# PROFESSIONAL TRAINING REPORT

**at**

**Sathyabama Institute of Science and Technology (Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

## CHAKALI GANGADHAR

**REG. NO. 39110202**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI – 600119, TAMILNADU**

**NOVEMBER 2021**

|  |  |  |
| --- | --- | --- |
|  | **SATHYABAMA**  **INSTITUTE OF SCIENCE AND TECHNOLOGY** (DEEMED TO BE UNIVERSITY) **Accredited with Grade "A" by NAAC**  (Established under Section 3 of UGC Act, 1956)  JEPPIAAR NAGAR, RAJIV GANDHI SALAI  CHENNAI– 600119  [**www.sathyabama.ac.in**](http://www.sathyabama.ac.in) |  |

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **CHAKALI GANGADHAR (Reg. No: 39110202),** who carried out the project entitled "**Rice Type Classification using Machine Learning Algorithm**" under my supervision from June 2021 to November 2021.

## Internal Guide

## Dr. M. Selvi, M.E., Ph.D.

**Head of the Department**

**Dr. S. Vigneshwari, M.E., Ph.D.**

**Dr. L. Lakshmanan, M.E., Ph.D.**

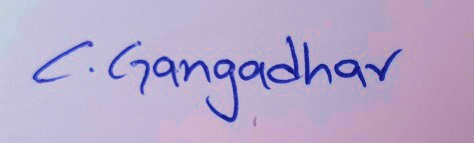


## Submitted for Viva-voce Examination held on

**InternalExaminer ExternalExaminer**

**DECLARATION**

I, **CHAKALI GANGADHAR,** hereby declare that the project report entitled **Rice Type Classification using Machine Learning Algorithm** done by me under the guidance of **Dr. M. Selvi** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

****

## DATE:

**PLACE:** Chennai, Tamil Nadu **SIGNATURE OF THE CANDIDATE**

**ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.M. Selvi, M.E., Ph.D.,** for his valuable guidance, suggestions, and constant encouragement that paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project

**TRAINING CERTIFICATE**

****



# ABSTRACT

Rice is one of the most broadly created and devoured cereal crops in the world. It is also one of the principal food in our country due to its economic and nutritious nature. Rice, beginning from farm to our table, goes through some manufacturing steps, for example, a cleaning cycle, shading arranging, and classification. If these stages are to be referenced momentarily, cleaning is the most common way of isolating rice from unknown substances, and classification is a method involved with separating broken ones with durable ones; shading extraction is the method involved with isolating the stained and striped ones aside from the whiteness on the rice surface.

In this review, an automated vision framework was created to recognize two exclusive rice species, and a dataset is prepared with its features. A sum of 3810 rice grain pictures was taken for the two varieties, processed, and highlight features were made. Seven morphological features were acquired for each grain of rice. With these features, a machine learning model was made utilizing Random Forest (RF), machine learning algorithm techniques, and acquired performance metrics values.

Achievement rates in the classification were obtained at **93.96%** accuracy using Random Forest (RF) algorithm techniques. By looking at the results when that kind of success rate was obtained, it is possible to say that the study achieved success. In this latter part of the study, Random Forest Classifier is referred to as a 'model.'

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| CHAPTER No. | TITLE | PAGE No |
|  | ABSTRACT | i |
|  | LIST OF FIGURES | iii |
|  | LIST OF TABLES | v |
|  | LIST OF ABBREVIATIONS | vi |
| 1. | INTRODUCTION | 1 |
|  | 1.1 Area of research | 2 |
| 2. | AIM AND SCOPE OF THE PRESENT INVESTIGATION | 3 |
|  | 2.1 Aim of the project | 3 |
|  | 2.2 Scope of the project | 3 |
| 3. | MATERIALS AND METHODS, ALGORITHMS USED | 4 |
|  | 3.1 Understanding Dataset | 5 |
|  | *3.1.1 Understanding dataset via tables* | 6 |
|  | *3.1.2 Understanding dataset via Graphs* | 9 |
|  | 3.2 Data Preprocessing | 11 |
|  | *3.2.1 Label Encoding* | 12 |
|  | 3.3 Random Forest Algorithm | 13 |
|  | 3.4 Understanding the model algorithm | 14 |
|  | 3.5 Building and testing the model | 15 |
|  | 3.6 Performance metrics | 17 |
|  | *3.6.1 Confusion Matrix* | 18 |
|  | *3.6.2 Log loss* | 21 |
|  | *3.6.3 Accuracy score* | 21 |
|  | *3.6.4 ROC AUC curve* | 21 |
|  | *3.6.5 ROC AUC score* | 22 |
| 4. | RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS | 23 |
|  | 4.1 Analyzing performance | 23 |
|  | 4.2 Final result | 24 |
| 5 | SUMMARY AND CONCLUSIONS | 25 |
|  | 5.1 Summary of the project | 25 |
|  | 5.2 Conclusion of the project | 27 |
|  | REFERENCES | 28 |
|  | APPENDIX | 30 |
|  | A. SCREENSHOTS | 30 |
|  | B. SOURCE CODE | 34 |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| FIGURE NO. | FIGURE NAME | PAGE NO. |
| 3.1 | Architecture map of the project | 4 |
| 3.2 | Correlation among features of rice | 9 |
| 3.3 | Feature graphs in the dataset | 10 |
| 3.4 | Graph of to be predicted class | 10 |
| 3.5 | Label encoding process of categorical data | 12 |
| 3.6 | Random Forest general structure | 13 |
| 3.7 | Model complexity factors | 15 |
| 3.8 | Results of the best model at different test sizes | 16 |
| 3.9 | First decision tree sample of RF model | 17 |
| 3.10 | Structure of confusion matrixfor binary classification | 18 |
| 3.11 | Confusion matrix for testing data | 19 |
| 3.12 | ROC curve of RF model | 22 |
| 4.1 | Best model result at 0.2 test size | 23 |

## 

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| TABLE NO. | TABLE NAME | PAGE NO. |
| 3.1 | Features of rice extracted from image processing | 5 |
| 3.2 | Sample records of rice at the beginning of the dataset | 6 |
| 3.3 | Sample records of rice in the ending of the dataset | 6 |
| 3.4 | Unique values in the dataset. | 7 |
| 3.5 | Information about the dataset | 7 |
| 3.6 | Description of the Dataset | 8 |
| 3.7 | Description of Cammeo Rice type | 8 |
| 3.8 | Description of Osmancik Rice type | 8 |
| 3.9 | Performance Measurements | 20 |
| 4.1 | Random Forest Model parameters | 24 |
| 4.2 | Results of random forest model | 24 |

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| **ABBREVIATION** | **EXPANSION** |
| ML | Machine Learning |
| RF | Random Forest |
| b/w | between |
| No. | Number of |
| min | Minimum |
| max | Maximum |
| DT | Decision Tree |
| int | integer |
| tp | True Positives |
| fp | False Positive |
| tn | True Negatives |
| fn | False Negatives |

Chapter-1

**INTRODUCTION**

Rice is the most produced crop in various parts of India. India stands the second position in making of rice in overall the world, soon after China. Rice is an essential food source for around 80% of the Southeast Asia population alone. When we look at the data of grains production worldwide, rice is the most produced, following wheat and corn. Rice is very rich in carbohydrates and starch. Rice has extraordinary significance in human nutrition in our nation, just as in the world as far as being nutritious and economical. In addition, it is popularly utilized in industry.

Different quality measures for rice production in our nation are made available. These are physical appearance, cooking characteristics, fragrance, taste, and smell are the issues, such as efficiency next to the properties. According to the point of view of the end purchaser, the first feature is the physical appearance that rings a bell from the criteria that stand out in the rice varieties that are sold bundled on market shelves. After production, it is seen that the requirement for technology acquired techniques increments because the calibration of rice, assurance of its types, and separation of different quality components are wasteful and tedious, particularly in terms of those with high production volume using human resources.

When studies were examined, rice features were extracted with automated image processing machines based on the physical characteristics of rice by utilizing various image preprocessing techniques with resultant images. These extracted features have been used in multiple ways. It is observed that more than 90% of the success rate has been gained by using the Random Forest algorithm, particularly with similar features evaluated together.

The second part of my review explains the methods used to understand acquired features by processing features with python build-in modules and cleaning the dataset by eliminating unnecessary data and filling missing values in the dataset. The methods used for visualizing data with graphs for comparisons and better understanding, building data structure for ML algorithm, with compatible physical features of rice for RF model. In the third part of my review, it is explained that the characteristics of the RF Algorithm, a method to build, test, and evaluate the RF algorithm for better accuracy, explaining the selection of model features for the obtained score. In the fourth part of my review, we discuss the result of my model, and the analysis is done on the performance of the model where it can be better understood by visualizing it. In the fifth part of my review, we conclude the things about the model and summarize the project with its scope, usability, compatibility, prerequisites, limits, and achievement.

**1.1 Area of Research**

Machine learning (ML) investigates computer science algorithmic calculations that can work naturally through experience and data. It is part of artificial intelligence. Machine learning algorithms fabricate a model dependent on example data, known as "Training data," to make predictions or decisions without programming for every application. Machine learning algorithms are utilized in various applications, such as medication, email sorting, speech recognition, and computer vision. It is troublesome or unworkable to develop ordinary algorithms to perform the required tasks. A subset of machine learning is rigidly identified with computational statistics, which centers around making predictions utilizing computers, yet not all machine learning is statistical learning. The investigation of mathematical optimization conveys strategies, hypotheses, and application areas to the field of machine learning. Data mining is a related field of study, focusing on exploration data investigation through unsupervised learning. Some executions of machine learning use data and neural networks that impersonate the working of an actual brain. In its application across business issues, machine learning is likewise referred to as prescient analysis.

Chapter-2

**AIM AND SCOPE OF THE PRESENT INVESTIGATION**

**2.1 Aim of the project**

The project's primary goal is to train an ML model with a given Algorithm that can predict the type of rice-based on the input features. The main challenge of this project is to understand the dataset, deal with missing values, use the right performance metrics for the algorithm and train the model with good accuracy for classification. Using python and python integrated modules helps to face the challenges of a dataset and make an efficient model for predicting things.

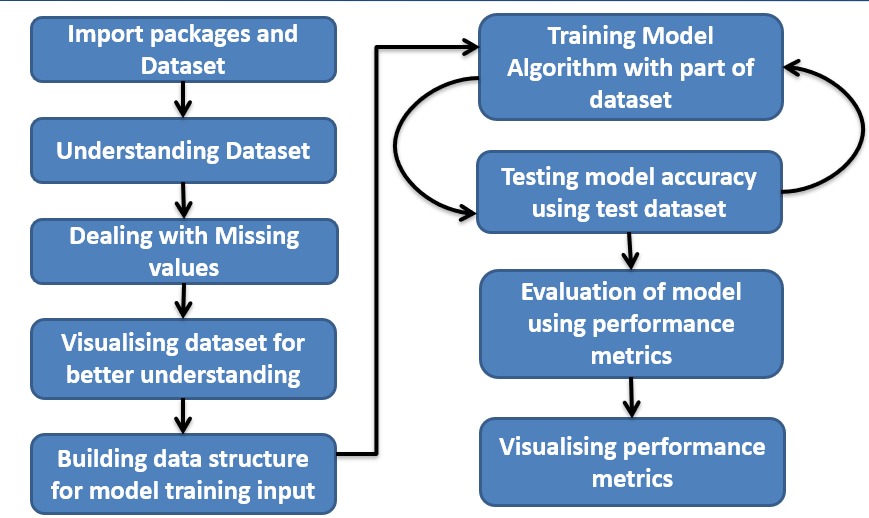
**2.2 Scope of the project**

The ML model we are building is used by machines in rice industries where there is a need for rice types separation based on the physical structure. Many ML Algorithms can solve this problem, but In this project, we are adopting the Random Forest algorithm to classify rice, which is one of the better performing ML algorithms. Using source code and compatible software for machines makes it possible to make working machines to differentiate rice types, which reduces time and human resources for classifying rice and can be invested in something productive.

Chapter-3

**MATERIALS AND METHODS, ALGORITHMS USED**

In my study, the rice examples were acquired by various image processing techniques, and values are noted in the dataset. Before developing the model, we need to understand the dataset, examine each feature, find the relations b/w the features, how the relations impact my results, and extract morphological features of rice. This phase is called understanding data. After understanding the data, we need to clean the dataset, drop unnecessary values, fill the missing values, and deal with categorical data. This phase is called data preprocessing. This data preprocessing helps to make compatible data structure for the model. After this preprocessing data phase, we need to understand the model, how it works, the advantages and disadvantages of algorithms, and its hyperparameters that help in performance. After understanding the model, we need to build the model with studied hyperparameters, test them with different variables of hyperparameters. This phase is called testing and validation. In this phase, we also need to understand the performance metrics of my problem statement, which helps in validating the model to reach my goal. All of these phases of the project are explained in detail in upcoming chapters and subchapters.



***Fig:3.1:*** *Architecture map of the project*

**3.1 Understanding Dataset**

The features of rice extracted from image processing techniques have been noted in the dataset, which is numerical data and represents the physical structure of rice. For understanding data with python integrated modules, we need to import necessary modules for working with a dataset and import the dataset to the appropriate python notebook using pandas. Pandas module is designed to work with datasets. With the help of pandas, it would be easy to understand the dataset. Using pandas [[6]](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html) commands, import the dataset and read the dataset. After we go through the rice features in the dataset, the best morphological features and characteristics utilized in feature extraction for the two kinds of rice are given in Table1.

**Table 3.1:** Features of rice extracted from image processing

|  |  |  |
| --- | --- | --- |
| S.no | Name | Characteristics |
| 1 | Area | It is a value of a number of pixels in the image within the boundaries of the rice grain. |
| 2 | Perimeter | Calculates the circumference by calculating the distance between pixels around the boundaries of the rice grain. |
| 3 | MajorAxisLength | The longest line that can be drawn on the rice grain, i.e., the main axis distance, gives. |
| 4 | MinorAxisLength | The shortest line that can be drawn on the rice grain, i.e., the small axis distance, gives. |
| 5 | Eccentricity | It measures how round the ellipse, which has the same moments as the rice grain, is. |
| 6 | ConvexArea | It is a value of the pixel count of the smallest convex shell of the region formed by the rice grain. |
| 7 | Extent | It is a value of the ratio of the region formed by the rice grain to the bounding box pixels. |
| 8 | Class | Labels of rice types |

***3.1.1 Understanding dataset via tables***

These features are used in the dataset are understood by definition. Still, we also need to understand the dataset's structure, how data is represented in the tabular form, find out the missing values, and fill the missing values. This data is described in the below tables.

**Table 3.2:** Sample records of rice at the beginning of the dataset

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | AREA | PERIMETER | MAJORAXIS | MINORAXIS | ECCENTRICITY | CONVEX\_AREA | EXTENT | CLASS |
| 0 | 15231 | 525.578979 | 229.749878 | 85.093788 | 0.928882 | 15617 | 0.572896 | Cammeo |
| 1 | 14656 | 494.311005 | 206.020065 | 91.730972 | 0.895405 | 15072 | 0.615436 | Cammeo |
| 2 | 14634 | 501.122009 | 214.106781 | 87.768288 | 0.912118 | 14954 | 0.693259 | Cammeo |
| 3 | 13176 | 458.342987 | 193.337387 | 87.448395 | 0.891861 | 13368 | 0.640669 | Cammeo |
| 4 | 14688 | 507.166992 | 211.743378 | 89.312454 | 0.906691 | 15262 | 0.646024 | Cammeo |

**Table 3.3:** Sample records of rice in the ending of the dataset

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | AREA | PERIMETER | MAJORAXIS | MINORAXIS | ECCENTRICITY | CONVEX\_AREA | EXTENT | CLASS |
| 3805 | 11441 | 415.858002 | 170.486771 | 85.756592 | 0.864280 | 11628 | 0.681012 | Osmancik |
| 3806 | 11625 | 421.390015 | 167.714798 | 89.462570 | 0.845850 | 11904 | 0.694279 | Osmancik |
| 3807 | 12437 | 442.498993 | 183.572922 | 86.801979 | 0.881144 | 12645 | 0.626739 | Osmancik |
| 3808 | 9882 | 392.296997 | 161.193985 | 78.210480 | 0.874406 | 10097 | 0.659064 | Osmancik |
| 3809 | 11434 | 404.709991 | 161.079269 | 90.868195 | 0.825692 | 11591 | 0.802949 | Osmancik |

Among the certified rice agriculture in our country, the Osmancik species, which has a substantial establishing region starting around 1997, and the Cammeo species developed starting about 2014 have been chosen for the study. When examining the overall attributes of Osmancik species, they have a wide, long, shiny, and dull appearance. When looking at the general characteristics of the Cammeo species, they have a wide furthermore, long, lustrous and dull appearance. In this review, the dispersion of 3810 rice grains was acquired because of handling pictures of the two species. We need to build the description of the dataset and unique rice types for data preprocessing. The no. unique values of the dataset are represented in Table 3.4.

