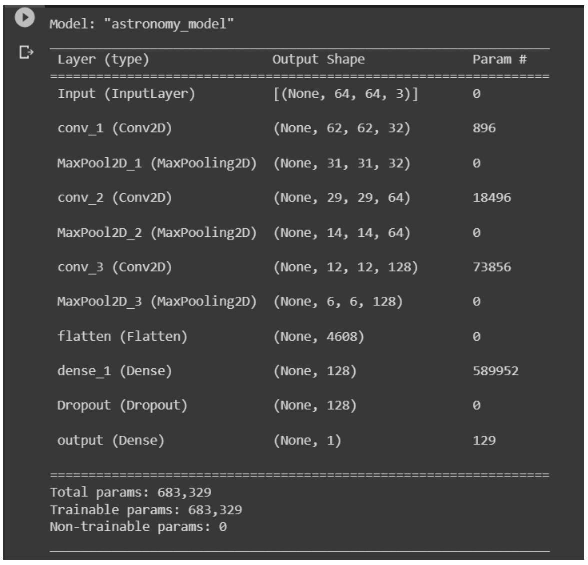

Input | InputLayer | input: [(None, 64, 64, 3)] | output: [(None, 64, 64, 3)]
conv_1 | Conv2D | input: (None, 64, 64, 3) | output: (None, 62, 62, 32)
MaxPool2D_1 | MaxPooling2D | input: (None, 62, 62, 32) | output: (None, 31, 31, 32)
conv_2 | Conv2D | input: (None, 31, 31, 32) | output: (None, 29, 29, 64)
MaxPool2D_2 | MaxPooling2D | input: (None, 29, 29, 64) | output: (None, 14, 14, 64)
conv_3 | Conv2D | input: (None, 14, 14, 64) | output: (None, 12, 12, 128)
MaxPool2D_3 | MaxPooling2D | input: (None, 12, 12, 128) | output: (None, 6, 6, 128)
flatten | Flatten | input: (None, 6, 6, 128) | output: (None, 4608)
dense_1 | Dense | input: (None, 4608) | output: (None, 128)
Dropout | Dropout | input: (None, 128) | output: (None, 128)
output | Dense | input: (None, 128) | output: (None, 1)

Figure 21: CNN model layers along with size of them.



```
Model: "astronomy_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (InputLayer)          [(None, 64, 64, 3)]       0

 conv_1 (Conv2D)             (None, 62, 62, 32)        896

 MaxPool2D_1 (MaxPooling2D)  (None, 31, 31, 32)        0

 conv_2 (Conv2D)             (None, 29, 29, 64)        18496

 MaxPool2D_2 (MaxPooling2D)  (None, 14, 14, 64)        0

 conv_3 (Conv2D)             (None, 12, 12, 128)       73856

 MaxPool2D_3 (MaxPooling2D)  (None, 6, 6, 128)         0

 flatten (Flatten)           (None, 4608)              0

 dense_1 (Dense)             (None, 128)               589952

 Dropout (Dropout)           (None, 128)               0

 output (Dense)              (None, 1)                 129

=================================================================
Total params: 683,329
Trainable params: 683,329
Non-trainable params: 0
_________________________________________________________________
```

Figure 22: Summary of CNN model.

The input layer is of size 64*64 with 3 colour channels associated with it after which comes the first convolution layer termed as “conv_1” which has 32 filters also called as feature maps or channels in it with each of size 3*3 and “relu” activation function for performing non-linearity part. This is followed by the corresponding max pooling layer which has pool size of 2*2 reducing the spatial dimension of the data which helps in retaining the important features and down sampling the whole representation. In a similar way, the model is built till the convolutional layer reaches the 128 *128 size and then I have added another max pooling layer names as “flatten” to convert the whole 2D spatial to 1D vector for connecting the convolutional layers to fully connected layers. In the next step, I have included “dense_1” as my fully connected layer which has 128 neurons in it with activation function “relu” and lastly, adding a regularization layer as well to overcome the problem of overfitting in which the model might not be able to train on smaller details and takes major features only for classification purpose which decreases the overall capability of the model and makes it less accurate. Thus, for the same reason the concept of regularization was introduced to avoid such overfitting problem while training the model. Here, I have used “dropout” regularizer with value “0.5” which means that whenever overfitting will occur, the model will dropout half of the features to take into consideration the important ones only making model more accurate. Apart from this, there are other regularization techniques as well like L1, L2 and Elastic Net.

Lastly, I have had my “output” layer with only “1” neuron with activation function of “sigmoid” which is preferably used for binary classification purposes and is quite appropriate to my model as I needed to classify stars and galaxy images in general.

<u>3.4.1.2 COMPILE CNN MODEL –</u>

The part of model compilation includes optimizers, loss function, metrics and training the model appropriately which has certain parameters involved in it. Firstly, understanding the optimizers which is a prominent parameter in compilation of my CNN model. So, the optimizers are used to make model more optimized, and the process is called optimization where optimizers are algorithms which is used to change the attributes of the model like weights and learning rate that is beneficial in reducing loss of CNN model (Doshi, 2020). There are several types of most preferred optimizers available to include in the model as per the requirements that are Stochastic Gradient Descent, Adagrad, AdaDelta and Adam.

Stochastic Gradient Descent (SGD) is a variant of the above explained optimizer which tries to update the parameters of model quite frequently that is it will change the model parameter the number of times depending on the number of rows involved which is quite different from the Gradient Descent one which just updates the parameters ones in the whole cycle. Also, due to this there are high chances of fluctuations in loss functions as well which makes it less usable but as it frequently updates itself it takes lesser time in converging which gives it the benefit of getting to minima earlier (Doshi, 2020).

Mini-Batch Gradient Descent is preferred to be one of the best as compared to Gradient Descent and SGD as it updates itself after every batch which reduces the chances of high variance and fluctuations and requires lesser memory.

Adagrad is a type of optimizer that is different from all other ones as it changes the learning rate for every step which makes it second order optimizer and it works on the derivative of the error function (Doshi, 2020). In this, the optimizer stores the sum of the square of the gradients where gradients are calculated up to time “t”.

AdaDelta is an extension of Adagrad as in this it tends to remove the problem of decaying Learning Rate where instead of gathering all the previously calculated square gradients it limits the window the accumulated past gradients to a specific size “w” so because of which rather than the sum, the moving average value is used for calculation purposes (Doshi, 2020).

Adam (Adaptive Momentum Estimation) is one of the best optimizers as compared to all other available ones as the aim of this is that it wants to limit the velocity so that it does not jump over the minimum too fast, and it also stores the values of exponentially decaying average of past gradients as well (Doshi, 2020).

Thus, I have used “Adam” optimizer for compiling my model effectively along with loss function as “binary_crossentropy” one. Here, while performing binary classification of stars and galaxies there are three major types of loss functions that are useful in this scenario which are “Binary Cross-Entropy”, “Hinge Loss” and “Squared Hinge Loss”.

Binary Cross-Entropy Loss function is used for classification of binary classes where the labels have two classes and the target values are in set of {0, 1}. In this, the loss function is calculated first, and it is changed only if there is a good chance of it in another calculations. It calculates the score that summarizes the average difference between the actual and predicted probability distributions for the prediction of class 1. Furthermore, the score is minimized and as an ideal cross-entropy value is 0 (Brownlee, 2019).

Hinge Loss is used for Support Vector Machine models where the target value is in set of {-1, 1}> It introduces a penalty when the prediction is quite close to the correct label and beyond the margin of 1. Here, the loss is calculated as 0 when the prediction is correct, and it increases relatively if the prediction is going far from class label.

Squared Hinge Loss is an extension of the “hinge loss” function which calculates the square of the score hinge loss and has the effect of smoothening the surface of the error function (Brownlee, 2019).

Here, out of all the loss functions of binary classification problem, “binary_crossentropy” has been proven to be the best and that is the reason I chose that in my model building.

Lastly, I have used “accuracy” as my model’s metrics to calculate the accuracy of my model for evaluation purposes. Thereafter, I have ran it on 10 number of epochs meaning that my code will be iterated 10 number of times in one cycle.

A noticeable thing in this my model was the “learning rate” parameter which I tried to modify according to model requirements as a result of which I started with the value of “0.1” and then started gradually decreasing it by the factor of 0.05 but due to this trial-and-error method, my model was giving an unexpectedly higher value of validation loss due to which the accuracy was falling down to 65%. Thus, to avoid that problem, I decided to choose the default learning rate for my model, as in that case I was able to minimise the validation loss and increase my accuracy to 91%.

3.4.1.3 CNN MODEL EVALUATION –

The CNN model that I built has given me accuracy of 91% with loss of 19% and from the graphs below it is quite predictable that the model has been having exceptions in in middle due to which it tends to fluctuate its validation part but at the same time it has been seen to follow a pattern of curvature giving the desired accuracy at last and minimising the loss function as well.

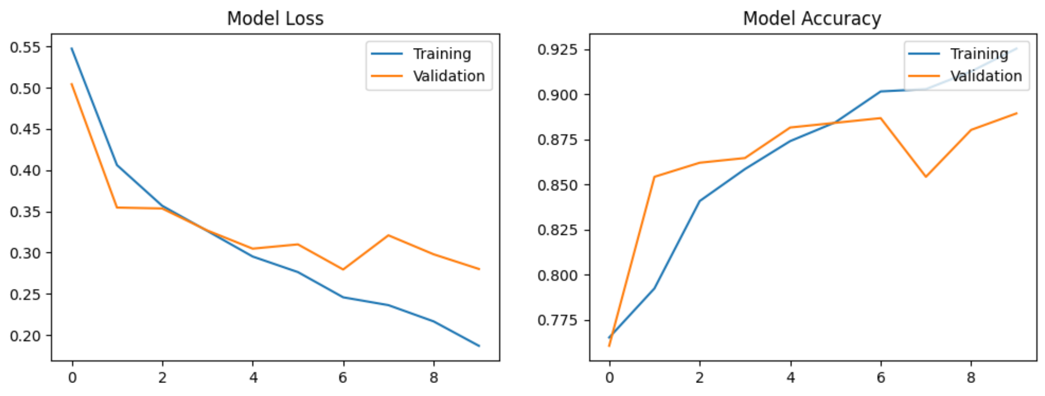


Figure 23: CNN model loss and accuracy curves.

3.4.1.4 CNN MODEL IMAGE CLASSIFICATION –

Here, I have created subplots of 5*5 where I have chosen the threshold value to be "0.5" for predicting the probability of either stars or galaxies and including labels on top of that as well in similar manner. Thus, from the image given below it is quite evitable that the CNN model that I built was able to correctly classify most of the images whereas there are still some of the images that were predicted incorrectly due to which the loss part reached till 19% making the accuracy to 91% which is far better than any other model.

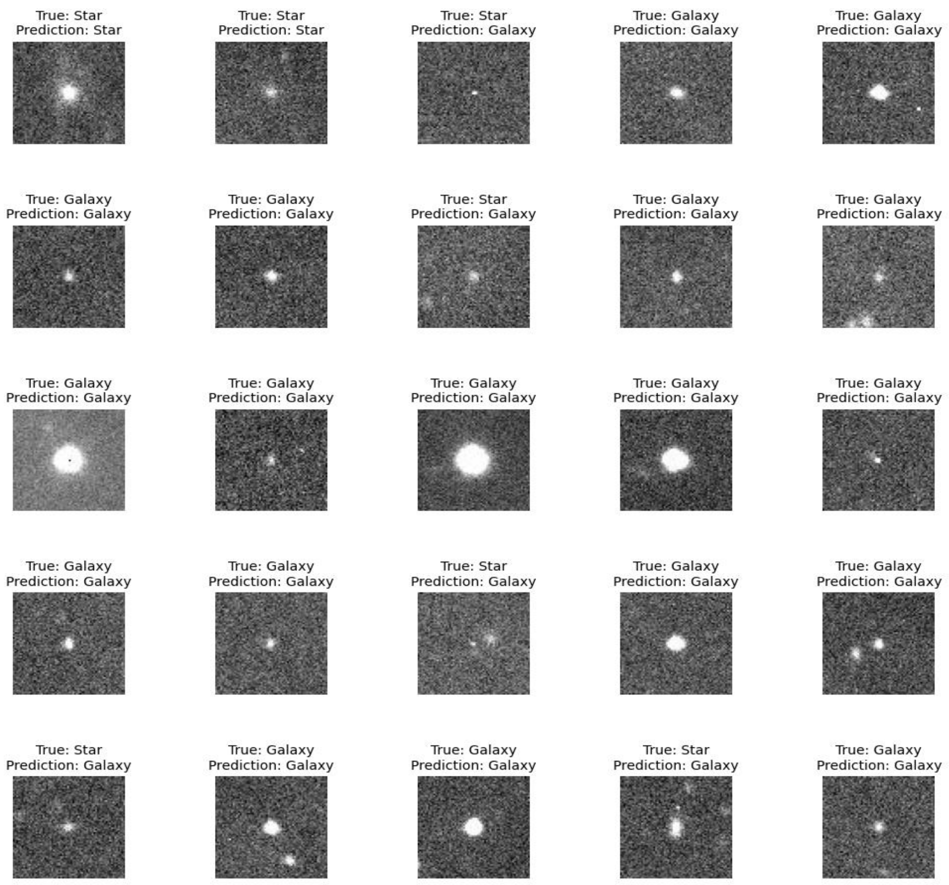


Figure 24: Subplots of correctly classified images along with labels on top.

<u>3.4.2 RANDOM FOREST CLASSIFIER MODEL –</u>

It is one of the types of models used for classification purposes and is widely used as well as preferred after CNN model. Random Forest Classifier was proposed by Leo Breiman in 2000's for building a predictor with sets of decision trees . It can be defined as a classifier consisting of a collection of trees-structured classifiers which have independent identically distributed random vectors, and each tree casts its vote for the popular class at its node (Kulkarni, 2013). A simple representation of its working is shown below:

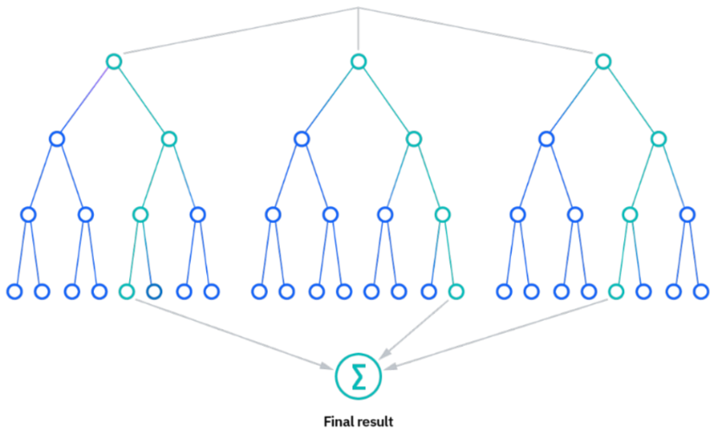


Figure 25: Random Forest Classifier tree-structure for classification (Source: https://www.ibm.com/topics/random-forest)

There are three prominent hyperparameters that needs to be set which are node size, number of trees and number of features. The fundamental of random forest classifier is that it is many relatively uncorrelated trees which are operating as a group will outperform any individual constituent models. Thus, to ensure that the behaviours of tree are not correlated, it uses two techniques that is Bagging and Feature randomness (Yiu, 2019).

Here, Bagging (Bootstrap Aggregation) can be defined as building different decision trees with same sample size but allowing randomly sampling from the dataset with replacement which will result in different trees.

Next is Feature Randomness, the random forest must choose a certain kind of feature only from a given subset of random features which makes more diversified and reduces correlation in between trees as variation will increase in each tree with different sets of features available (Yiu, 2019). An instance of it is shown in image below:

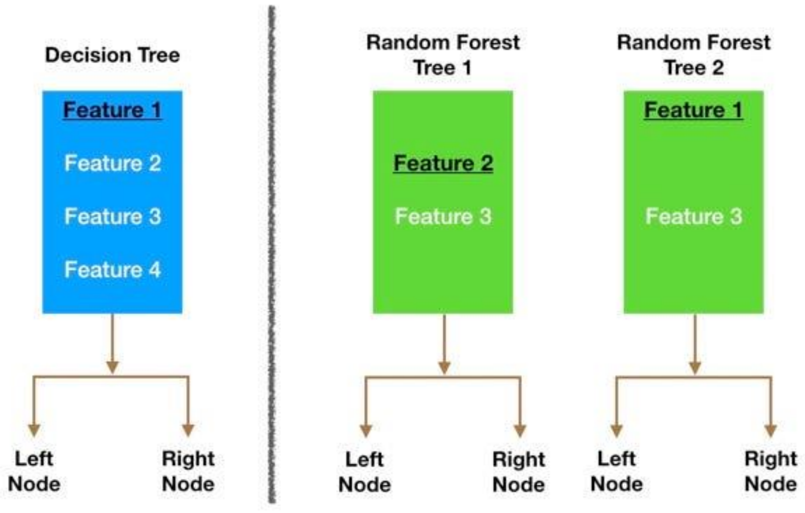


Figure 26: Random feature selection for RF classifier (Source: Yiu, 2019).

Another interesting area in Random Forest Classifier is of hyperparameters which are used for increasing the predictiveness and model's speed. Moreover, for elevating the performance of the model the first parameter is termed as "n_estimators" which means that the number of trees that will be built or taken into consideration for casting their votes for predicting the class of the object. The relativeness of choosing higher number of trees is directly proportional to better predictions. Another parameter is "max_features" which is the maximum number of features the random forest is considering for splitting a node and, there is "max_sample_leaf" which means that maximum number of leafs required to split the node (builtin.com, n.d.).