**Table 3.4:** Unique values in the dataset.

|  |  |  |
| --- | --- | --- |
| S.no | Name | No. of values |
| 1 | Cammeo | 1630 |
| 2 | Osmancik | 2180 |
| Total |  | 3810 |

To find the missing values, we can count non-null values present in each column. If the count is the same as no. of records, then there are no missing values. The columns names, non-null values count data types are represented in Table 3.5

**Table 3.5:** Information about the dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.no. | Column | Non-null Count | Data Types | |
| 0 | AREA | 3810 non-null | int64 | |
| 1 | PERIMETER | 3810 non-null | float64 | |
| 2 | MAJORAXIS | 3810 non-null | float64 | |
| 3 | MINORAXIS | 3810 non-null | float64 | |
| 4 | ECCENTRICITY | 3810 non-null | float64 | |
| 5 | CONVEX\_AREA | 3810 non-null | int64 | |
| 6 | EXTENT | 3810 non-null | float64 | |
| 7 | CLASS | 3810 non-null | object | |
|  | dtypes: float64(5), int64(2), object(1) | | |

The number of records, minimum value, maximum value, standard deviation, mean, 25% of max value, 50%(median) of the max value, 75% of the max values on the min-max range values of the dataset are represented in Table 3.6.

**Table 3.6:** Description of the Dataset

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | AREA | PERIMETER | MAJORAXIS | MINORAXIS | ECCENTRICITY | CONVEX\_AREA | EXTENT |
| count | 3810.000000 | 3810.000000 | 3810.000000 | 3810.000000 | 3810.000000 | 3810.000000 | 3810.000000 |
| mean | 12667.727559 | 454.239180 | 188.776222 | 86.313750 | 0.886871 | 12952.496850 | 0.661934 |
| std | 1732.367706 | 35.597081 | 17.448679 | 5.729817 | 0.020818 | 1776.972042 | 0.077239 |
| min | 7551.000000 | 359.100006 | 145.264465 | 59.532406 | 0.777233 | 7723.000000 | 0.497413 |
| 25% | 11370.500000 | 426.144753 | 174.353855 | 82.731695 | 0.872402 | 11626.250000 | 0.598862 |
| 50% | 12421.500000 | 448.852493 | 185.810059 | 86.434647 | 0.889050 | 12706.500000 | 0.645361 |
| 75% | 13950.000000 | 483.683746 | 203.550438 | 90.143677 | 0.902588 | 14284.000000 | 0.726562 |
| Max | 18913.000000 | 548.445984 | 239.010498 | 107.542450 | 0.948007 | 19099.000000 | 0.861050 |

**Table 3.7:** Description of Cammeo Rice type

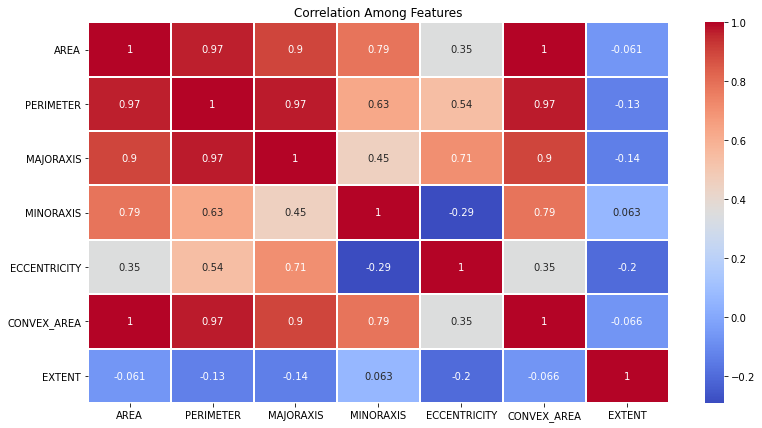
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | AREA | PERIMETER | MAJORAXIS | MINORAXIS | ECCENTRICITY | CONVEX\_AREA | EXTENT |
| count | 1630.000000 | 1630.000000 | 1630.000000 | 1630.000000 | 1630.000000 | 1630.000000 | 1630.000000 |
| mean | 14162.892025 | 487.438942 | 205.478589 | 88.767532 | 0.901047 | 14494.426994 | 0.651420 |
| std | 1286.770521 | 22.181518 | 10.333854 | 5.350244 | 0.013381 | 1309.418680 | 0.082197 |
| min | 9908.000000 | 410.506012 | 170.781647 | 67.695343 | 0.837433 | 10205.000000 | 0.497413 |
| 25% | 13289.250000 | 473.090248 | 198.580872 | 85.376022 | 0.893274 | 13620.250000 | 0.580921 |
| 50% | 14212.000000 | 488.179504 | 205.716743 | 88.830879 | 0.901759 | 14536.500000 | 0.634436 |
| 75% | 14997.000000 | 502.637756 | 212.433681 | 92.127367 | 0.909835 | 15361.000000 | 0.717667 |
| Max | 18913.000000 | 548.445984 | 239.010498 | 107.542450 | 0.948007 | 19099.000000 | 0.861050 |

**Table 3.8:** Description of Osmancik Rice type

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | AREA | PERIMETER | MAJORAXIS | MINORAXIS | ECCENTRICITY | CONVEX\_AREA | EXTENT |
| count | 2180.000000 | 2180.000000 | 2180.000000 | 2180.000000 | 2180.000000 | 2180.000000 | 2180.000000 |
| mean | 11549.783486 | 429.415505 | 176.287755 | 84.479042 | 0.876271 | 11799.585780 | 0.669796 |
| std | 1041.908607 | 20.154394 | 9.362405 | 5.302667 | 0.018999 | 1062.804346 | 0.072340 |
| min | 7551.000000 | 359.100006 | 145.264465 | 59.532406 | 0.777233 | 7723.000000 | 0.501078 |
| 25% | 10850.500000 | 416.207993 | 169.989491 | 81.333870 | 0.863816 | 11097.000000 | 0.610567 |
| 50% | 11552.500000 | 429.239502 | 175.665390 | 84.633549 | 0.876206 | 11813.500000 | 0.652695 |
| 75% | 12269.000000 | 442.506744 | 182.099850 | 87.932961 | 0.889286 | 12524.000000 | 0.732591 |
| max | 15420.000000 | 503.459991 | 209.651169 | 101.762260 | 0.935528 | 15800.000000 | 0.832747 |

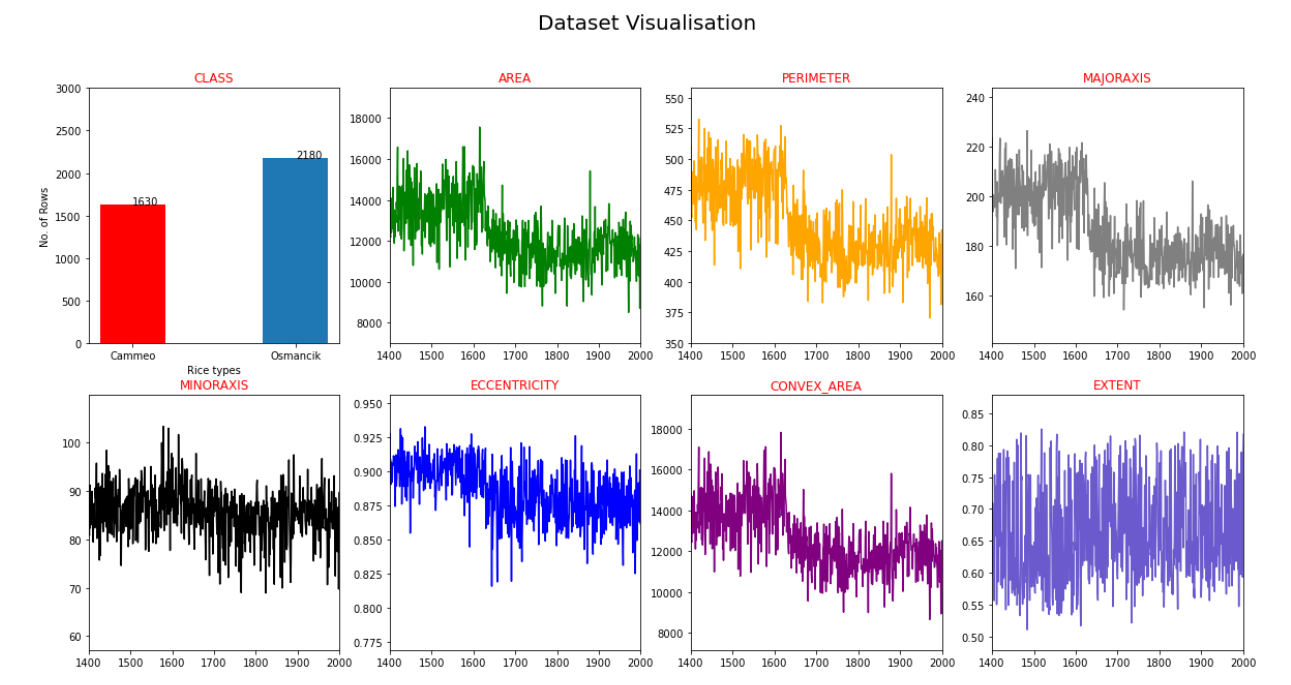
***3.1.2 Understanding dataset via Graphs***

Correlation is a statistical term portraying how much two variables move in coordination with each other. It can also be said as how two variables are dependent on each other. If the two variables move a similar way, those variables have a positive correlation [[16]](https://www.investopedia.com/terms/c/correlation.asp#:~:text=Correlation%20is%20a%20statistical%20term,they%20have%20a%20negative%20correlation.). If they move in inverse ways, they have a negative correlation. The correlation [[15]](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html.) among the features of the rice in the dataset has been shown in figure 3.2



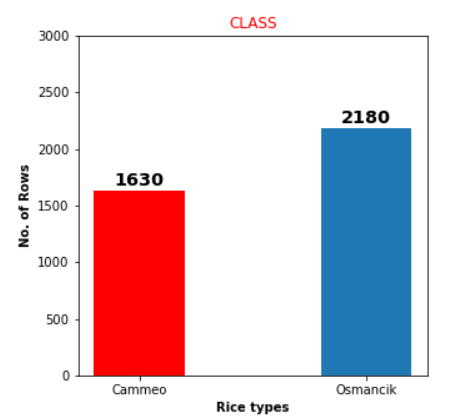
***Fig:3.2:*** *Correlation among features of rice*

As we know that the values of cammeo type of rice are at the beginning of the data set and have 1630 records of it, so to observe the differences in both types in every feature, we need to plot graph values ranging from 1400 to 2000. By plotting records of dataset range of 1400 to 2000, we observe that there are significant differences b/w two rice types in all features except 'Extent.' So, we assume that two rice types vary in range(min, max) values of features that represented in figure 3.3 [[2]](https://scikit-learn.org/stable/visualizations.html.)[[4]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html.)

****

***Fig:3.3:*** *Feature graphs in the dataset*

The rice type labels in the features are called class. It is shown in the graph for the number of rice types and their count in figure 3.4.

****

***Fig:3.4:*** *Graph of to be predicted class*

If we plot a feature vs. feature graph for all features in the dataset of two rice types, there are more similarities in the structure of graphs but the difference in coordinates. It is due to the main physical structure of rice being the same, but in terms of directions, they are different. This significant difference in range b/w two rice types helps to predict their classification with Random Forest Algorithm.

**3.2 Data Preprocessing**

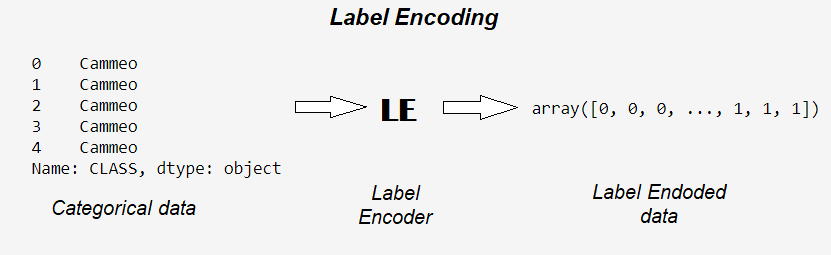
This phase is vitalfor building the model because, in this phase, we build a data structure that is compatible with the model, and we make random shuffles with the data to make the model learn the efficient way. Here we divide the data in terms of x and y, where x is the input of the model and y is the model's output.

*F(x) is the ML model, x is the input, and y is the model's output.*

After dividing input and output (x, y), we observe that output y that needs to be trained is an object data type. Random forest or any machine learning algorithm doesn't understand string or object datatype. This type of data is also known as categorical data. For making the machine understand object data type, it needs to be encoded. To deal with this categorical data, we need methods for encoding the data that the model can understand. There are different methods for encoding the categorical data like dropping columns, label encoding, and one-hot encoding, but we need to choose the method that suits the requirement. For my problem statement, we have a binary classification, which is classifying two rice types. The categorical data is prediction class. Here, we choose label encoding to deal with categorical data.

***3.2.1 Label Encoding***

Label Encoding refers to changing over the labels into a numeric structure to change over them into the machine-understandable structure. ML Algorithms would then be able to choose in a superior manner how those labels should be worked. It is a significant preprocessing step for the structured dataset in supervised learning. So basically converts cammeo and osmancik to 0 and 1 as labeling as integer data type, which becomes machine-readable data. The conversation of objects using python, in which there is a module called "LabelEncoder"[[17]](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html.) under sklearn. Pre-processing is used for label encoding, as shown in figure 3.5.

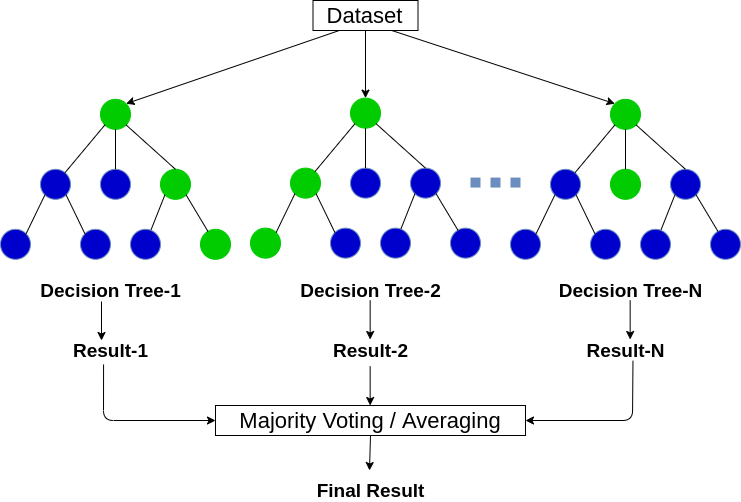


***Fig:3.5:*** *Label encoding process of categorical data*

After label encoding the prediction class, we need to split the data for the training and testing of the model. For dividing the dataset, we have an integrated python function called train\_test\_split under sklearn.model\_selection, which divided the dataset with train\_size or test\_size. This train\_size, test\_size ranges from 0 to 1, and another parameter called 'shuffle' shuffle the dataset's records to make the model learn efficiently. This function takes two variables which are the input and output of the model. It returns four values: input training data, output training data, input testing data, and output testing data. These returned values are stored in the variables.

**3.3 Random Forest Algorithm**

Random forest (RF) is a classifier that is a combination of many DT's. To make another classification, each DT gives a classification to the input sources. Then, in the end, RF evaluates the classifications and chooses the classification which has the most votes. RF can accommodate a large number of variables in the dataset. It is also excellent at estimating missing data. The most significant disadvantage of RF is the need for reproducibility. In addition, it is likewise hard to interpret the final model and its subsequent results. This is because of the way that it contains numerous independent decision trees. Figure 3.6 shows the general construction of RF.



***Fig:3.6:*** *Random Forest general structure*

Building a model needs a methodology to achieve good accuracy for the problem. We are following the bagging method [[1]](https://scikit-learn.org/stable/modules/ensemble.html#bagging-meta-estimator.) in this study. It is a combination of learning methods to increase the result. Random forest algorithm is based on bagging method, which is based on decision tree algorithm.

**3.4 Understanding the model algorithm**

Before developing the model, we need to understand the model's nature. Make Random forest classifier ML algorithm has parameters called hyperparameters which can be tuned to achieve maximum performance. The hyperparameters [[7]](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) used in my model are

* + n\_estimators: The number of decision trees in the forest,(int,default=100).
  + bootstrap: Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree. (bool, default=True).
  + max\_depth: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or contain less than min\_samples\_split samples. (int, default=None).
  + min\_samples\_leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least ``min\_samples\_leaf`` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression. - If int, then consider `min\_samples\_leaf` as the minimum number. - If float, then `min\_samples\_leaf` is a fraction, and `ceil(min\_samples\_leaf \* n\_samples)` is the minimum number of samples for each node. (int or float, default=1).
  + random\_state: Controls both the randomness of the bootstrapping of the samples used when building trees and the sampling of the features to consider when looking for the best split at each node (int, default=None).

Here the n\_estimators(default=100), bootstrap(bootstrap =True) and random\_state(random\_state =12) are unchanged values. n\_estimator represents no. of decision trees, random state is given to test values for same shuffle data.

**3.5 Building and testing the model**

After understanding the model, we need to train and test the model to achieve better performance results. Model complexity is one of the factors that affect performance. The complexity of the model decides the overfitting, underfitting, appropriate fitting decisions made by the model. Overfitting occurs when your model learns the training data too well and incorporates details and noise specific to your dataset. A model is overfitting when it performs great on your training/validation set but poorly on the test set (or new real-world data). Underfitting occurs when your model over-generalizes and fails to incorporate relevant variations in your data that would give your model more predictive power. A model is underfitting when it performs poorly on both training and test sets. Best Fit /Appropriate fit occurs when your model actualizes and learns the patterns of data to incorporate relevant variations in your data that would give your model more predictive power. A model is Best /Appropriate fitting when the model performs excellently in both training and testing datasets. Overfitting, Underfitting and Appropriate fitting are explained in figure 3.7.



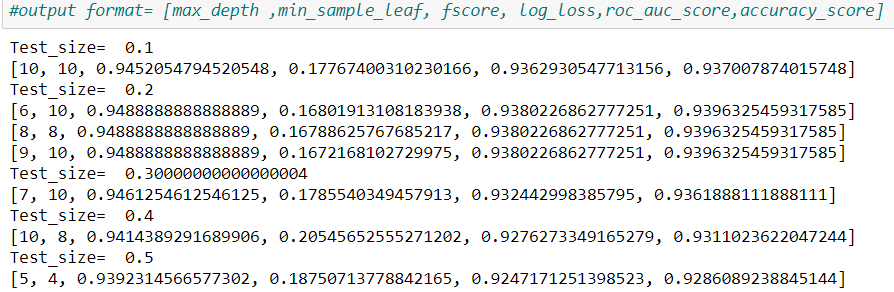
***Fig:3.7:*** *Model complexity factors*

We need a model to appropriate fit where it can go wrong in some values in prediction but gets the good majority of it. The parameters which affect the performance of my model when they variate are max\_depth and min \_sample\_leaf. If max\_depth increases, model complexity increases, and min\_sample\_leaf decreases, model complexity increases. So, in terms of model complexity, max\_depth is directly proportional, and min\_sample\_leaf is inversely proportional.

α Max depth

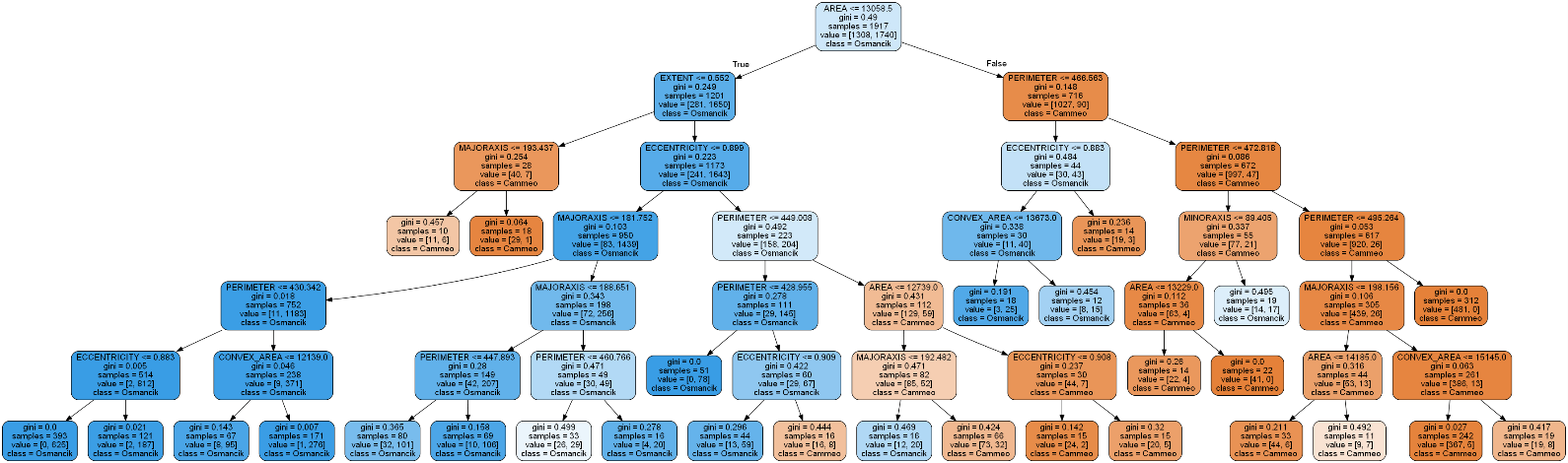
α

For testing the hyperparameters max depth and min sample leaf, we build a function called **"bestmodel,"**which tests the model by changing parameters inversely and fed to the model, and performance is tested. For evaluating performance, we are taking f1\_score, log loss,roc\_auc\_score, and accuracy\_score as our performance metrics of the best model in which mainly f1\_score is considered for performance. f1\_score and other performance metrics are explained in further chapters. bestmodel returns the max\_depth and min\_sample\_leaf, where f1\_score is better from all iterations. I tested this model at different test sizes got different values of hyperparameters which are listed in figure 3.8



***Fig:3.8:*** *Results of the best model at different test sizes*

From the results of best model we have best results at test size=0.2 with max\_depth=6, min\_sample\_leaf=10 as hyperparameters with performance of f1\_score =0.9488 , log loss=0.168 ,roc\_auc\_score=0.9380 , and accuracy\_score =0.9396. A first decision tree sample of RF model has been visualized shown on figure 3.9.



***Fig:3.9:*** First decision tree sample of RF model

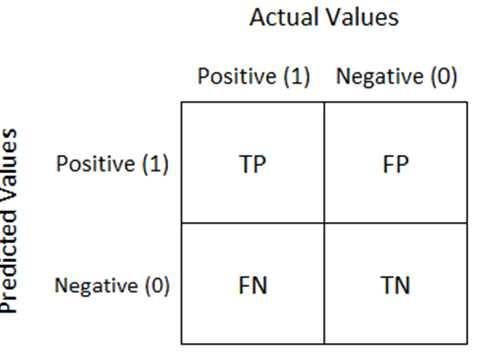
**3.6 Performance metrics**

Many learning algorithms have been proposed to date. It is generally expected that it is essential to evaluate the efficiency of an algorithm. Individuals even ended up making their metrics that suit the application. Performance metrics are often used to test and evaluate the model performance based on the requirement. For this project, the problem type is classification. In this review, we see the absolute most common measurements in a classification setting of a problem. Possible Performance metrics for classification problems are [[10]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007.)

* + - Confusion Matrix
    - F1score (balances the precision and recall)
    - Log loss
    - Accuracy score
    - ROC curve
    - ROC-AUC score (Area Under Curve in ROC Graph)

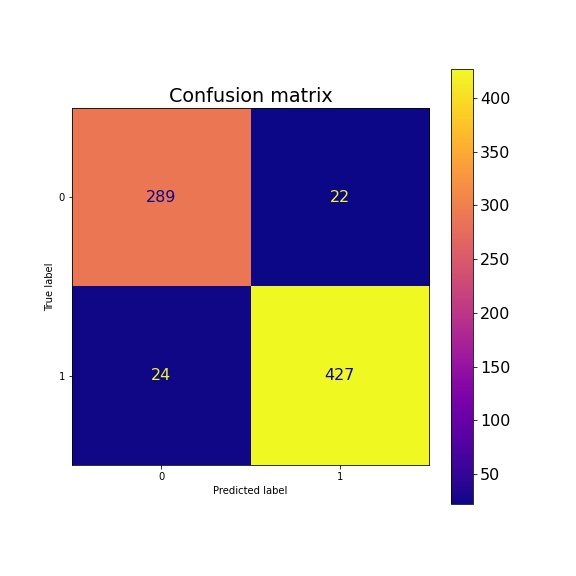
***3.6.1 Confusion Matrix***

A confusion matrix is a technique used for the performance of a classification algorithm. A confusion matrix is used to evaluate the accuracy of classification. By definition, a confusion matrix C is such that Ci,j equals the number of observations known to be in group i and predicted to be in group j. Thus in binary classification, the count of true negatives is C0,0, false negatives are C1,0, true positives are C1,1, and false positives are C0,1 [[5]](https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62)[[10]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007.)[[11]](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html). The tp, fp, tn, fn are shown in figure 3.10



***Fig:3.10:*** *Structure of confusion matrix for binary classification*

Python has a module called seaborn [[12]](https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea) which is used for color-separated visualizations. It is the advanced version of matplotlib.pyplot. With hyperparameters from bestmodel, the Random Forest model gives the confusion matrix result when tested with testing data, is plotted using seaborn as shown in figure 3.11.



***Fig:3.11:*** *Confusion matrix for testing data*

The Confusion Matrix has four parameters, as seen in figure 3.10. They are:

• tp: true positive

• fp: false positive

• fn: false negative

• tn: true negative

The confusion matrix structure is a 2 x 2 matrix for binary classification because of tp, fp, tn, fn. Performance measurements such as Accuracy, Sensitivity, Specificity, Precision, F1-Score, Negative Predictive Value, False Positive Rate, False Discovery Rate, False Negative Rate are calculated using figure 3.10. Calculation formulas are given in Table 3.9.

**Table 3.9:** Performance Measurements

|  |  |  |
| --- | --- | --- |
| Sno. | Performance measure | Formula |
| 1 | Accuracy |  |
| 2 | Sensitivity |  |
| 3 | Specificity |  |
| 4 | Precision |  |
| 5 | F1-Score |  |
| 6 | Negative Predictive Value |  |
| 7 | Negative Positive Rate |  |
| 8 | False Discovery Rate |  |
| 9 | False Negative Rate |  |

* + The F1 score can be explained as a weighted average of precision and recall [[10]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007.).
  + It is calculated with tp,tn,fp, and fn obtained from the confusion matrix
  + **F1 score reaches its best at value 1 and worst score at 0.**

***3.6.2 Log loss***

Log loss is also known as cross-entropy loss function. The loss function used in logistic regression and extensions such as neural networks is the negative log-likelihood of a logistic model that returns output predictions probabilities for its training data actual output. The log loss is only defined for two or more labels. For a single sample with true label  and a probability estimate  the log loss is defined as [[10]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007.)[[13]](file:///D:\COLLEGE\SEM%205\PT-1-Machine%20Learning\Sklearn.metrics.log_loss.%20scikit.%20(n.d.).%20Retrieved%20November%205,%202021,%20from%20https:\scikit-learn.org\stable\modules\generated\sklearn.metrics.log_loss.html):

By calculating log loss with RF model that has been developed with hyperparameters that gives a value of 0.168

***3.6.3 Accuracy score***

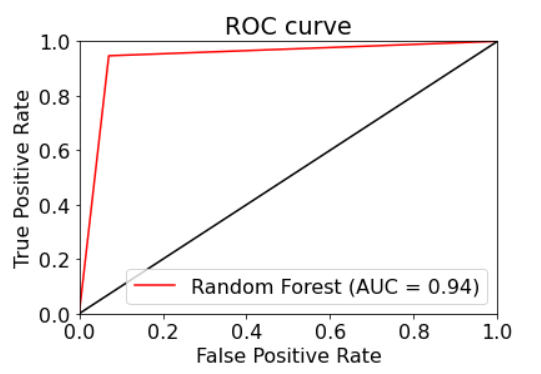
Accuracy score is the measurement that tests the model's performance by an average of output prediction vs. actual output [[10]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007.)[[14]](file:///D:\COLLEGE\SEM%205\PT-1-Machine%20Learning\Sklearn.metrics.accuracy_score.%20scikit.%20(n.d.).%20Retrieved%20November%206,%202021,%20from%20https:\scikit-learn.org\stable\modules\generated\sklearn.metrics.accuracy_score.html).

* + It ranges from 0 to 1, where 0 is worst, and 1 is Best.
  + It is often multiplied by 100 to get the accuracy percentage

The accuracy score of the RF model with hyperparameters is **93.96%.**

***3.6.4 ROC AUC curve***

ROC curve is plotted against True Positives VS False Positives. Greater the area under the curve(AUC) better the model. The roc curve of the RF model is shown in figure 3.12.



***Fig:3.12:*** *ROC curve of RF model*

***3.6.5 ROC AUC score***

Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

* + It is used to find Area Under Curve(AUC) in the ROC graph.
  + It ranges from 0 to 1, where 0 is worst, and 1 is Best

ROC AUC score of the RF model with hyperparameters is **0.94.**

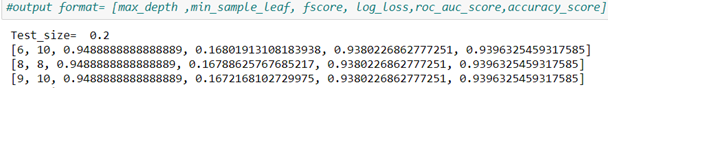
Chapter-4

**RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS**

Characterization of the rice varieties utilized in our review, preliminary processing was applied to the photos acquired with Computerized Vision Systems, and a total of 3810 rice grains were obtained. Besides, seven morphological components have been induced for each grain. A dataset has been made for the properties acquired. A model has been created and tested performance using Random forest classifier (RF) [[7]](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) machine learning techniques for classification.

**4.1 Analyzing performance**

To test the hyperparameters max depth and min sample leaf, we build a function called bestmodel that tests the model by changing parameters inversely and fed to the model, and performance is tested. The results obtained from the function are in figure 3.8. We observe that at test size =0.2, we have three results in which we have the same f1\_score, roc\_auc score, accuracy, but there is some difference in log loss values for different max depth and min sample leaf shown in figure 4.1.