There are another 3 kinds of parameters involved for increasing the speed of the random forest model which are "n_jobs" that specifies the number of processors the tree is allowed to use. Here, if the value is set to "1" then the random forest can use only 1 processor and if the value is set to "-1" then the tree can use as many processors as possible it wants without any limit. Next is "random_state" which makes the model's output replicable given the same initial configuration. This feature is quite essential for consistency of the model as if we do not use this feature, then the model might run differently each time which can result in increasing loss part. Lastly, there is “oob_score” which are called as out-of-bag samples where one-third of the data is not used from training dataset and can be used further to evaluate the performance of the model and this is also called as cross-validation method (builtin.com, n.d.).

3.4.2.1 RF CLASSIFIER LOAD DATA –

Random Forest Classifier code starts from loading the dataset into the system which happens by creating the function named as “load_image_data”. In this, I have taken two empty parameters into consideration which are lists of “X” for feature vectors and “y” for class labels of my stars and galaxy images. Thereafter, the data is loaded into the code by creating class labels for stars and galaxies and then giving the path to my dataset directory for loading the images into my random forest model.

3.4.2.2 COMPILE RF CLASSIFIER MODEL –

The next step that comes after performing the dataset loading part is compiling the model for which I created four variables named as “X_train”, “X_test”, “y_train” and “y_test” which has “test_size=0.2” that means the training part has 80% of the images whereas the testing part has 20% of the remaining images. Thus, the model will be tested on these 20% of images and will give the results in form of classification. Here, “random_state” is set to 42 which is just a random number used for purpose to seed the random forest model with the surety of reproducibility and any other number can be set instead of 42 as it is does not have any specific significance in general.

Here, I have chosen “n_estimators” to be set on 100 as the random forest will consist of 100 individual trees which will be beneficial in performing the model more accurately. Moreover, I have made model to be fitted on the training data by performing “rf_classifier.fit(X_train, y_train)” and it will be tested on the remaining 20% by “y_pred = rf_classifier.predict(X_test)” where predictions will be made on the testing dataset part.

3.4.2.3 RF CLASSIFIER MODEL EVALUATION –

The RF classifier model that I built from scratch has given me the accuracy of 75% overall with loss of 25%. Firstly, I have plotted the graph for accuracy that I got in my RF model so that I can get an insight into the model more precisely as in the graph, I have tried to use intervals of “n_estimators” so that I was able to understand that after which point the results has got better, which can be seen in the image that I have attached below:

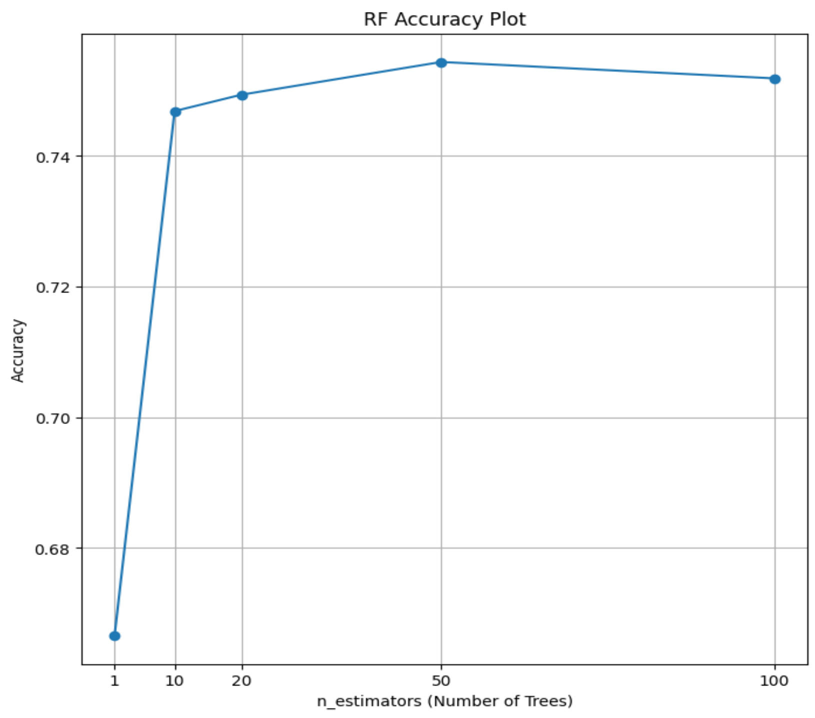


Figure 27: Accuracy graph for RF classifier at different intervals of n_estimators.

Here, initially the accuracy of my RF model till 20 n_estimators was near to 74% which got increase and reached to its peak at 50th one where the accuracy came to 75% and has remained steady after that till the 100th n_estimator.

Also, I have plotted the correctly classified images as well to get a look that where my model failed to perform accurately, and the results can be seen in image attached below. Moreover, there are quite a few instances where the model predicted “star” and the image was of “galaxy” which shows a failure in feature extraction part where my CNN model quite performed in a better way. Nevertheless, the RF model was still able to correctly classify most of the images based on the parameters that I inferred into the model.

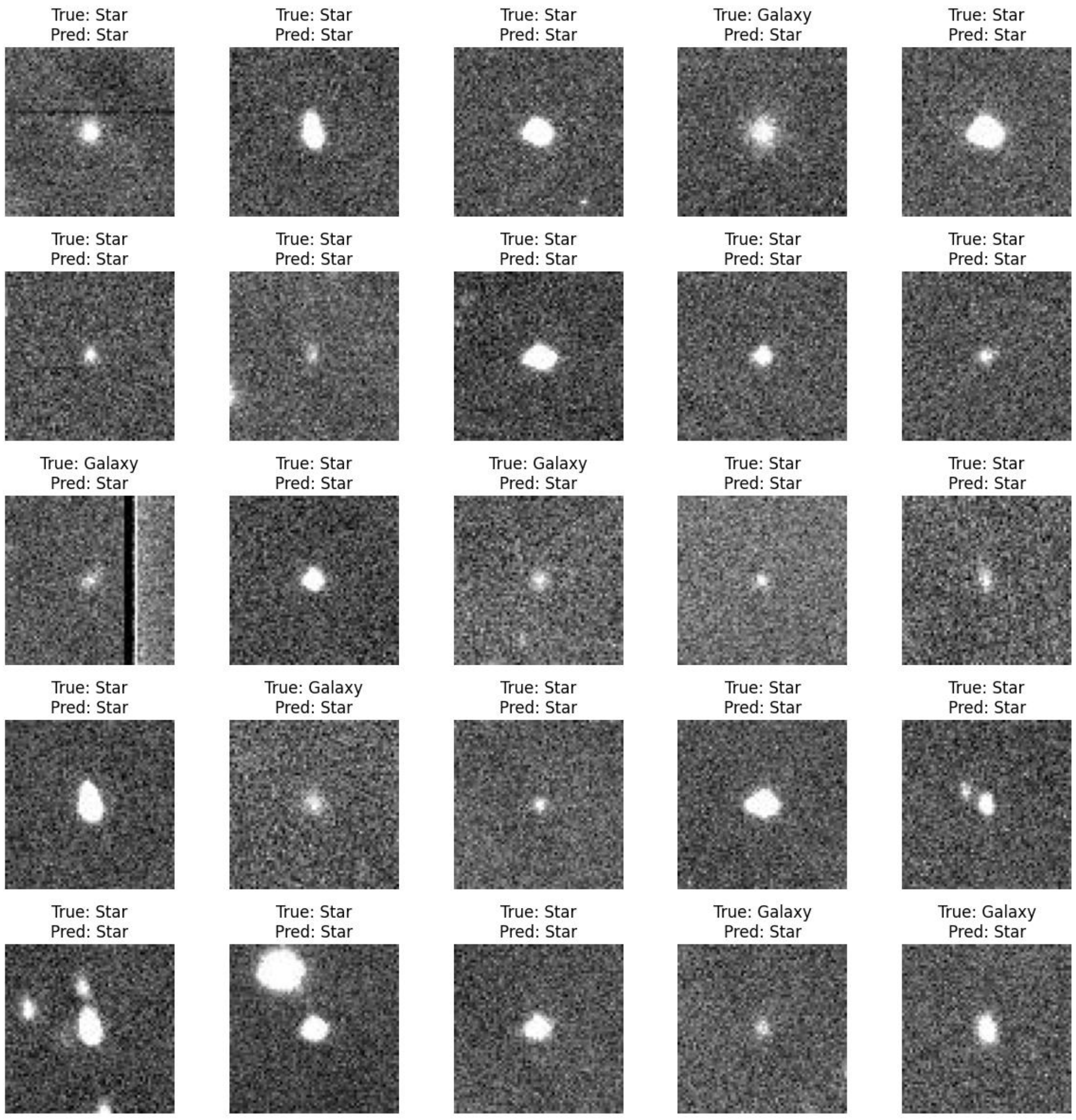
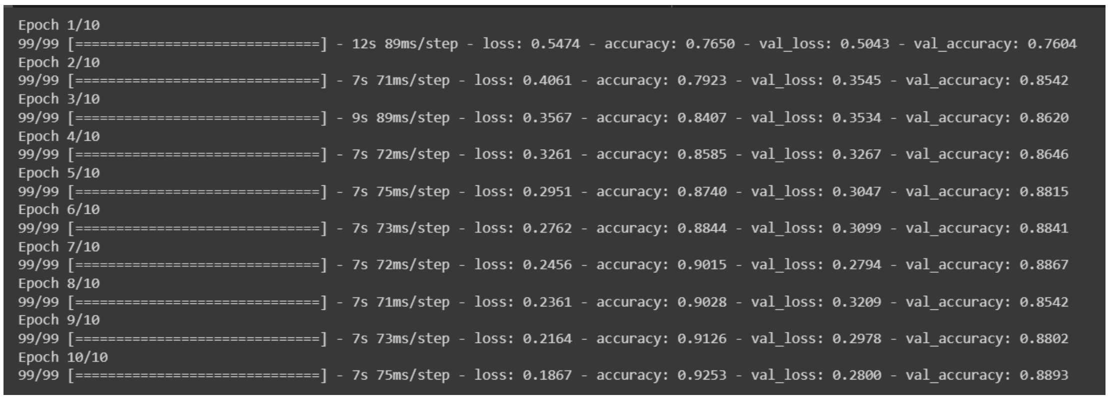


Figure 28: Correctly classified images by RF classifier along with labels on top of them.