***Fig:4.1:*** *Best model result at 0.2 test size*

In that third value of log loss of most minor, we know that smaller the value better the model performance, but we are choosing the first value of log loss because of the model complexity. In a dataset of 3810 values, the difference b/w first and last log loss values are small, but in large datasets where the dataset size is over 10,000 to 1 lac, the log loss matters, but model complexity is also necessary. With higher max depth, the model complexity increases; therefore, overfitting of the results happens. Overfitting is explained in figure 3.7. Model complexity is the reason for choosing max\_depth=6, min\_sample\_leaf=10 as model hyperparameters to gain the best results.

**4.1 Final result**

After testing the model hyperparameters, the final result parameters used to build the model are mentioned in table 4.1.

**Table 4.1:** Random Forest Model parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Random state** | **Test size** | **Max depth** | **Min sample leaf** |
| 12 | 0.2 | 6 | 10 |

For this classification problem, the performance metrics calculated for the model are Confusion Matrix, f1\_score, logloss, accuracy score, roc\_auc score. These performance calculations are presented in table4.2.

**Table 4.2:** Results of random forest model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion matrix** | **F1score** | **Log loss** | **Roc Auc score** | **Accuracy** |
| |  |  | | --- | --- | | 289 | 22 | | 24 | 427 | | 0.9488 | 0.16801 | 0.94 | **93.96%** |

Chapter-5

**SUMMARY AND CONCLUSIONS**

**5.1 Summary of the project**

In this study, a qualitative methodology has been used to reach our aim. Rice type classification problem arises when there is a demand for rice, based on its features like physical structure, appearance, cooking qualities, and more. The main feature among them is physical appearance. The classification of rice becomes tedious when there are significant production units. Because of this reason, there has been seen a requirement of technology to make machines to classify rice instead of human resources. Machine learning algorithms can solve most of our daily problems at ease. ML algorithms make a perfect solution for our problem of classifying rice using the machine. Training machines to identify things is called machine learning, and an ML algorithm does the learning.

Random forest classifier is one of the ML algorithms which can solve this problem. This algorithm uses the bagging method, which means considering the results of different ML algorithms to decide for the given input. The random forest algorithm consists of decision trees built on the inputs of the problem. While algorithm building the decision tree its takes one or more features as the main priority and makes decision tree. In this way, by making different features its main priority, the model finally makes some number of DTs as requested to decide for the given input after it has been trained. These decision outputs are considered by the RF algorithm and give output as the majority voted output [[1]](https://scikit-learn.org/stable/modules/ensemble.html#bagging-meta-estimator.).

To solve this classification problem, we need to build a dataset of rice that is used to train the RF algorithm for classifying the rice. Dataset can be prepared by computerized image vision technology used to scan images of the rice and extract its morphological features to make the machine understand differences in rice physical structure. These extracted features are stored in comma-separated value(.csv) format. These self-prepared or computerized datasets may not always be perfect, and they consist of missing values, fault symbols. We need to understand the dataset and clean it to make it easier for machines to handle these errors. This process is called data preprocessing.

Data preprocessing can be done using integrated python pandas [[6]](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html) libraries in a jupyter notebook. After understanding and cleaning data, there is a need for understanding the relations between the features and how these relations affect our aim. for understanding relations, a dataset can be visualized using matplotlib.pyplot [[4]](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html.) module and its techniques [[2]](https://scikit-learn.org/stable/visualizations.html.) inside jupyter notebook. With these visualizations, we can understand data clearly and build a model according to our requirements. After data preprocessing and understanding, we need to study ML algorithms to build the model.

Random forest algorithm has some tuning parameters called hyperparameters [[7]](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.). These hyperparameters can be changed as per the requirement to achieve better performance. Some parameters are assigned once according to the requirement in these hyperparameters and left unchanged for the rest of the testing. The parameters that are changed for testing are max\_depth and min\_sample\_leaf. In terms of model complexity, max\_depth is directly proportional, and min\_sample\_leaf is inversely proportional. To test these parameters, we can build the python function that feeds the parameters inversely to the model and test the performance. The function's output is the result of the model is best f1\_score and the parameters that have been built. With these parameters, we can build the model for rice type classification. But there is a need to validate the performance according to the requirements.

Performance metrics of classification problems help's us to validate the performance of the model. The performance metrics used are confusion matrix, f1\_score,logloss, accuracy\_score, roc\_auc\_curve, roc\_auc\_score. With these, we can validate our model the requirements of the problem.

At the end of this study, we achieved an accuracy of **93.96%** with the dataset of rice. By looking at the results when that kind of success rate was obtained, it is possible to say that the study achieved success. To understand this algorithm, we need to know python, sklearn libraries, and ML technics. This Random Forest model can be modified and installed into physical machines for classification. This model is binary classification, but this model can further be remodeled and used for other classification problems.

**5.2 Conclusion of the project**

The random forest model is validated with different performance metrics in which we took f1\_score as our primary metric to test performance. The f1\_score is calculated by balancing both precision and recall factors. After developing the model with better hyperparameters, we should validate with other performance metrics. With a performance metric accuracy score, we can test model accuracy against training data. Visualizing performance makes us understand the loose ends of the algorithm.

The Rice type classification model for machines in rice industries using random forest ML algorithm has been successfully developed with 93.96% accuracy. This RF model can be implemented in the machines to classify the rice types.

**REFERENCES**

[1] 1.11. ensemble methods. scikit. (n.d.). Retrieved November 2, 2021, from [https://scikit-learn.org/stable/modules/ensemble.html#bagging-meta-estimator](https://scikit-learn.org/stable/modules/ensemble.html).

[2] 5. visualizations. scikit. (n.d.). Retrieved November 2, 2021, from <https://scikit-learn.org/stable/visualizations.html>.

[3] The absolute basics for beginners. NumPy. (n.d.). Retrieved November 2, 2021, from <https://numpy.org/doc/stable/user/absolute_beginners.html>.

[4] Matplotlib.pyplot. matplotlib.pyplot - Matplotlib 3.4.3 documentation. (n.d.). Retrieved November 2, 2021, from <https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html>.

[5] Narkhede, S. (2021, June 15). Understanding confusion matrix. Medium. Retrieved November 2, 2021, from <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>.

[6] Pandas. dataframe. pandas.DataFrame - pandas 1.3.4 documentation. (n.d.). Retrieved November 2, 2021, from <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>.

[7] Sklearn.ensemble.randomforestclassifier. scikit. (n.d.). Retrieved November 2, 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.

[8] Sklearn.metrics.roc\_auc\_score. scikit. (n.d.). Retrieved November 2, 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html>

[9] Sklearn.metrics.roc\_curve. scikit. (n.d.). Retrieved November 2, 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html>.

[10] Vidiyala, R. (2020, July 26). Performance metrics for classification machine learning problems. Medium. Retrieved November 2, 2021, from <https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007>.

[11] *Sklearn.metrics.confusion\_matrix*. scikit. (n.d.). Retrieved November 5, 2021, from

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>.

[12] T, D. (2019, July 25). *Confusion matrix visualization*. Medium. Retrieved November 5, 2021, from <https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea>.

[13] *Sklearn.metrics.log\_loss*. scikit. (n.d.). Retrieved November 5, 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html>.

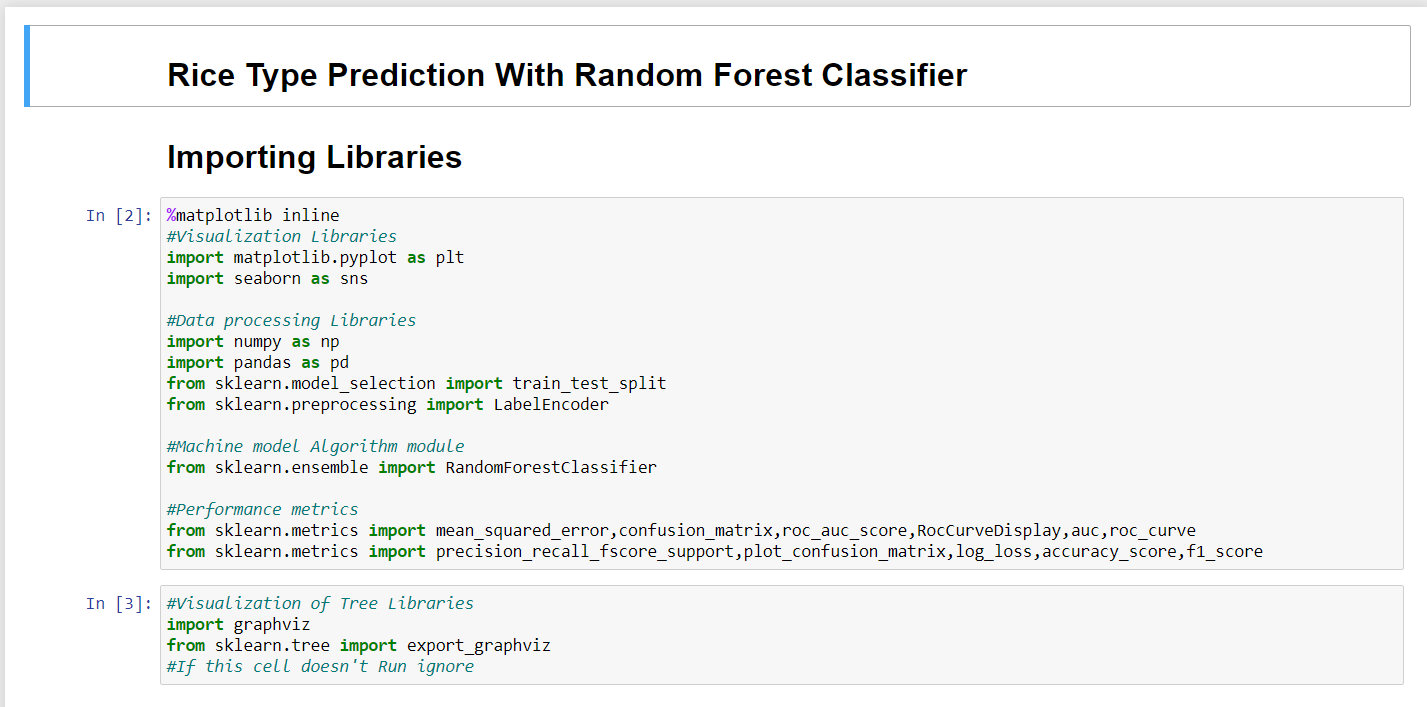
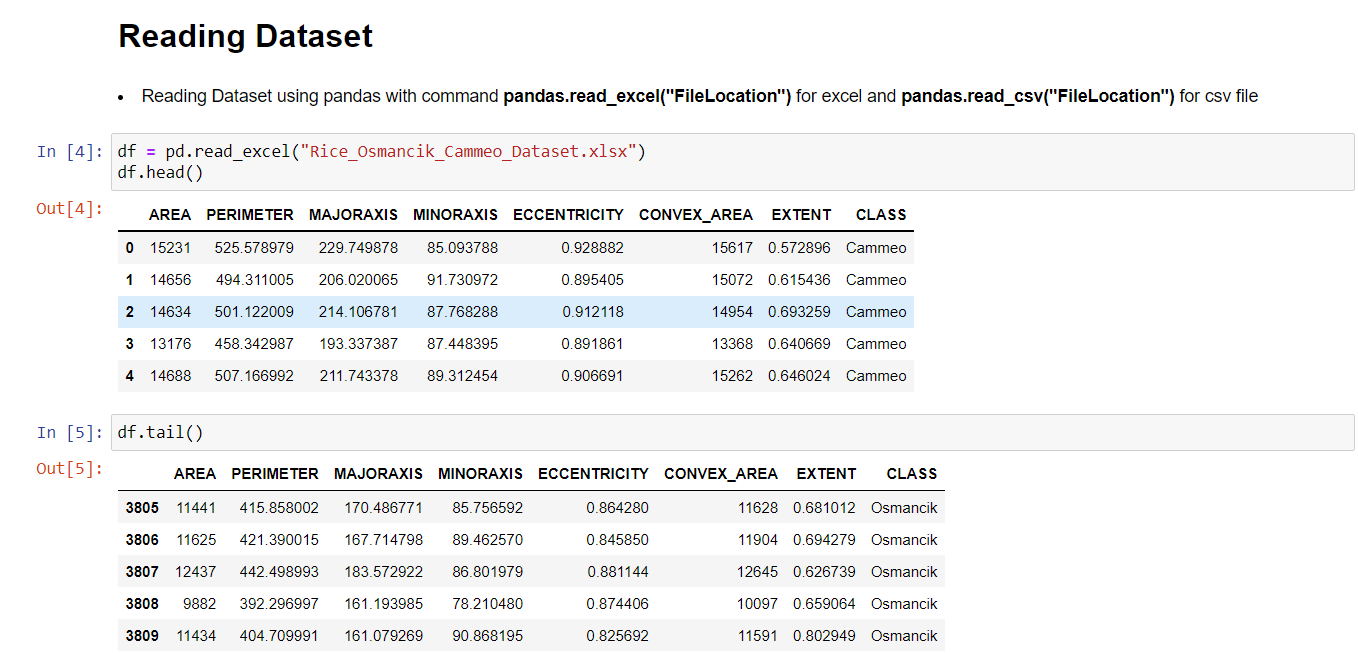
[14] *Sklearn.metrics.accuracy\_score*. scikit. (n.d.). Retrieved November 6, 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html>.

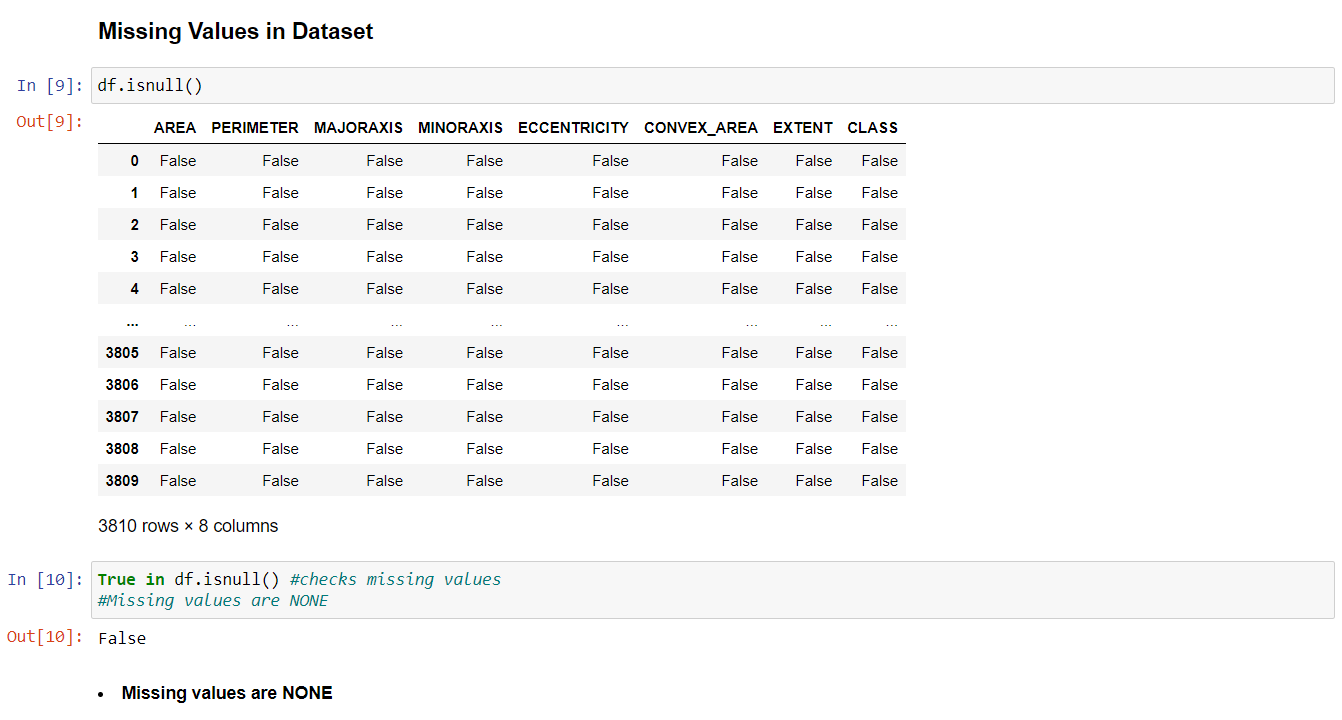
[15] *Pandas.dataframe.corr*. pandas.DataFrame.corr - pandas 1.3.4 documentation. (n.d.). Retrieved November 6, 2021, from <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html>.