CHAPTER-4: RESULTS AND ANALYSIS

4.1 CNN MODEL RESULTS –

The model that I made from scratch gave me an accuracy of 94% with loss of 16% when I ran it on 10 number of epochs, an image of which is attached below:

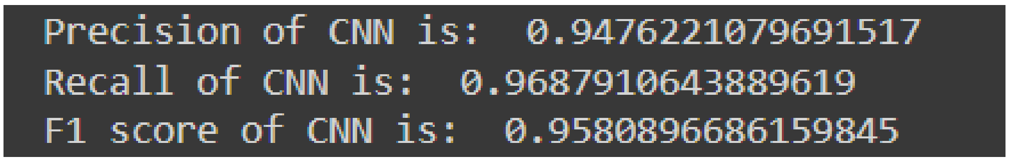


```
Epoch 1/10
99/99 [==============================] - 12s 89ms/step - loss: 0.5474 - accuracy: 0.7650 - val_loss: 0.5043 - val_accuracy: 0.7604
Epoch 2/10
99/99 [==============================] - 7s 71ms/step - loss: 0.4061 - accuracy: 0.7923 - val_loss: 0.3545 - val_accuracy: 0.8542
Epoch 3/10
99/99 [==============================] - 9s 89ms/step - loss: 0.3567 - accuracy: 0.8407 - val_loss: 0.3534 - val_accuracy: 0.8620
Epoch 4/10
99/99 [==============================] - 7s 72ms/step - loss: 0.3261 - accuracy: 0.8585 - val_loss: 0.3267 - val_accuracy: 0.8646
Epoch 5/10
99/99 [==============================] - 7s 75ms/step - loss: 0.2951 - accuracy: 0.8740 - val_loss: 0.3047 - val_accuracy: 0.8815
Epoch 6/10
99/99 [==============================] - 7s 73ms/step - loss: 0.2762 - accuracy: 0.8844 - val_loss: 0.3099 - val_accuracy: 0.8841
Epoch 7/10
99/99 [==============================] - 7s 72ms/step - loss: 0.2456 - accuracy: 0.9015 - val_loss: 0.2794 - val_accuracy: 0.8867
Epoch 8/10
99/99 [==============================] - 7s 71ms/step - loss: 0.2361 - accuracy: 0.9028 - val_loss: 0.3209 - val_accuracy: 0.8542
Epoch 9/10
99/99 [==============================] - 7s 73ms/step - loss: 0.2164 - accuracy: 0.9126 - val_loss: 0.2978 - val_accuracy: 0.8802
Epoch 10/10
99/99 [==============================] - 7s 75ms/step - loss: 0.1867 - accuracy: 0.9253 - val_loss: 0.2800 - val_accuracy: 0.8893
```

Figure 29: 10 epochs run for CNN model.

Here, it is observed that the loss function is fluctuating a little in between epoch number 5 and 6 whereas it is seen to be decreasing otherwise in every epoch run. Initially, it started from 0.5043 with accuracy of 0.7650 and at last it reached to 0.2800 with accuracy of 0.9253 along with loss of 0.1867. Moreover, another noticeable point is that the model loss is continuously decreasing throughout the entire run which is making model more accurate towards the results.

Next, coming to different performance matrix parameters that I performed for deep evaluation in my CNN model such as “confusion matrix”, “precision”, “recall”, “F1 score” and “ROC curve”. Firstly, I displayed results of precision, recall and F1 score which are as shown in below given image:



```
Precision of CNN is:  0.9476221079691517
Recall of CNN is:  0.9687910643889619
F1 score of CNN is:  0.9580896686159845
```

Figure 30: Performance metrics that are precision, recall and f1-score of CNN model.

The base for these three parameters is formed from four terms which are True Positive, True Negative, False Positive and False Negative where True Positive refers to the scenario when the prediction was true as compared to the kind of image, True Negative means that the classification of image was wrongly predicted like the image was star and not galaxy and the image was also of a galaxy only, False Positive means that the prediction of classification was done incorrectly like the image was not of galaxy and predicted galaxy and lastly, False Negative refers that the image was not of a star but it predicted star instead.

Precision can be defined as the measure of true positives which means that the number of predictions that were made are correct in nature as well. The formula to calculate precision can be found below:

$$\frac{\text{True Positives}}{\text{True Positives + False Positives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Predictions you Made}}$$

Figure 31: Precision performance metrics formula.

Recall is the measure of correct predictions made over all the other positive cases that were there in the scenario. The formula is given below:

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Instances in the Dataset}}$$

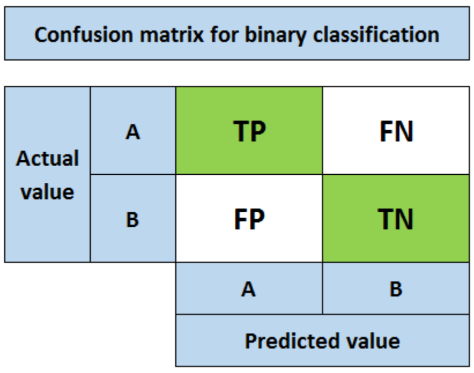
Figure 32: Recall performance metrics formula.

F1 Score is technically the combination of precision and recall meaning that it is harmonic mean of the given two parameters defined which is nothing but the average to provide single metric for evaluation. It can be calculated as follows:

$$2 * \frac{\text{Precision * Recall}}{\text{Precision + Recall}}$$

Figure 33: F1-Score performance metrics formula.

Next, coming to Confusion Matrix parameter which is a matrix for classification of the values in form of True Positive, True Negative, False Positive and False Negative which can be seen below:



Confusion matrix for binary classification

Actual value	A	TP	FN
	B	FP	TN
		A	B
		Predicted value	

Figure 34: layout of confusion matrix for binary classification.

Here, there are four sections where it gives the detailed evaluation of model's predictions on and the confusion matrix that I plotted as follows:

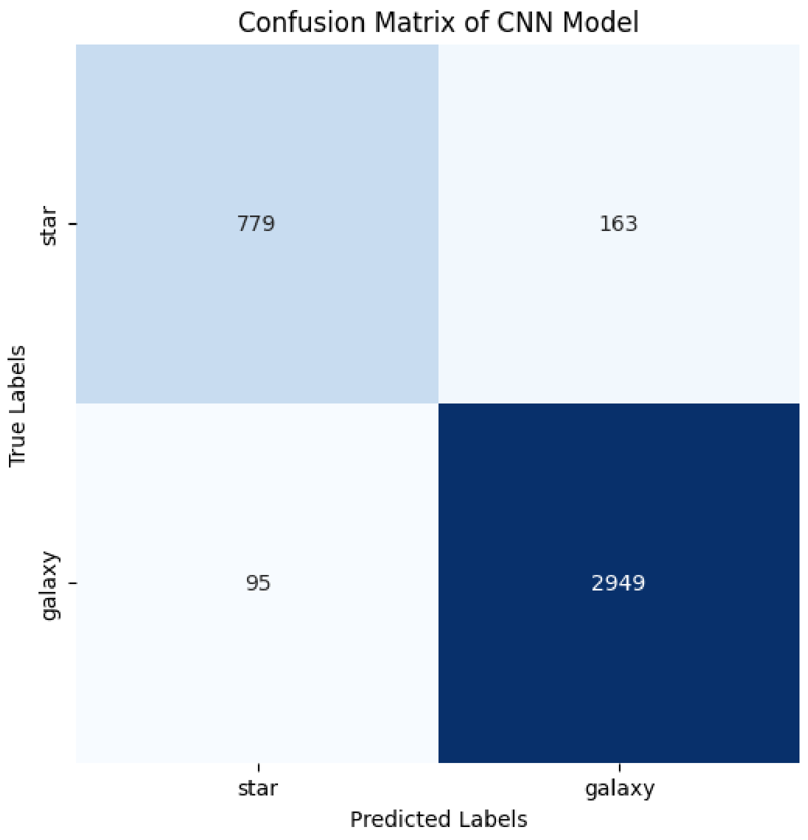


Figure 35: CNN model confusion matrix with labels.

The image attached above interprets that 779 of the images that were correctly classified as predicted that means that they were predicted as "stars" and the images were of "stars" only that comes in category of "True Positive", then comes the figure of 2949 which is under the category of "True Negative" where the images were of "galaxies" and the model was able to correctly predict that it was not "stars". Furthermore, other squares given are of "False Negative" which has the figure of 163 meaning that the prediction was of "galaxy", but it was an image of "star" actually and lastly, it is "False Positive" where there are 95 images that were initially was of "galaxy" but were incorrectly classified as "stars". This is the total in detail breakdown of 3986 images that I had in my dataset and were used for classification purposes.

Lastly, there is another parameter which I included in my model results analysis part which is "AUC-ROC Curve" where "ROC" stands for "Receiver Operator Characteristic" that depicts the accuracy of the model in classification case. It is a curve of probability between "True Positive Rate" and "False Positive Rate", and it helps in looking at the different levels of certainty model makes while coming to a decision. Here, "AUC" means the "Area Under the Curve" which interprets the ability of the model to distinguish between stars and galaxies and acts as a summary of ROC Curve. There are different cases of AUC like the first one is when "AUC=1" which is an ideal scenario where the model performs perfectly and can classify images correctly which can be seen below:

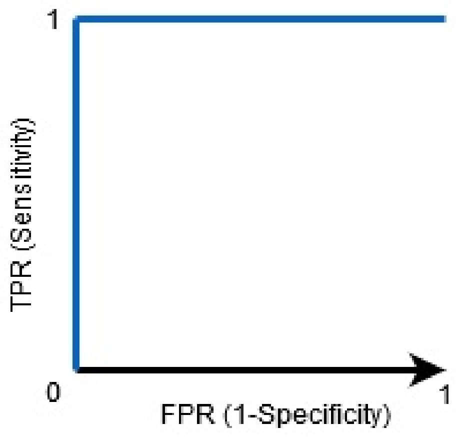


Figure 36: AUC curve under condition "AUC=1".

Secondly, when the case is of “0.5<AUC<1” at that time the chances of getting better accuracy and classification are higher as the model will be able to understand a difference between the positives and the negatives easily. An instance of it is shown below:

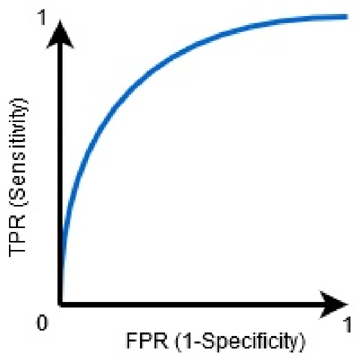


Figure 37: AUC curve under condition "0.5<AUC<1".

The last case is of “AUC=0.5” which is when the model will fail to classify the positives from the negatives and will give lowest accuracy and high ratio of loss function which will make the model perform abruptly. Thus, it can be concluded that better the AUC value for a model, better will be the accuracy. The graph of the last case can be seen as follows:

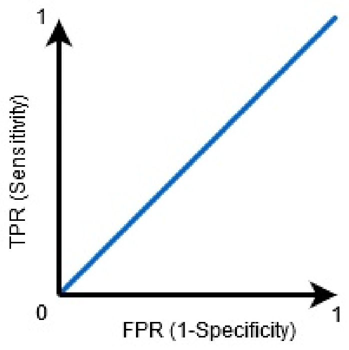


Figure 38: AUC Curve under condition "AUC=0.5".

Now, coming to the ROC Curve that I got for my CNN model where the red line suggests the general scenario that I discussed above, and the blue curve is my model's performance curve which depicts that the model is performing well as the value is near 1 for it making the model more accurate by giving proper classification of stars and galaxies. The graph of my ROC Curve can be found below:

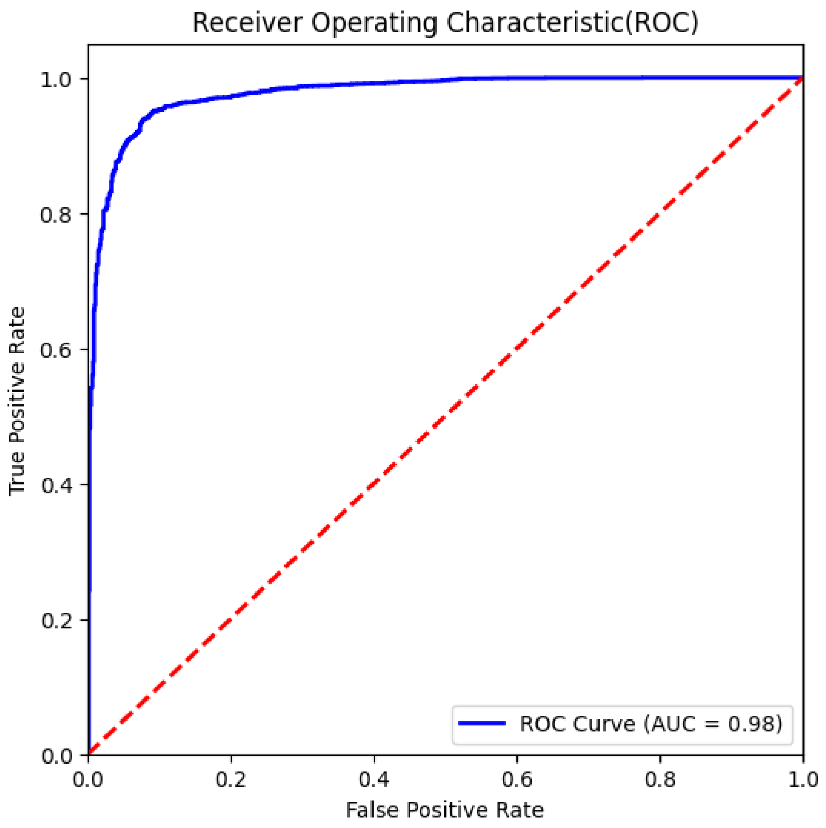
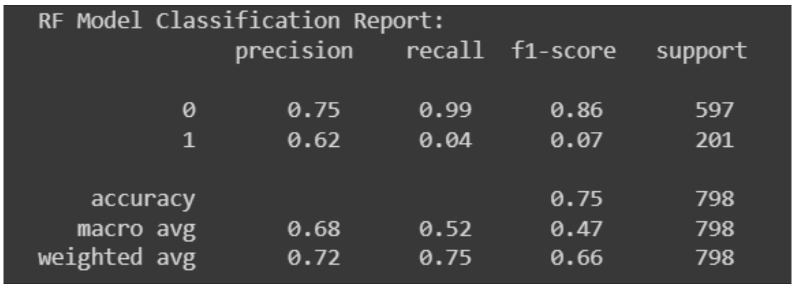


Figure 39: ROC AUC Curve for CNN model giving result of 0.98.

To conclude, the CNN model that I made was able to give the accuracy of 92% along with loss of 18% in which I tried to perform different performance matrices to gain in-depth knowledge of my model's working and I was able to achieve that by trying several techniques and learning from them.

<u>4.2 RF CLASSIFIER MODEL RESULTS –</u>

Another performance matrices that I have included for evaluation purposes are precision, recall, F1 score and support as they are categorized under the "classification report" part and have plotted confusion matrix as well so that I can have appropriate comparison between CNN and RF model that I have built. A snapshot of the results of my classification report is as follows:



```
RF Model Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.99      0.86       597
           1       0.62      0.04      0.07       201

    accuracy                           0.75       798
   macro avg       0.68      0.52      0.47       798
weighted avg       0.72      0.75      0.66       798
```

Figure 40: Performance metrics like precision, recall, f1-score, and support for RD classifier model along with macro and weighted avg.

The image above shows the figures of different performance matrices in two different classes which are “0” and “1”. Here, it is evitable that the model was able to classify correct positive classes quite accurately and the harmonic mean of correctly classifying individually as well as overall has been reached to 0.86 which is termed as “F1-score”. Furthermore, there is “support” parameter included which as well can be defined as the number of instances of each class where there are 597 instances of class “0” while only 201 instances of class "1”. Apart from this there is “macro avg” and “weighted avg”, where the former one can be defined as the result of average performance of across all classes when considered independently which means that it does not consider the frequency of each class in the dataset giving equal weightage to each class so that there is no biased decision made. However, the later one means that the average result of performance across each class but here the frequency parameter is considered which means that it will give more weightage to the classes whose frequency is higher than the frequency of other classes and that can be used when given an imbalanced class in dataset. Thus, the model is performing better in weighted average criteria rather than the macro average ones as in the dataset that I have selected has a greater number of images of stars than galaxies, making weighted average across precision, recall and F1-score to be around 0.70.