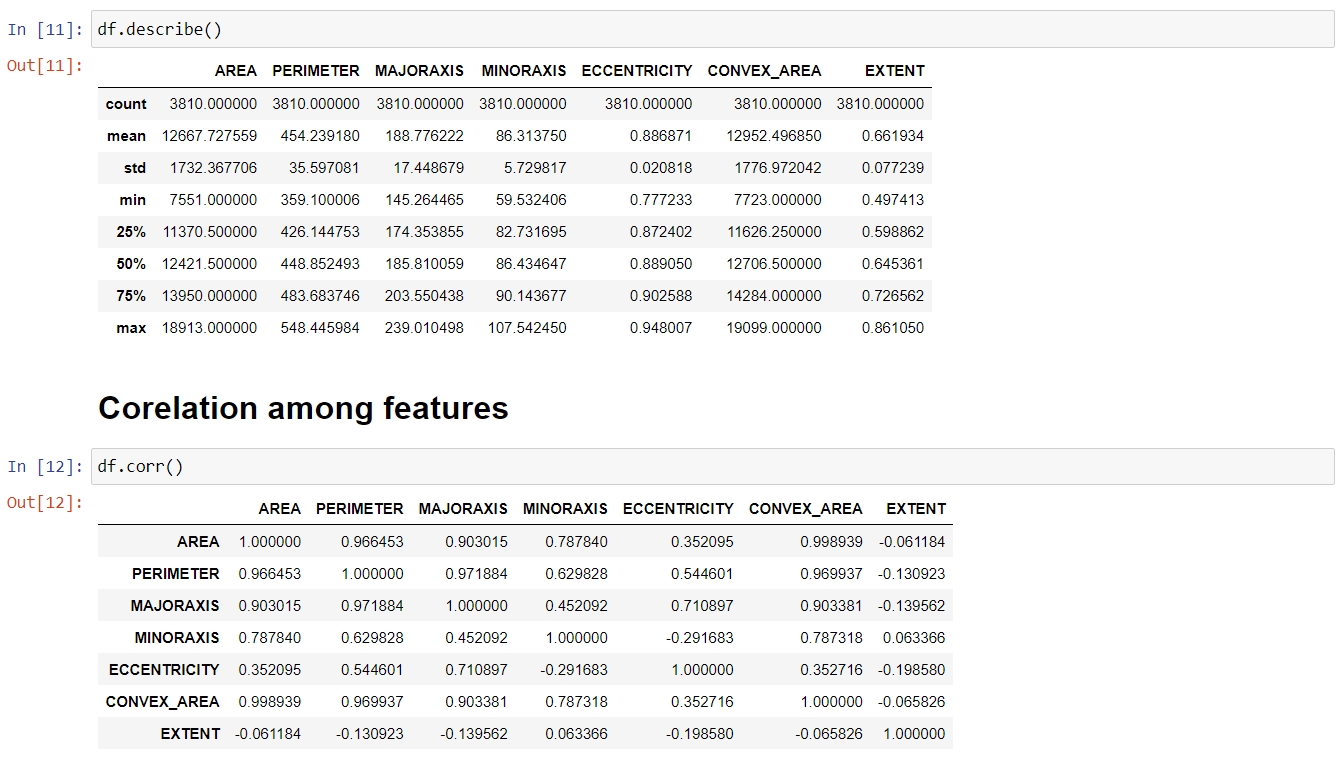
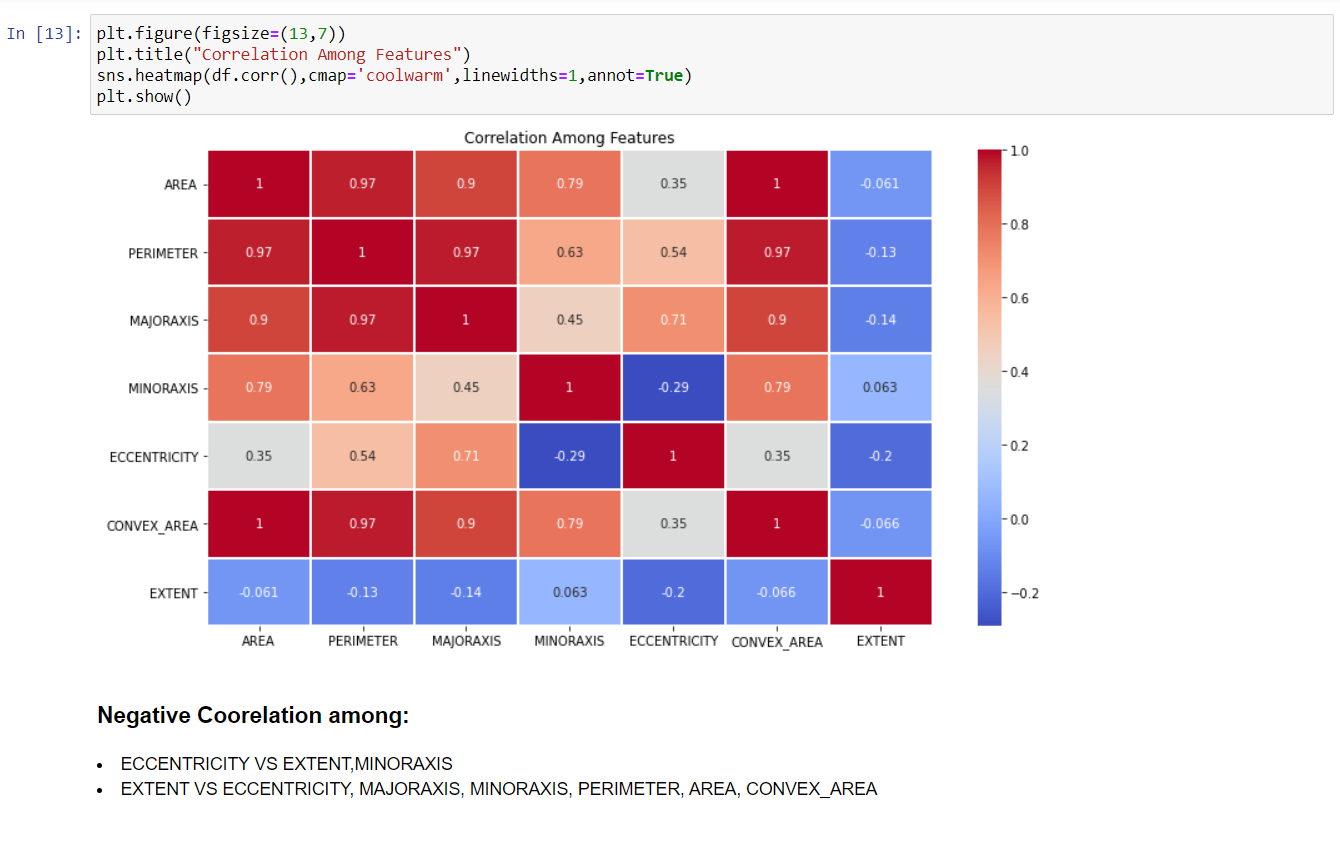
[16] Hayes, A. (2021, November 3). *What is correlation in finance?* Investopedia. Retrieved November 6, 2021, from <https://www.investopedia.com/terms/c/correlation.asp#:~:text=Correlation%20is%20a%20statistical%20term,they%20have%20a%20negative%20correlation>.

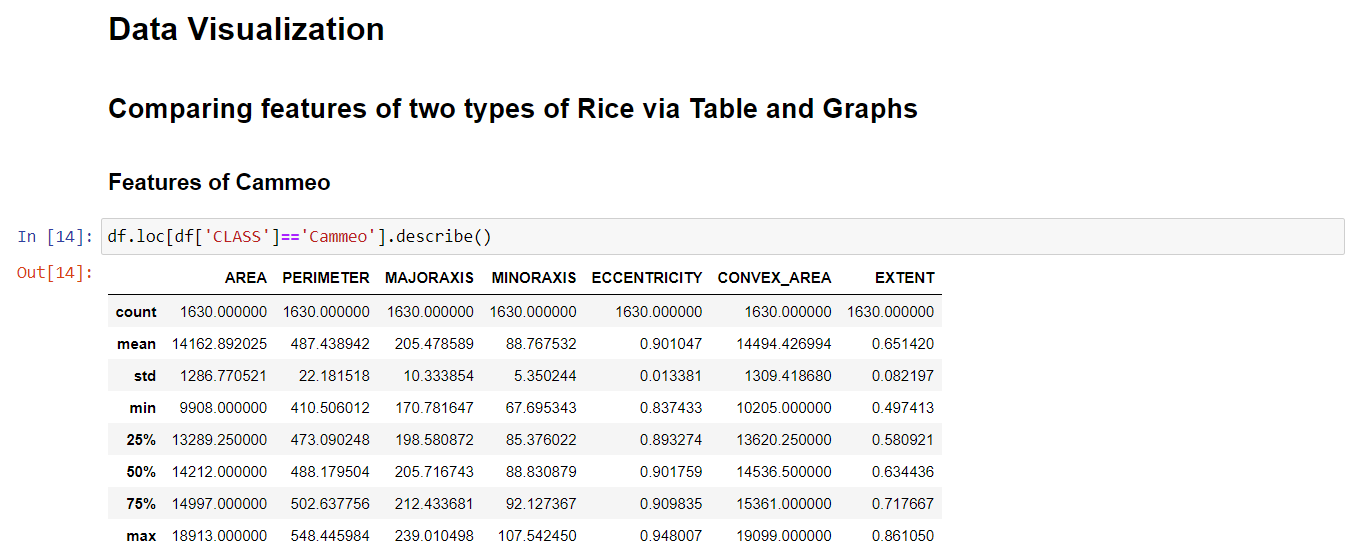
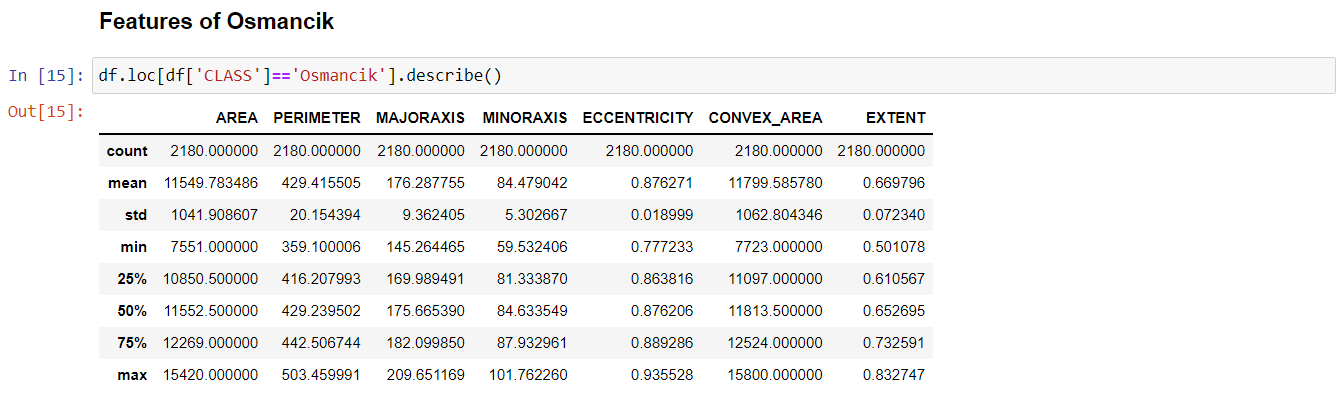
[17] *Sklearn.preprocessing.LabelEncoder*. scikit. (n.d.). Retrieved November 6, 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>.