Next, coming to the confusion matrix parameter for evaluation of the True Positives, True Negatives, False Positives and False Negatives in the performance of the RF model and the result of it can be seen below:

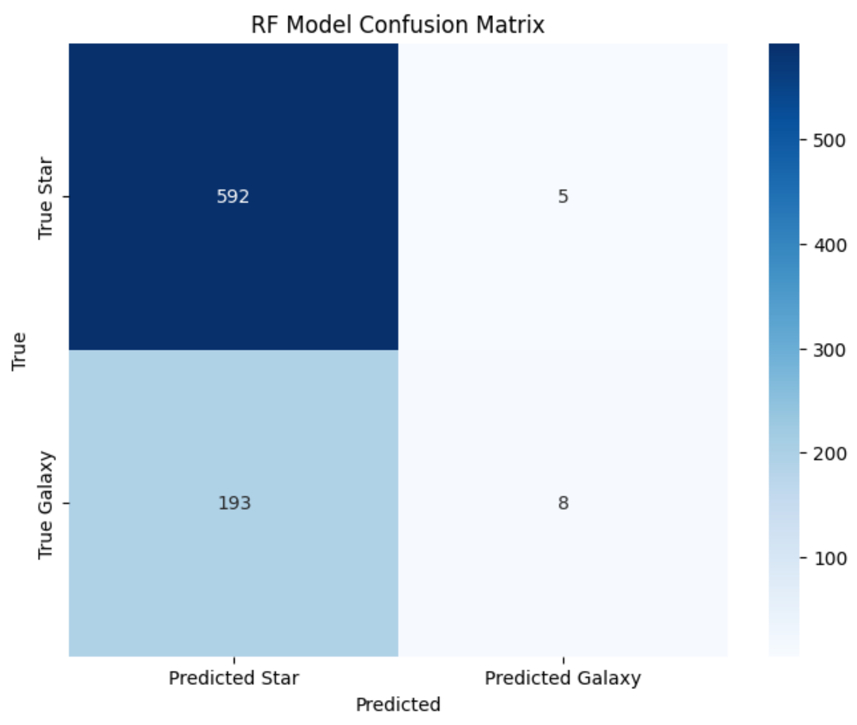


Figure 41: Confusion matrix of RF model.

Here, it can be depicted from the image that the prediction of star and the actual image of star has been classified well with the instances of 592 but the scenario of the actual image of galaxy and the prediction of star has been the reason for decrease in accuracy of the whole model as there are 193 cases where the model incorrectly classified the images of galaxies.

Lastly, there is AUC ROC Curve parameter which has 0.81 of AUC making the model quite compatible but still not completely reliable as compared to the CNN model. The result is attached below:

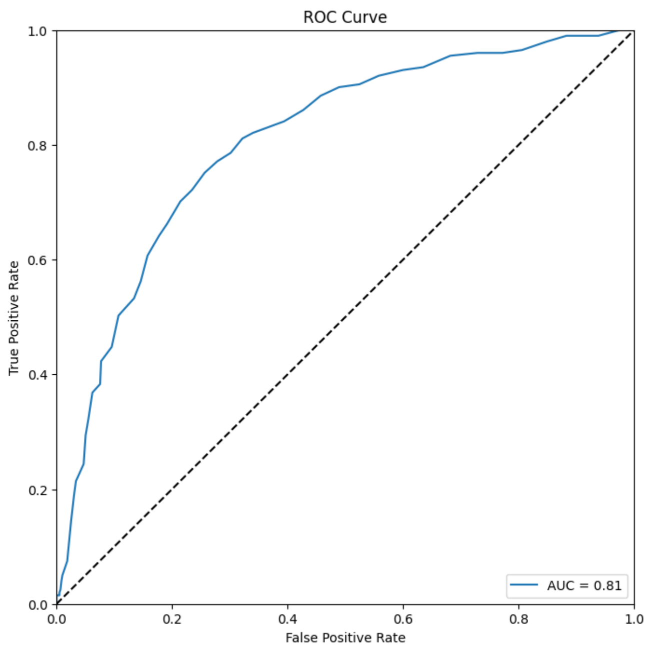


Figure 42: AUC ROC Curve for RF model which has AUC value of 0.81.

The curve that is being made is not as sharp as it should be as the accuracy of RF model is of 75% only as a reason of which the ROC Curve seems to be bended near to the intermediate line.

4.3 COMPARISON BETWEEN CNN and RF CLASSIFIER MODEL (ANALYSIS)–

The results and evaluation that I performed, clearly interprets that my CNN model has been able to give far better results than my RF classifier model along with accuracy of 92% whereas RF classifier was able to give far accurate results of 75%. The performance matrix that I have shown above also suggests that as the CNN model is able to extract features more adequately which is making it to understand the orientations and patterns of images of stars and galaxies in much better way than the RF classifier model which considers limited feature vectors. Also, data augmentation part in CNN made it more accurate as I was able to feed into it the necessary techniques which made the model perform better whereas in RF classifier model, I chose to have 100 “n_estimators” so that there are 100 number of trees for varied classification parameters to be included. Thus, looking at the precision, recall, F1-scores, and confusion matrix along with AUC ROC Curve matrices, they all are giving better results in CNN model as compared to the RF classifier one making the former model more desirable when performing the image classification part.

CHAPTER-5: CONCLUSION

To conclude, I would like to drive the focus towards the part where the literature review that I did throughout my project work period and the fact that it has benefited me in completing the models building part smoothly and efficiently. Here, I was able to get a proper insight into the numerous surveys that are currently involved in this process of image classification of astronomical objects and the different kinds of datasets they were able to extract from the research they performed using latest technologies as well as have been able to develop since earlier times. The datasets that were collected had images various kinds of images of galaxies which were in shapes of spiral, elliptical, round, and so on and their transformations throughout the period to understand the evolution of them adequately. In similar manner, the star-formation rate can be able to identify from the shape index they have which uses pixels in stars going from dark-white blob valley to understand the structure of it properly (Kremer et al., 2017).

In addition to this, I was able to understand several models that are highly preferred when performing image classification such as Naïve Bayes (NB), Convolutional Neural Networks (CNN), k-Nearest Neighbour (kNN), Random Forest Classifier (RF), Support Vector Machines (SVM) and so on from which CNN is the best model to be used for performing star-galaxy image classification followed by RF as compared to other models which gives better accuracy (Iyer, XXXX).

Thereafter, I thoroughly understood the working of CNN model along with its layers involved and the methods to overcome the overfitting problem that gave me an accuracy of 92% and then came RF model in which I had an accuracy of 73% where I used 100 number of trees and other parameters due to which I was able the final accuracy. Moreover, one of the major advantages of using RF along with CNN was that RF classifier can handle high-dimensional data that has certain features in it, which makes it the ultimate choice for image classification (Solorio-Ramírez, 2023).

Lastly, I have compared both the models on different performance metrics that are precision, recall, f1-score, and support as well so that I was able to get an insight into the working of both models as a result of which CNN was found to outperform RF classifier model in each of the metrics giving it the edge to perform better and more accurate.

CHAPTER-6: FUTURE WORK

There are number of scopes of improvements in the technology that we are working on in this modern world for classification of images but as the space is a vast place and new objects are coming into the scenario on the regular basis, the advancement will become a necessity to cope up with these changes. Therefore, there are multiple areas where the work will be done in upcoming years starting from the different surveys like LSST (Large Synoptic Survey Telescope) which aims to collect huge sets of data in fields like dark matter and so on to enhance the research on numerous phenomena in space and study them in detail, various telescopes are coming into actions for performing high activities on astronomical observations like Euclid Space Telescope, Wide Field Infrared Survey Telescope and James Webb Space Telescope.

Apart from that if I continue to pursue this project further on, then my prominent aim will be to classify more anonymous bodies of space by using feature extraction and data augmentation parts as well as to further classify these objects I also aim to get higher accuracy by using adequate techniques. Moreover, I will focus on adjusting some of the hyperparameters included in RF classifier model so that the mode will be able to give better accuracy in future and it will be a new addition as well to my project work.

REFERENCING

Garg, P. (2022). Star Galaxy Image Classification Via Convolutional Neural Networks.

Iyer, G.R.C. (XXXX). Deep Learning for Star-Galaxy Classification. University of California, San Diego.

Kim, E.J. and Brunner, R.J. (2016). Star–galaxy classification using deep convolutional neural networks. Monthly Notices of the Royal Astronomical Society, [online] 464(4), pp.4463–4475. doi:https://doi.org/10.1093/mnras/stw2672.

Kremer, J., Stensbo-Smidt, K., Gieseke, F., Pedersen, K.S. and Igel, C. (2017). Big Universe, Big Data: Machine Learning and Image Analysis for Astronomy. IEEE Intelligent Systems, 32(2), pp.16–22. doi:https://doi.org/10.1109/mis.2017.40.

Martinazzo, A. (n.d.). Deep Learning for Astronomical Object Classification: A Case Study. In Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020), Volume 5. doi:https://doi.org/10.5220/0008939800870095.