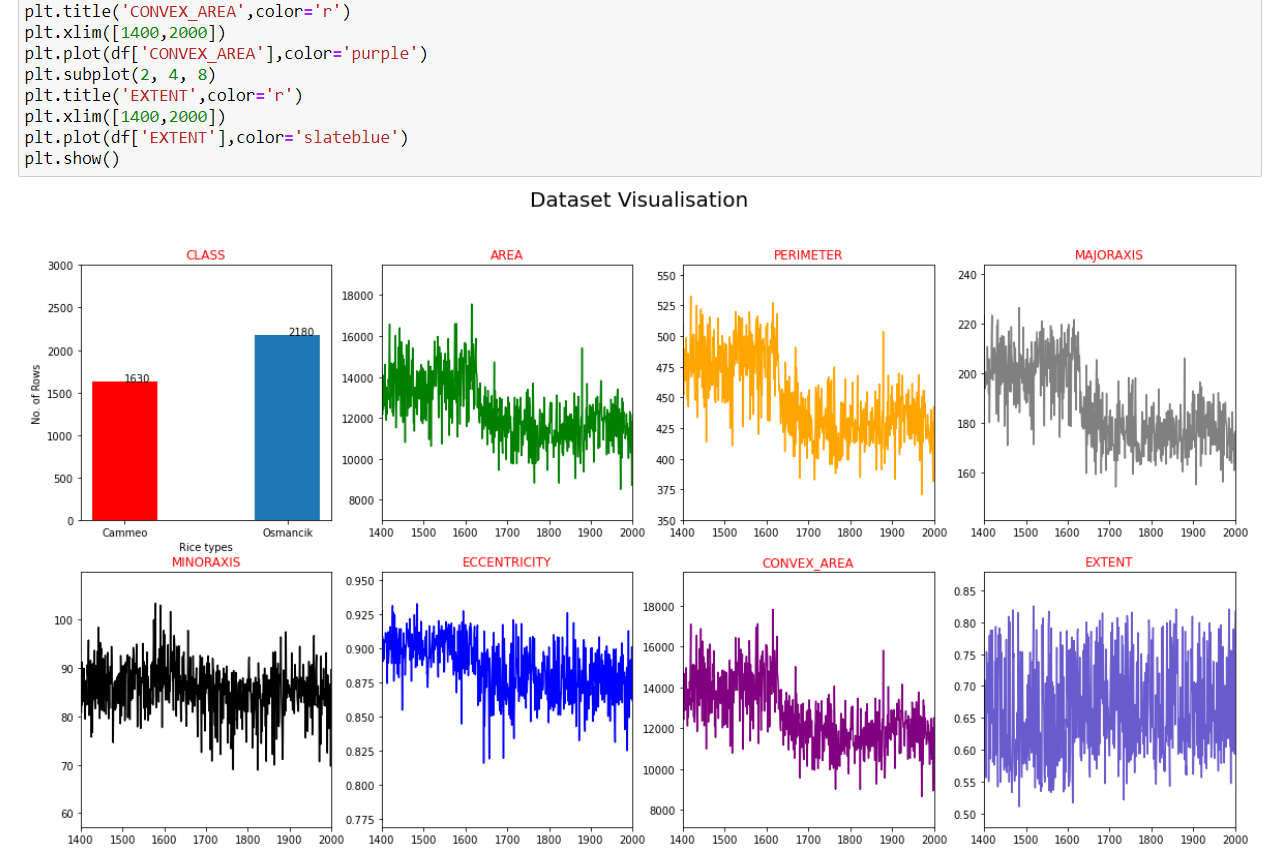
**APPENDIX**

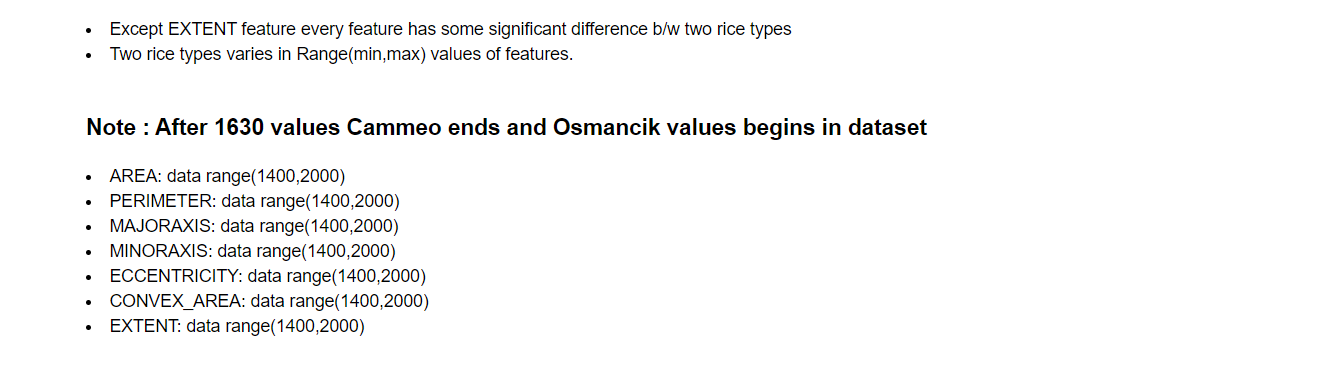
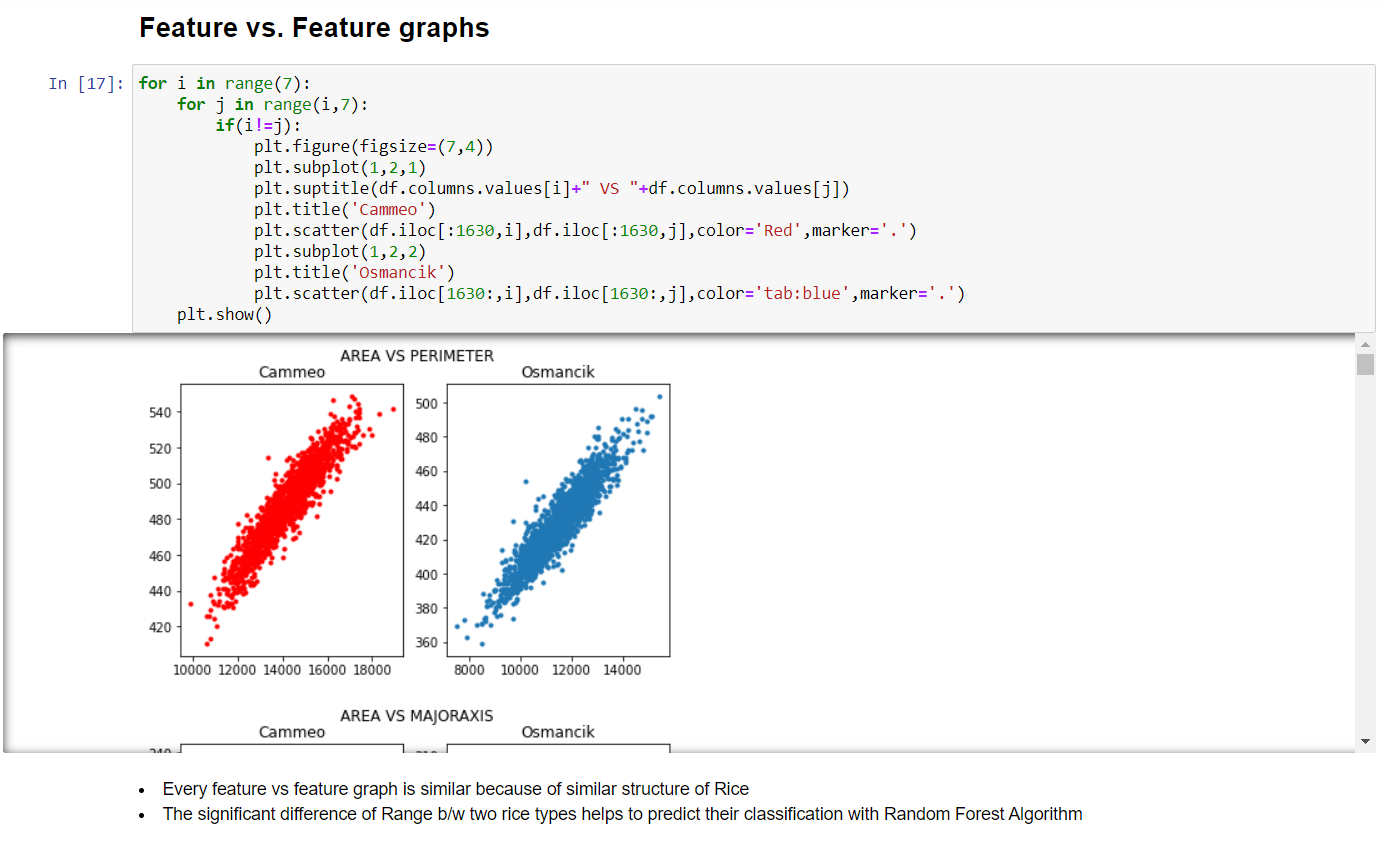
**A. SCREENSHOTS**

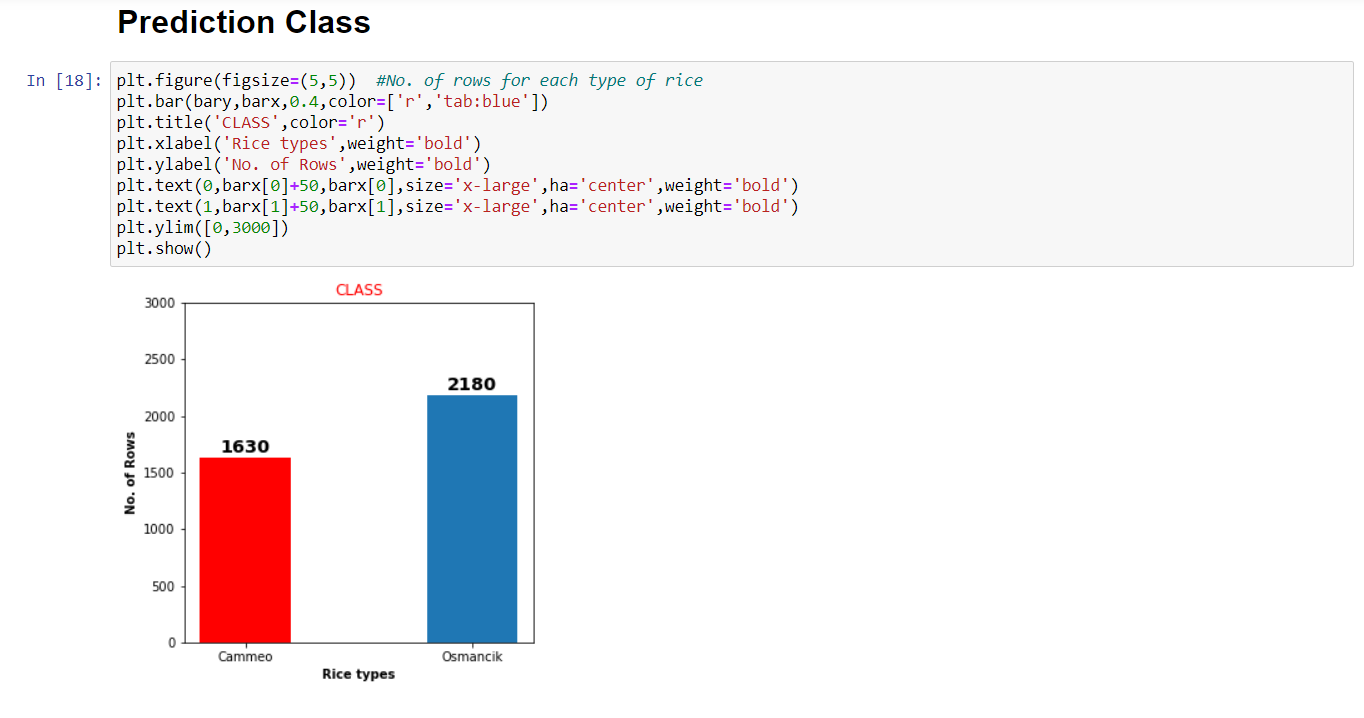
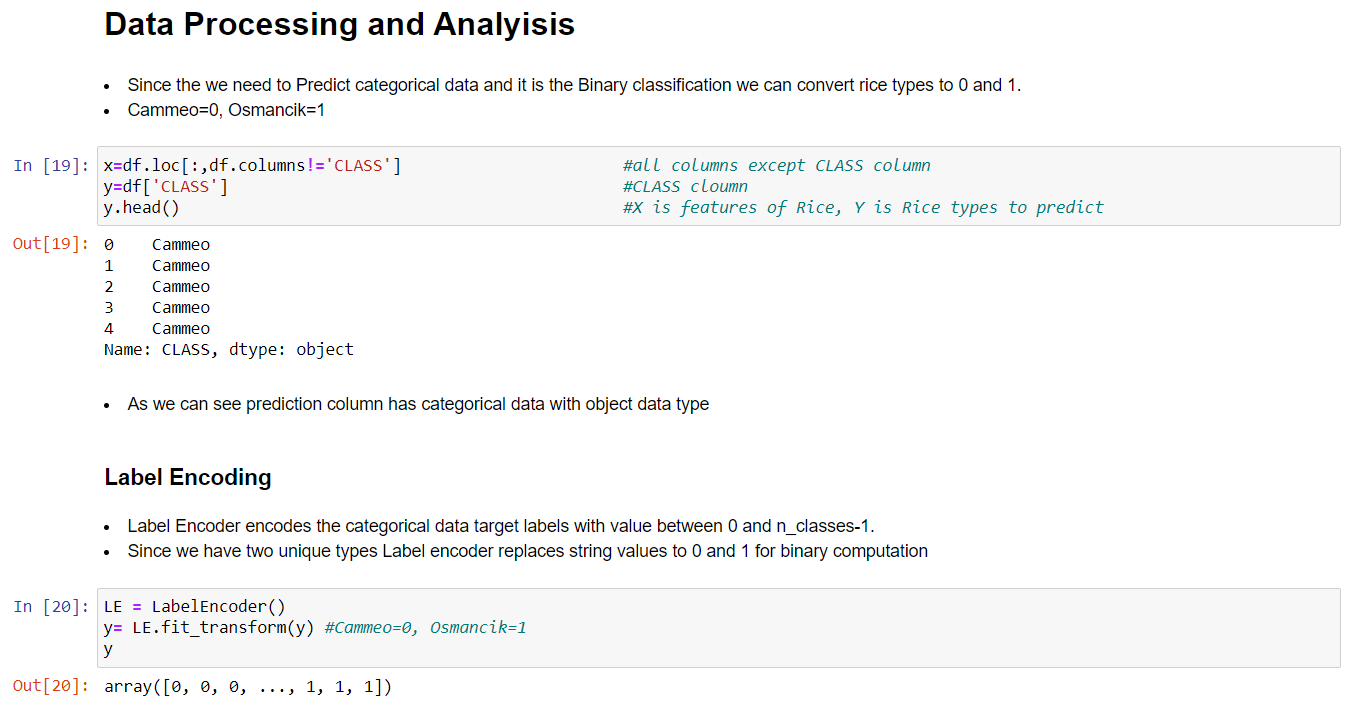
****

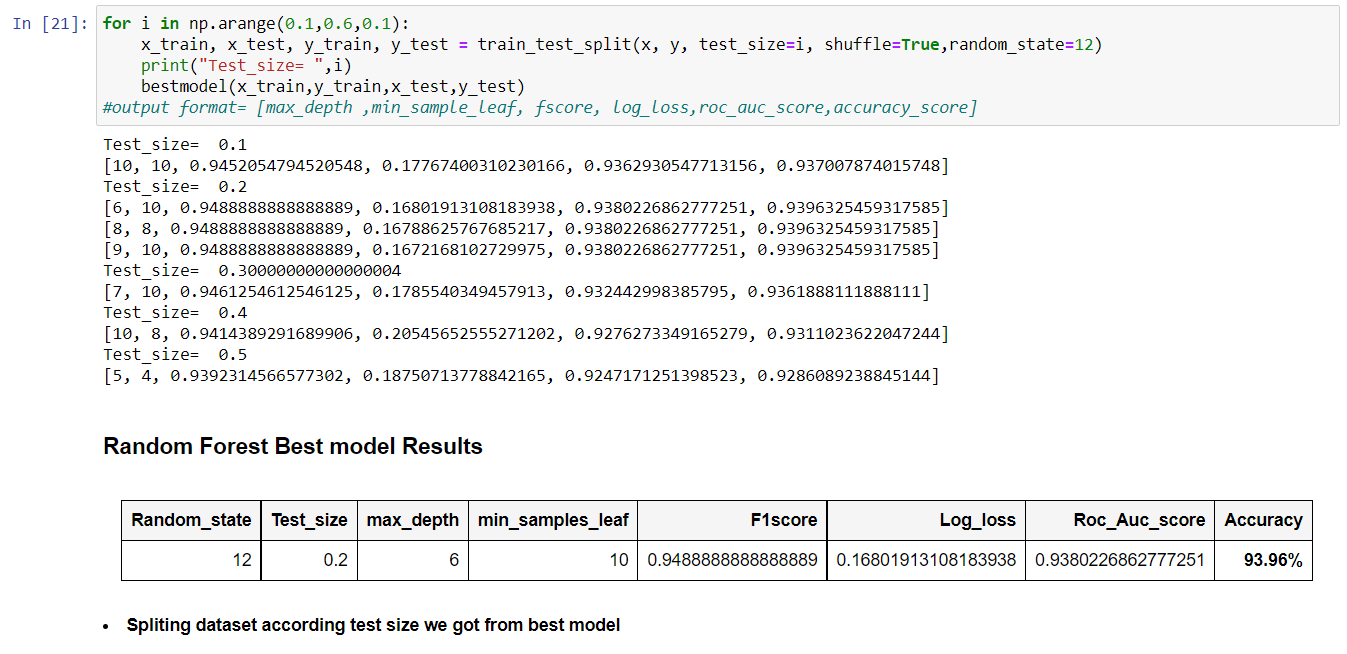
****

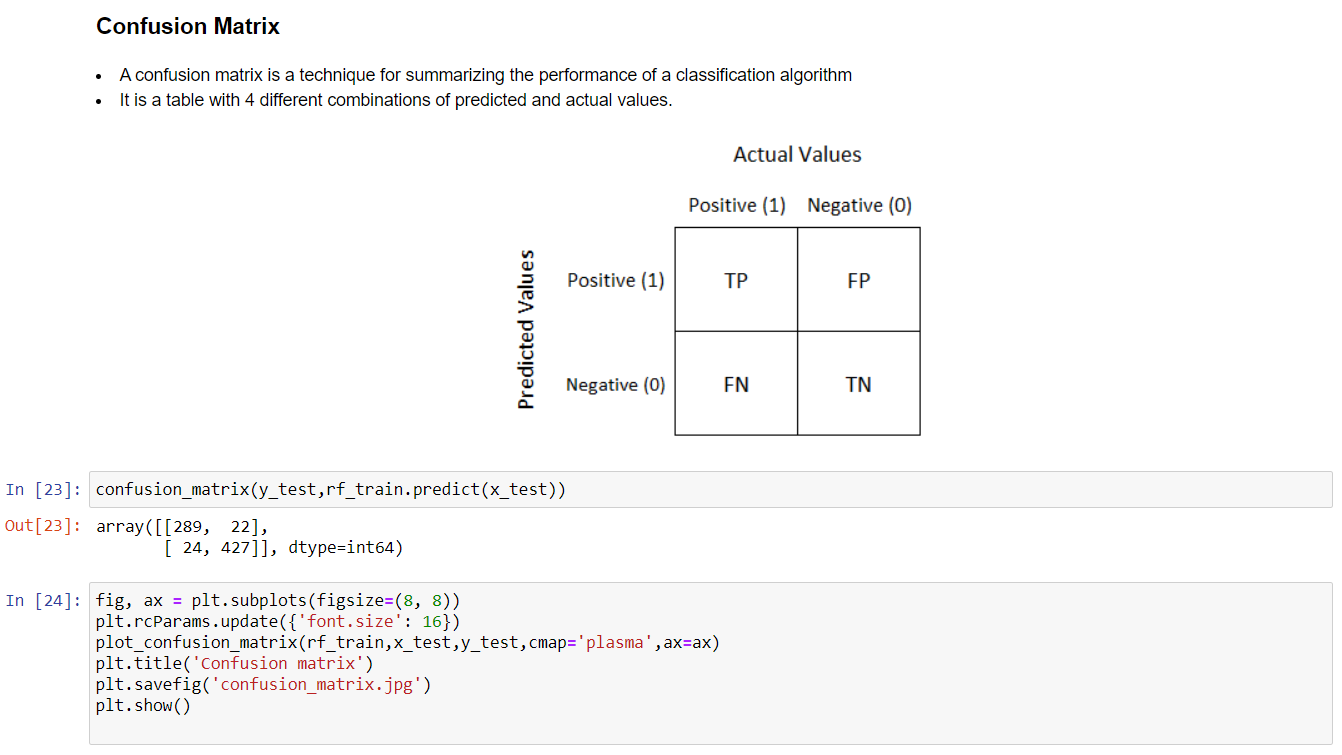
****

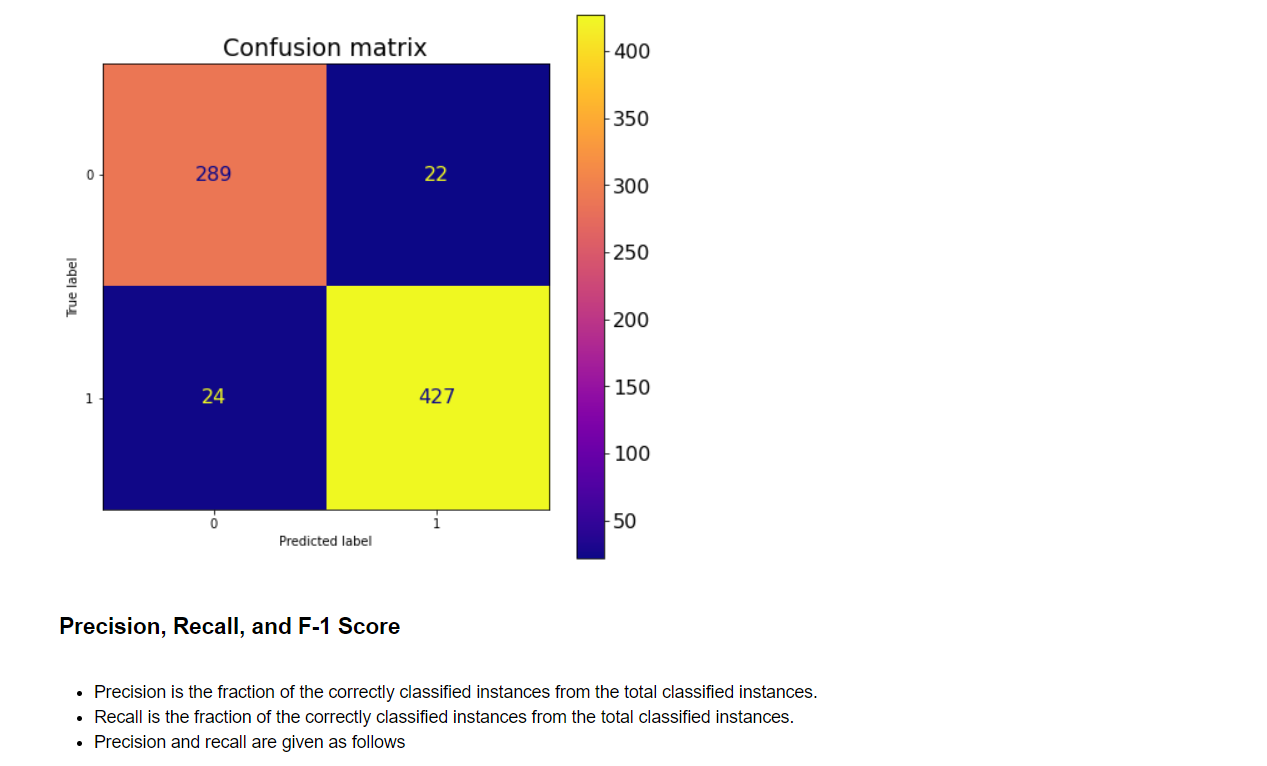
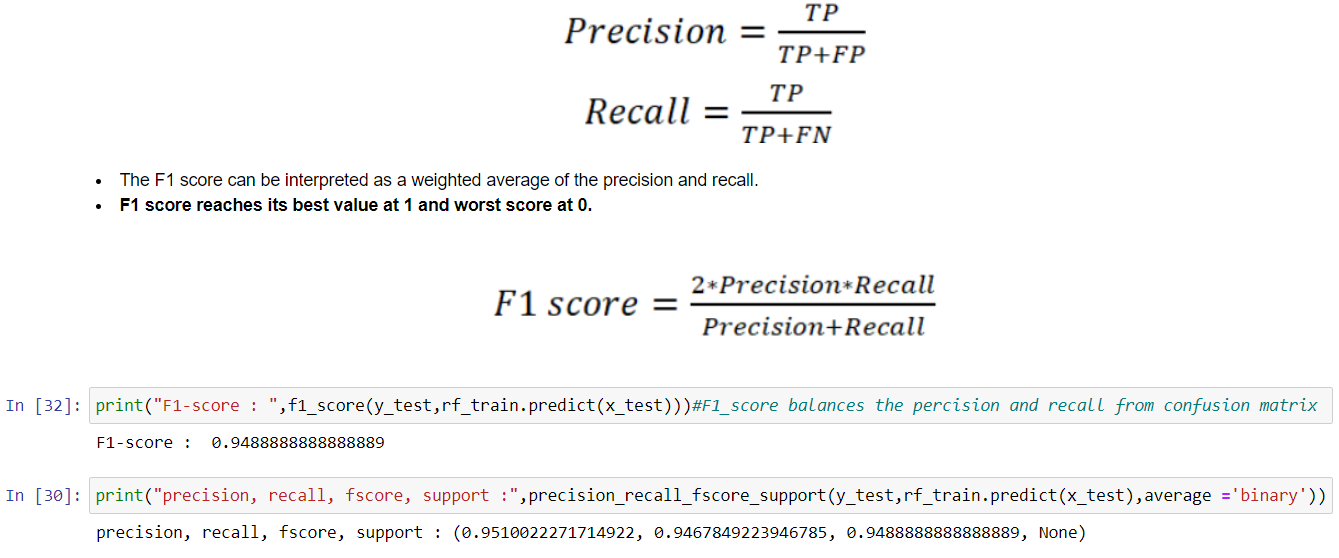
****