Savyanavar, A.S. (2023). STAR-GALAXY CLASSIFICATION USING MACHINE LEARNING ALGORITHMS AND DEEP LEARNING. International Journal on Information Technologies & Security, 15.

Solorio-Ramírez, J.-L. (2023). Random forest Algorithm for the Classification of Spectral Data of Astronomical Objects. doi:https://doi.org/10.3390/a16060293.

Yamashita, R., Nishio, M., Do, R.K.G. and Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. Insights into Imaging, [online] 9(4), pp.611–629. doi:https://doi.org/10.1007/s13244-018-0639-9.

Xi, E. (2022). Image Classification and Recognition Based on Deep Learning and Random Forest Algorithm. Wireless Communications and Mobile Computing, 2022, pp.1–9. doi:https://doi.org/10.1155/2022/2013181.

APPENDIX

CODE:

Connecting to google drive for direct access of dataset

```
[ ]
# mount google drive

from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive
```

CONVOLUTIONAL NEURAL NETWORK MODEL

Importing necessary modules for performing functions

```
[ ]
# import modules

import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras import layers, models, optimizers, losses
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix

from tensorflow.keras.preprocessing.image import load_img

from sklearn.metrics import precision_score, recall_score, f1_score, roc_curve,
roc_auc_score

from tensorflow.keras.utils import plot_model
```

Loading the dataset along with preprocessing images for a better clear image and splitting dataset into train and validation part

```
[ ]

# load dataset and set class labels


astronomy_data = '/content/drive/MyDrive/FinalProjectWork/TRAIN'

astronomy_class_labels = ['star', 'galaxy']




# preprocess images


star_galaxy_image_size = (64, 64)

astronomy_batch_size = 32


astronomy_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

astronomy_train_datagen = ImageDataGenerator(
   rescale=1./255,
   rotation_range = 45,  # range of rotation angles (in degrees)
   zoom_range=0.2, # zooming image by 20%
   horizontal_flip=True, # flipping image horizontally
   validation_split=0.2 # split 20% of dataset into for validation
)
```

```
# train CNN model from dataset features
astronomy_train_generator = astronomy_datagen.flow_from_directory(
    astronomy_data,
    target_size=star_galaxy_image_size,
    batch_size=astronomy_batch_size,
    class_mode='binary',
    subset='training'
)

astronomy_validation_generator = astronomy_datagen.flow_from_directory(
    astronomy_data,
    target_size=star_galaxy_image_size,
    batch_size=astronomy_batch_size,
    class_mode='binary',
    subset='validation'
)
Found 3190 images belonging to 2 classes.
Found 796 images belonging to 2 classes.
[ ]
augmented_img =
astronomy_datagen.random_transform(astronomy_train_generator[0][0][0])
plt.imshow(augmented_img)
plt.title('Final Augmented Image')
plt.axis('off')
plt.show()
```

```
[ ]
# Assuming train_generator contains your normal star-galaxy images
normal_img = astronomy_train_generator[0][0][0]
plt.imshow(normal_img)
plt.title('Normal Image Before Rotation')
plt.axis('off')
plt.show()
```

In next step, CNN model is built by use of Dropout regulariser and displayed.

```
[ ]
# CNN model build



# define input shape

input_shape = (star_galaxy_image_size[0], star_galaxy_image_size[1], 3)



# define input layer

inputs = layers.Input(shape=input_shape, name='Input')
conv_1 = layers.Conv2D(32, (3, 3), activation='relu', name='conv_1')(inputs)
pool_1 = layers.MaxPool2D(pool_size=(2, 2), name='MaxPool2D_1')(conv_1)
conv_2 = layers.Conv2D(64, (3, 3), activation='relu', name='conv_2')(pool_1)
pool_2 = layers.MaxPool2D(pool_size=(2, 2), name='MaxPool2D_2')(conv_2)
```

```
conv_3 = layers.Conv2D(128,(3, 3), activation='relu', name='conv_3')(pool_2)

pool_3 = layers.MaxPool2D(pool_size=(2, 2), name='MaxPool2D_3')(conv_3)

flatten = layers.Flatten(name='flatten')(pool_3)

dense_1 = layers.Dense(128, activation='relu', name='dense_1')(flatten)

dropout = layers.Dropout(0.5, name='Dropout')(dense_1)

output = layers.Dense(1, activation='sigmoid', name='output')(dropout)




# define model


astronomy_model = models.Model(inputs=inputs, outputs=output,
name='astronomy_model')




# display model


astronomy_model.summary()


Model: "astronomy_model"
_________________________________________________________________

 Layer (type)              Output Shape              Param #

=================================================================

 Input (InputLayer)        [(None, 64, 64, 3)]       0


 conv_1 (Conv2D)           (None, 62, 62, 32)        896


 MaxPool2D_1 (MaxPooling2D)  (None, 31, 31, 32)       0
 conv_2 (Conv2D)           (None, 29, 29, 64)       18496

 MaxPool2D_2 (MaxPooling2D)  (None, 14, 14, 64)       0

 conv_3 (Conv2D)           (None, 12, 12, 128)      73856

 MaxPool2D_3 (MaxPooling2D)  (None, 6, 6, 128)        0

 flatten (Flatten)         (None, 4608)             0

 dense_1 (Dense)           (None, 128)              589952

 Dropout (Dropout)         (None, 128)              0

 output (Dense)            (None, 1)                129

=================================================================
Total params: 683,329
Trainable params: 683,329
Non-trainable params: 0
_________________________________________________________________
```

[ ]

```
# display CNN model

plot_model(astronomy_model, show_shapes=True, show_layer_names=True)
```

Now, the CNN model that was built above will be compiled and trained for 10 epochs

```
[ ]
# compile model

astronomy_model.compile(optimizer='Adam',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])



# run model with 10 epochs

history = astronomy_model.fit(
   astronomy_train_generator,
   steps_per_epoch = astronomy_train_generator.samples // astronomy_batch_size,
   epochs = 10,
   validation_data = astronomy_validation_generator,
   validation_steps = astronomy_validation_generator.samples // astronomy_batch_size
)
Epoch 1/10
99/99 [==============================] - 11s 89ms/step - loss: 0.5284 - accuracy:
0.7619 - val_loss: 0.4284 - val_accuracy: 0.7891
Epoch 2/10
99/99 [==============================] - 7s 73ms/step - loss: 0.3922 - accuracy: 0.8011
- val_loss: 0.3640 - val_accuracy: 0.8516
Epoch 3/10
99/99 [==============================] - 7s 72ms/step - loss: 0.3568 - accuracy: 0.8436
- val_loss: 0.4135 - val_accuracy: 0.8216
Epoch 4/10
99/99 [==============================] - 7s 75ms/step - loss: 0.3354 - accuracy: 0.8547
- val_loss: 0.3216 - val_accuracy: 0.8672
```

```
Epoch 5/10

99/99 [==============================] - 7s 73ms/step - loss: 0.3049 - accuracy: 0.8711
- val_loss: 0.2963 - val_accuracy: 0.8711

Epoch 6/10

99/99 [==============================] - 7s 71ms/step - loss: 0.2694 - accuracy: 0.8847
- val_loss: 0.2952 - val_accuracy: 0.8737

Epoch 7/10

99/99 [==============================] - 7s 71ms/step - loss: 0.2510 - accuracy: 0.8965
- val_loss: 0.2910 - val_accuracy: 0.8841

Epoch 8/10

99/99 [==============================] - 9s 90ms/step - loss: 0.2426 - accuracy: 0.9003
- val_loss: 0.3171 - val_accuracy: 0.8646

Epoch 9/10

99/99 [==============================] - 39s 396ms/step - loss: 0.2167 - accuracy:
0.9120 - val_loss: 0.3086 - val_accuracy: 0.8737

Epoch 10/10

99/99 [==============================] - 7s 71ms/step - loss: 0.1918 - accuracy: 0.9243
- val_loss: 0.2819 - val_accuracy: 0.8919
```

Evaluating the model along with plotting accuracy and loss curves and values as well

```
[ ]

# model evaluation


astronomy_test_generator = astronomy_datagen.flow_from_directory(

    astronomy_data,

    target_size=star_galaxy_image_size,

    batch_size=32,

    class_model='binary',

    shuffle=False

)
```

```
# plot accuracy and loss curves

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training')
plt.plot(history.history['val_loss'], label='Validation')
plt.title('Model Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training')
plt.plot(history.history['val_accuracy'], label='Validation')
plt.title('Model Accuracy')
plt.legend(loc='upper right')
plt.show()
```

```
[ ]
# display loss and accuracy of cnn model

astronomy_loss, astronomy_accuracy =
astronomy_model.evaluate(astronomy_test_generator)
print(f"CNN Model Loss: {astronomy_loss:.2f}")
print(f"CNN Model Accuracy: {astronomy_accuracy:.2f}")
125/125 [==============================] - 7s 58ms/step - loss: 0.2052 - accuracy:
0.9190
CNN Model Loss: 0.21
CNN Model Accuracy: 0.92
```

Here, the prediction of classified images is performed, and correctly classified images are displayed along with their labels

```
[ ]
# display correctly classified images


x_test, y_test = astronomy_validation_generator.next()
y_pred = astronomy_model.predict(x_test)


fig, axes = plt.subplots(5, 5, figsize=(12, 12))
fig.subplots_adjust(hspace=0.8, wspace=0.8)
for i, ax in enumerate(axes.flat):
  ax.imshow(x_test[i])
  if y_pred[i][0] < 0.5: # set conditions for classification of stars and galaxies
    predict_label = 'Star'
  else:
    predict_label = 'Galaxy'


  if y_test[i] == 0:
    true_label = 'Star'
  else:
    true_label = 'Galaxy'


  title = f'True: {true_label}\nPrediction: {predict_label}'
  ax.set_title(title, fontsize=10)
  ax.axis('off')
plt.show()
```

Performing metrics insights like Precision, Recall, F1-score, Confusion Matrix and AUC ROC Curve on CNN model

```
[ ]
# evaluate model

y_true = astronomy_test_generator.classes
y_pred = astronomy_model.predict(astronomy_test_generator)
y_pred_classes = np.round(y_pred).flatten().astype(int)



# computing confusion matrix

astronomy_cm = confusion_matrix(y_true, y_pred_classes)



# claculating other metrics

astronomy_precision = precision_score(y_true, y_pred_classes)
astronomy_recall = recall_score(y_true, y_pred_classes)
astronomy_f1 = f1_score(y_true, y_pred_classes)



# display values of metrics

print("Confusion Matrix of CNN is: ", astronomy_cm)
print("Precision of CNN is: ", astronomy_precision)
print("Recall of CNN is: ", astronomy_recall)
print("F1 score of CNN is: ", astronomy_f1)




# plot confusion matrix


plt.figure(figsize=(6, 6))
sns.heatmap(astronomy_cm, annot=True, fmt='d', cmap='Blues', cbar=False,
        xticklabels=astronomy_class_labels, yticklabels=astronomy_class_labels)
plt.title("Confusion Matrix of CNN Model")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
# plt ROC curve


fpr, tpr, thresholds = roc_curve(y_true, y_pred)
astronomy_roc_auc = roc_auc_score(y_true, y_pred)


plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, color='blue', lw=2,
      label="ROC Curve (AUC = %0.2f)" % astronomy_roc_auc)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic(ROC)")
plt.legend(loc="lower right")
plt.show()
```

RANDOM FOREST CLASSIFIER MODEL

Importing necessary modules

```
[ ]
# import modules

import os
import numpy as np
from PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,
roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
import random
```

Dataset is loaded by use of function along with changing images to grayscale and extracting features from them

```
[ ]
# load dataset

def astronomy_load_data(image_dir):
  X_astronomy_data = []
  y_astronomy_label = []
  astronomy_classes = os.listdir(image_dir)
  for astronomy_class_name in astronomy_classes:
    astronomy_class_path = os.path.join(image_dir, astronomy_class_name)
    if os.path.isdir(astronomy_class_path):
      astronomy_class_label = 0 if astronomy_class_name == "star" else 1
      for image_file in os.listdir(astronomy_class_path):
        astronomy_image_path = os.path.join(astronomy_class_path, image_file)
        astronomy_image = Image.open(astronomy_image_path).convert('L') # convert image
to grayscale
        astronomy_image_array = np.array(astronomy_image)
        X_astronomy_data.append(astronomy_image_array.flatten()) # flattening the image for
creating feature vectors
        y_astronomy_label.append(astronomy_class_label)
  return np.array(X_astronomy_data), np.array(y_astronomy_label)


astronomy_image_dir = '/content/drive/MyDrive/FinalProjectWork/TRAIN'
X_astronomy_data, y_astronomy_label = astronomy_load_data(astronomy_image_dir)
```

RF classifier model is trained with test size of 20% and remaining 80% is used for training purposes

```
[ ]
# train the RF model

X_train, X_test, y_train, y_test = train_test_split(X_astronomy_data,
                                                    y_astronomy_label,
                                                    test_size=0.2,
                                                    random_state=42)
```

The model is built with 100 number of trees so that proper feature extraction is performed

```
[ ]
# build RF model

astronomy_rf = RandomForestClassifier(n_estimators=100, random_state=42)
astronomy_rf.fit(X_train, y_train)
```

Randomly classified images has been displayed along with labels on top of them

```
[ ]
# display correctly classified images




# randomly select subest for display of images
astronomy_samples = 25
random_indices = random.sample(range(len(y_test)), astronomy_samples)
astronomy_X = X_test[random_indices]
astronomy_y_true = y_test[random_indices]
astronomy_y_visualize_pred = astronomy_y_pred[random_indices]




# list for storing correctly classfied images

rf_correctly_classified = []




# interpret image shape from dataset

rf_image_shape = (int(np.sqrt(X_astronomy_data.shape[1])),
            int(np.sqrt(X_astronomy_data.shape[1])))
```

```
# display samples with labels

plt.figure(figsize=(12, 12))
for i in range(astronomy_samples):
  plt.subplot(5, 5, i + 1)
  plt.imshow(astronomy_X[i].reshape(rf_image_shape), cmap="gray")
  plt.title(f"True: {'Star' if astronomy_y_true[i] == 0 else 'Galaxy'}\nPred: {'Star' if
astronomy_y_pred[i] == 0 else 'Galaxy'}")
  plt.axis('off')
  if astronomy_y_true[i] == astronomy_y_pred[i]:
    rf_correctly_classified.append(i)

plt.tight_layout()
plt.show()
```

Accuracy curve is plotted with intervals set as number of trees

```
# plot accuracy of RF model in accordance to n_estimators

astronomy_num_trees = [0, 10, 20, 50, 100] # set intervals for n_estimators
rf_accuracy_score = []

for n in astronomy_num_trees:
  astronomy_rf = RandomForestClassifier(n_estimators=n, random_state=42)
  astronomy_rf.fit(X_train, y_train)
  astronomy_y_pred = astronomy_rf.predict(X_test)
  rf_accuracy = accuracy_score(y_test, astronomy_y_pred)
  rf_accuracy_score.append(rf_accuracy)
```

```
plt.figure(figsize=(8, 8))

plt.plot(astronomy_num_trees, rf_accuracy_score, marker='o')

plt.title("RF Accuracy Plot")

plt.xlabel("n_estimators (Number of Trees)")

plt.ylabel("Accuracy")

plt.xticks(astronomy_num_trees) # set intervals on x-axis as n_estimators

plt.grid(True)

plt.show()
```

RF classifier model evaluation is done in following part of code and classification report containing Precision, Recall, F1-score and Support has been generated

```
[ ]
# evaluate RF model




# predict the outcome

astronomy_y_pred = astronomy_rf.predict(X_test)





# calculate RF model accuracy


astronomy_rf_accuracy = accuracy_score(y_test, astronomy_y_pred)

print("RF Model Accuracy is: ", astronomy_rf_accuracy)
```

```
# calculate RF model classification report

astronomy_class_report = classification_report(y_test, astronomy_y_pred)
print("RF Model Classification Report: \n", astronomy_class_report)
RF Model Accuracy is:  0.7518796992481203
RF Model Classification Report:
             precision    recall  f1-score   support

         0       0.75      0.99      0.86       597
         1       0.62      0.04      0.07       201

    accuracy                           0.75       798
   macro avg       0.68      0.52      0.47       798
weighted avg       0.72      0.75      0.66       798
```

Along with this, Confusion Matrix has been plotted

```
[ ]
# calculate and display confusion matrix

astronomy_confusion_matrix = confusion_matrix(y_test, astronomy_y_pred)
print("RF Model Confusion Matrix: \n", astronomy_confusion_matrix)
print("\n") # extra space below after confusion matrix



# plot confusion matrix

plt.figure(figsize=(8, 6))
```

```
sns.heatmap(astronomy_confusion_matrix, annot=True, fmt='d', cmap="Blues")
plt.title("RF Model Confusion Matrix")
plt.xticks([0.5, 1.5], ["Predicted Star", "Predicted Galaxy"]) # set position for "x" labels
plt.yticks([0.5, 1.5], ["True Star", "True Galaxy"]) # set position for "y" labels
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

AUC ROC curve is displayed which depicts the behaviour and accuracy of the model overall

```
[ ]
# display AUC ROC Curve for RF Model



# calculate predicted probabilties for positive class

rf_y_pred_prob = astronomy_rf.predict_proba(X_test)[:, 1]



# calculate ROC Curve

false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,
                                                                rf_y_pred_prob)



# Calculate AUC for ROC Curve
```

```
rf_auc = roc_auc_score(y_test, rf_y_pred_prob)



# plot the results


plt.figure(figsize=(8, 8))

plt.plot(false_positive_rate, true_positive_rate, label=f'AUC = {rf_auc:.2f}')

plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve")

plt.legend(loc="lower right")

plt.show
```