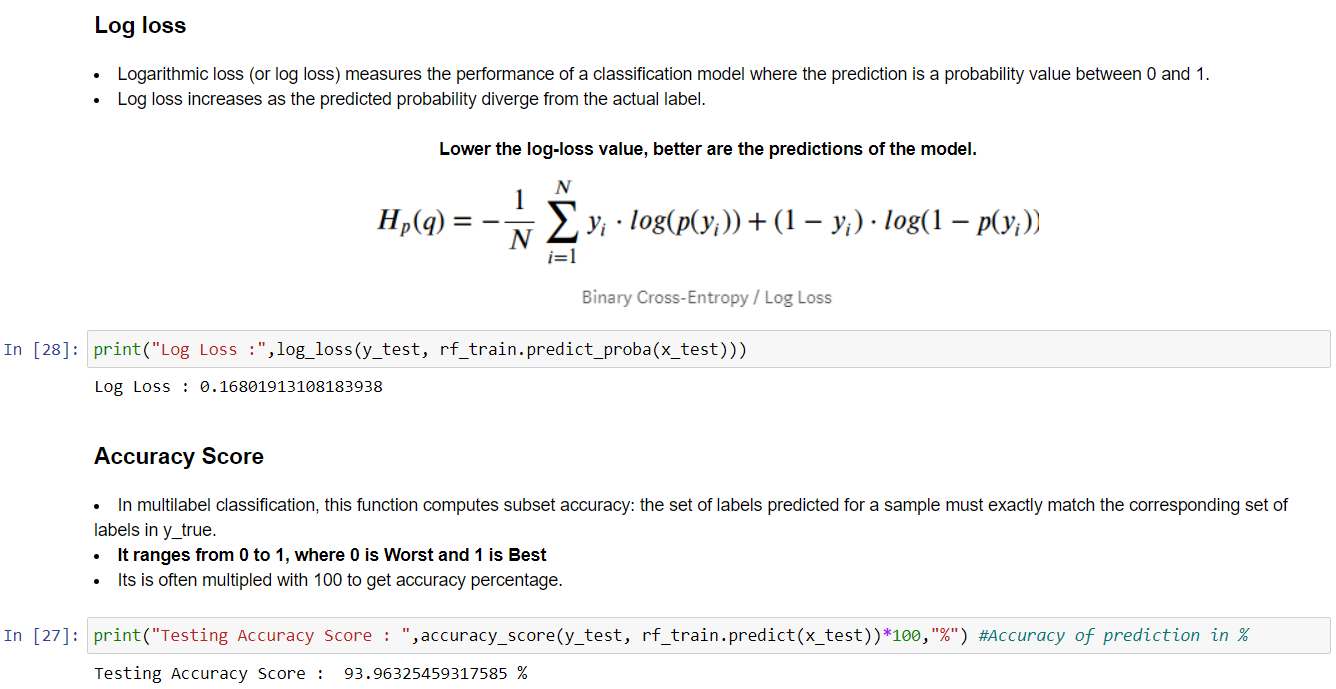
****

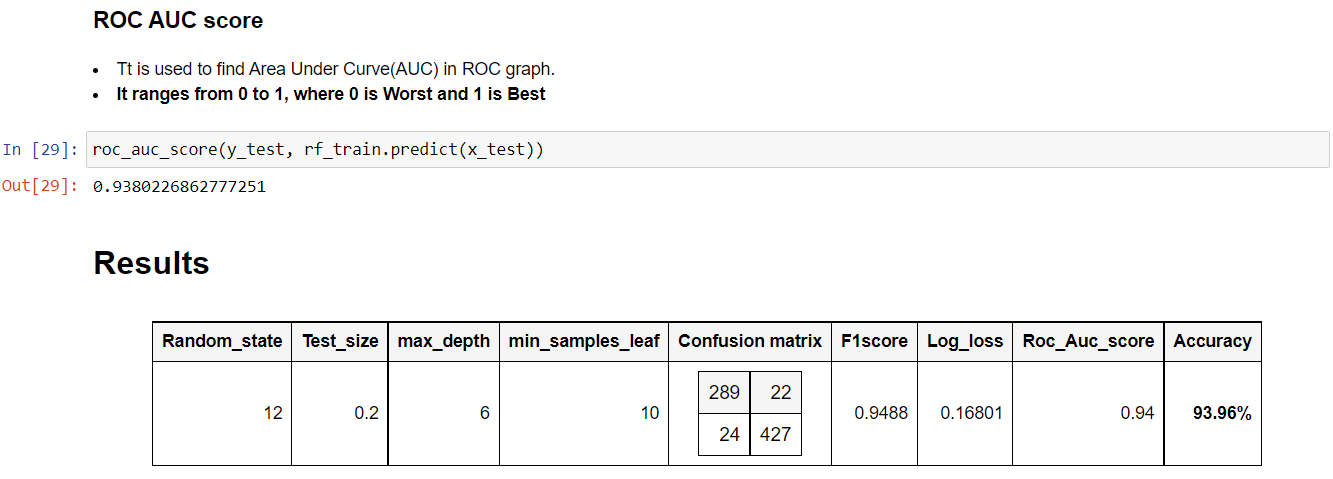
****

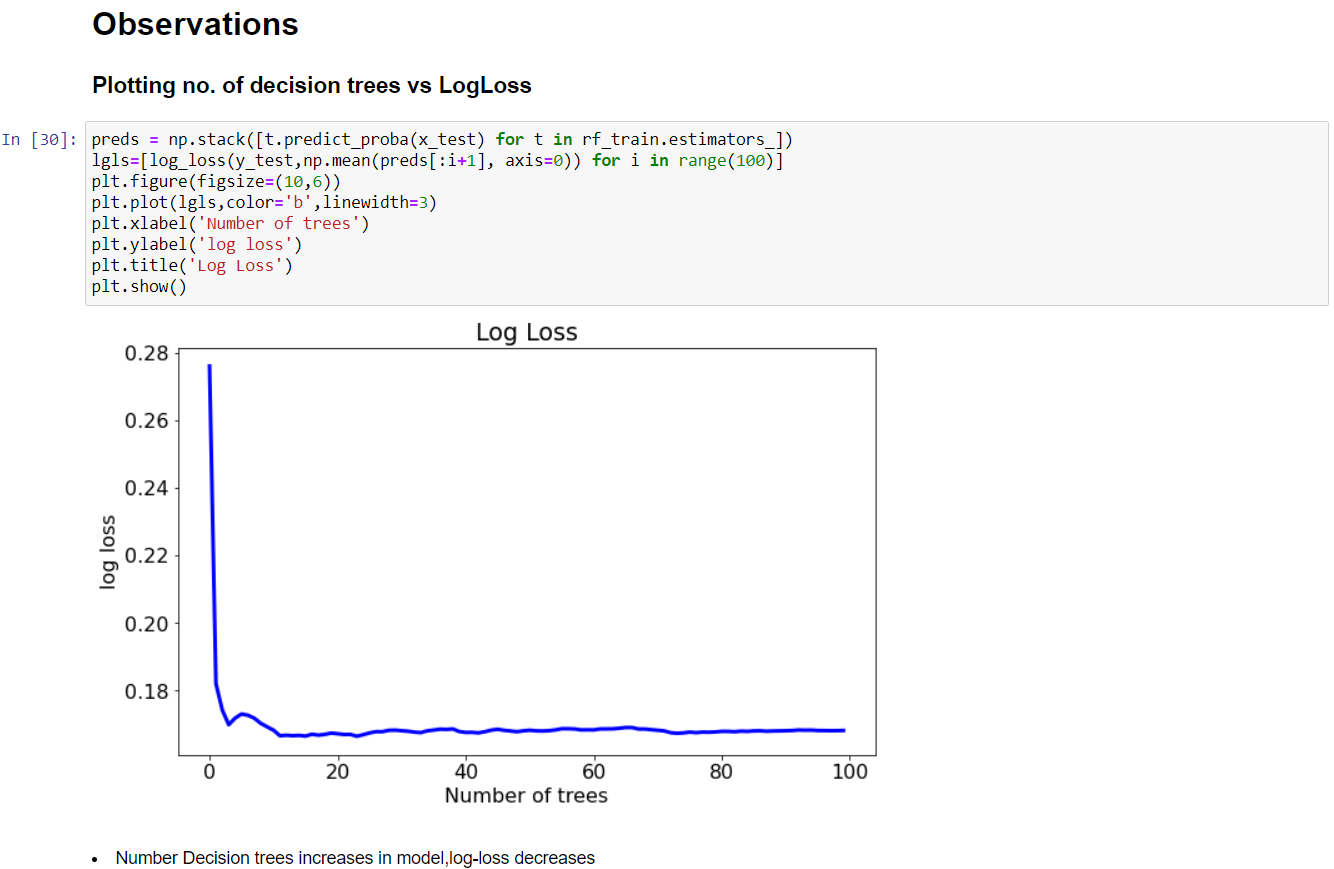
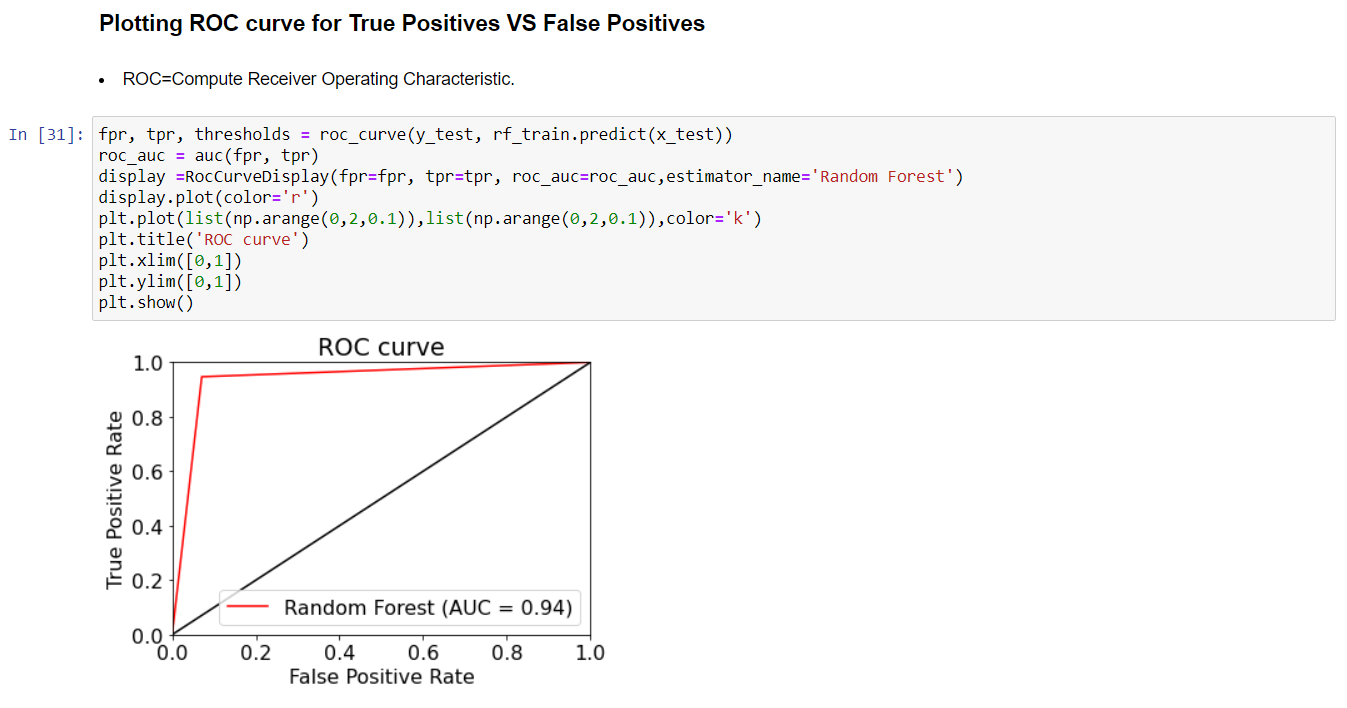
****

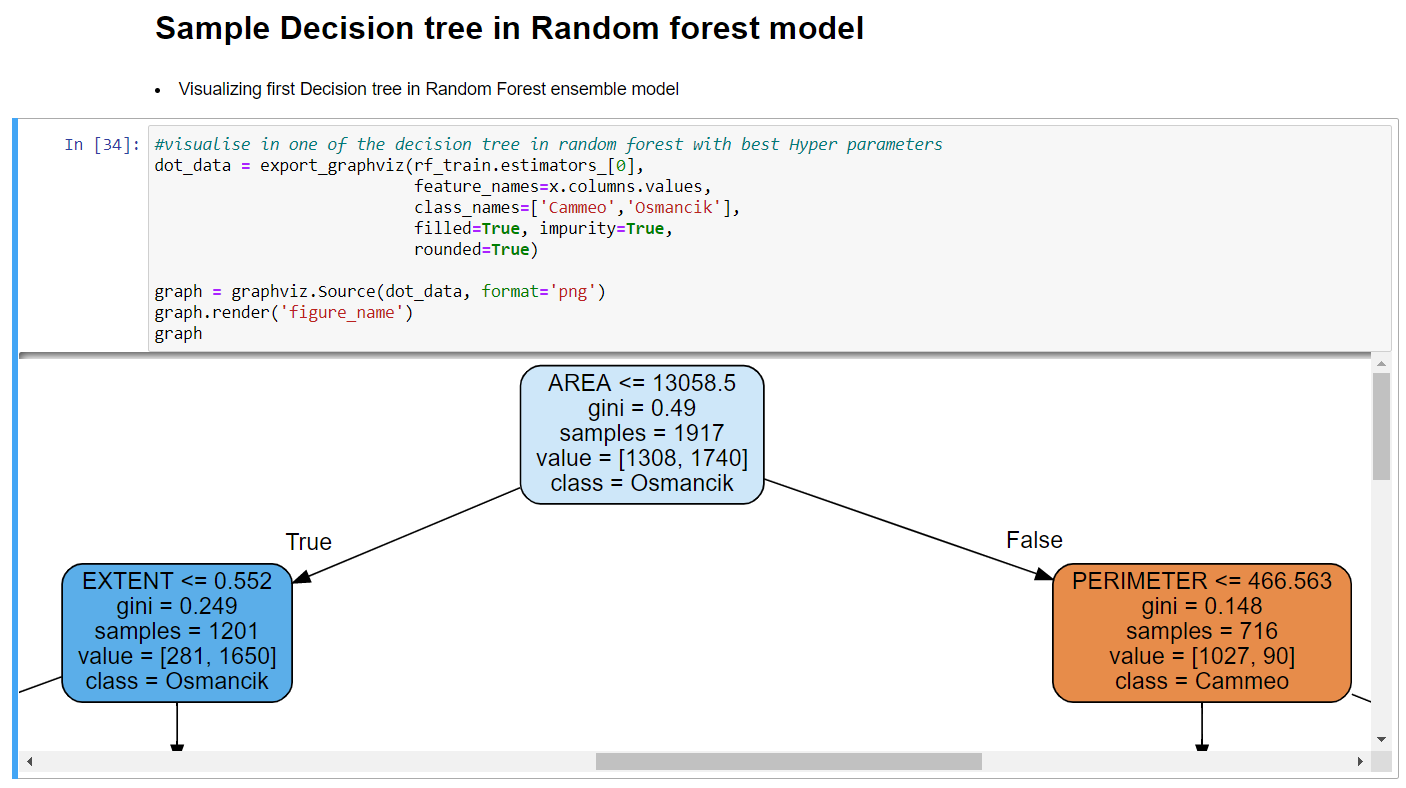
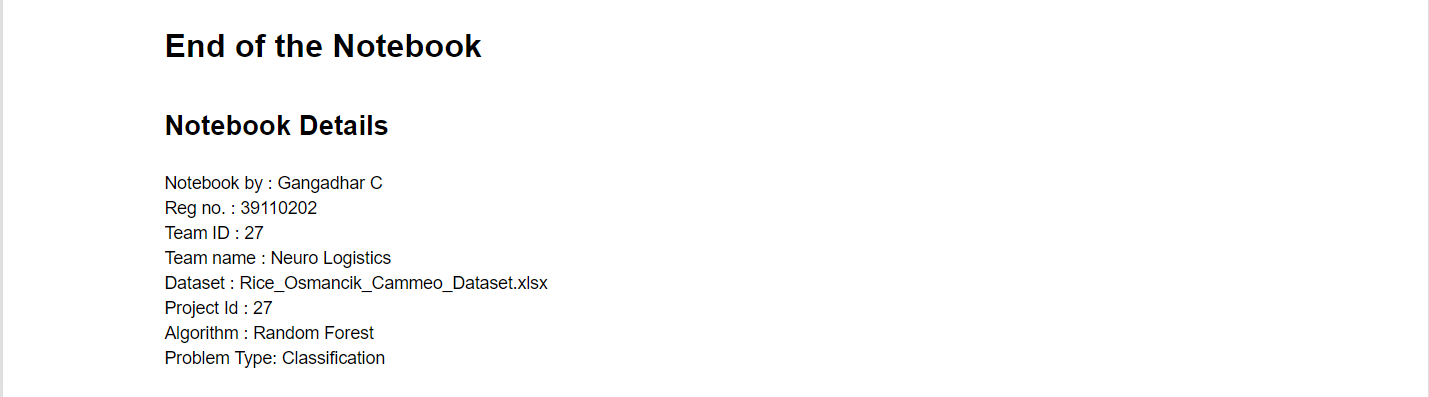
****

****

****

****

****

****

**B. SOURCE CODE**

|  |
| --- |
| %matplotlib inline  #Visualization Libraries  import matplotlib.pyplot as plt  import seaborn as sns  #Data processing Libraries  import numpy as np  import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import LabelEncoder  #Machine model Algorithm module  from sklearn.ensemble import RandomForestClassifier  #Performance metrics  from sklearn.metrics import confusion\_matrix,roc\_auc\_score,RocCurveDisplay,auc,roc\_curve  from sklearn.metrics import precision\_recall\_fscore\_support,plot\_confusion\_matrix,log\_loss,accuracy\_score,f1\_scorev  #Visualization of Tree Libraries  import graphviz  from sklearn.tree import export\_graphviz  #Reading Dataset  df = pd.read\_excel("Rice\_Osmancik\_Cammeo\_Dataset.xlsx")  df.head()  df.tail()  df['CLASS'].unique() #Two Rice Types 'Cammeo', 'Osmancik'  for i in range(len(df['CLASS'])):  if(df.iloc[i,7]!='Cammeo'):  print(df.iloc[i,7],i)  break    df.info()  df.isnull()  True in df.isnull() #checks missing values  df.describe()  #Corelation among features  df.corr()  plt.figure(figsize=(13,7))  plt.title("Correlation Among Features")  sns.heatmap(df.corr(),cmap='coolwarm',linewidths=1,annot=True)  plt.show()  #Data Visualization  ##Features of Cammeo  df.loc[df['CLASS']=='Cammeo'].describe()  ##Features of Osmancik  df.loc[df['CLASS']=='Osmancik'].describe()  ##Graphs for each Feature  barx=[len([0 for i in df['CLASS'] if(i=='Cammeo')]),len([1 for i in df['CLASS'] if(i=='Osmancik')])]  bary=['Cammeo','Osmancik']  plt.figure(figsize=(20,10))  plt.suptitle('Dataset Visualisation',size='20')  plt.subplot(2, 4, 1)  plt.title('CLASS',color='r')  plt.bar(bary,barx,0.4,color=['r','tab:blue'])  plt.xlabel('Rice types')  plt.ylabel('No. of Rows')  plt.text(0,barx[0],barx[0])  plt.text(1,barx[1],barx[1])  plt.ylim([0,3000])  plt.subplot(2, 4, 2)  plt.title('AREA',color='r')  plt.plot(df['AREA'],color='Green')  plt.xlim([1400,2000])  plt.subplot(2, 4, 3)  plt.title('PERIMETER',color='r')  plt.xlim([1400,2000])  plt.plot(df['PERIMETER'],color='Orange')  plt.subplot(2, 4, 4)  plt.title('MAJORAXIS',color='r')  plt.xlim([1400,2000])  plt.plot(df['MAJORAXIS'],color='grey')  plt.subplot(2, 4, 5)  plt.title('MINORAXIS',color='r')  plt.xlim([1400,2000])  plt.plot(df['MINORAXIS'],color='k')  plt.subplot(2, 4, 6)  plt.title('ECCENTRICITY',color='r')  plt.xlim([1400,2000])  plt.plot(df['ECCENTRICITY'],color='blue')  plt.subplot(2, 4, 7)  plt.title('CONVEX\_AREA',color='r')  plt.xlim([1400,2000])  plt.plot(df['CONVEX\_AREA'],color='purple')  plt.subplot(2, 4, 8)  plt.title('EXTENT',color='r')  plt.xlim([1400,2000])  plt.plot(df['EXTENT'],color='slateblue')  plt.show()  ##Feature vs. Feature graphs  for i in range(7):  for j in range(i,7):  if(i!=j):  plt.figure(figsize=(7,4))  plt.subplot(1,2,1)  plt.suptitle(df.columns.values[i]+" VS "+df.columns.values[j])  plt.title('Cammeo')  plt.scatter(df.iloc[:1630,i],df.iloc[:1630,j],color='Red',marker='.')  plt.subplot(1,2,2)  plt.title('Osmancik')  plt.scatter(df.iloc[1630:,i],df.iloc[1630:,j],color='tab:blue',marker='.')  plt.show()  ##Prediction Class  plt.figure(figsize=(5,5)) #No. of rows for each type of rice  plt.bar(bary,barx,0.4,color=['r','tab:blue'])  plt.title('CLASS',color='r')  plt.xlabel('Rice types',weight='bold')  plt.ylabel('No. of Rows',weight='bold')  plt.text(0,barx[0]+50,barx[0],size='x-large',ha='center',weight='bold')  plt.text(1,barx[1]+50,barx[1],size='x-large',ha='center',weight='bold')  plt.ylim([0,3000])  plt.show()  #Data Processing and Analyisis  x=df.loc[:,df.columns!='CLASS'] #all columns except CLASS column  y=df['CLASS'] #CLASS cloumn  y.head() #X is features of Rice, Y is Rice types to predict  ##Label Encoding  LE = LabelEncoder()  y= LE.fit\_transform(y) #Cammeo=0, Osmancik=1  y  #Random Forest Classifier Training and Testing  ##Best model Function  def bestmodel(x\_train,y\_train,x\_test,y\_test):# Gives me Best Random forest model with optimal Hyper parameters  def ele(v):  return v[2]  f2x=[]  for i in range(1,11):  for j in range(10,0,-1):  model=RandomForestClassifier(n\_estimators=100, bootstrap=True, max\_depth=i, min\_samples\_leaf=j, random\_state=12)  rf\_train=model.fit(x\_train,y\_train)  f2=f1\_score(y\_test,rf\_train.predict(x\_test))  lgls=log\_loss(y\_test, rf\_train.predict\_proba(x\_test))  acc=accuracy\_score(y\_test, rf\_train.predict(x\_test))  rac=roc\_auc\_score(y\_test, rf\_train.predict(x\_test))  f2x.append([i,j,f2,lgls,rac,acc])  f2x.sort(key=ele,reverse=True) #sorts according to fscore in decending order  for i in f2x:  if(f2x[0][2]==i[2]):  print(i)  else:  break  for i in np.arange(0.1,0.6,0.1):  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=i, shuffle=True,random\_state=12)  print("Test\_size= ",i)  bestmodel(x\_train,y\_train,x\_test,y\_test)  ####output format= [max\_depth ,min\_sample\_leaf, fscore, log\_loss,roc\_auc\_score,accuracy\_score]  #Random Forest Model  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, shuffle=True,random\_state=12)  model=RandomForestClassifier(n\_estimators=100, bootstrap=True, max\_depth=6, min\_samples\_leaf=10, random\_state=12)  rf\_train=model.fit(x\_train,y\_train)  rf\_train  print("Training Accuracy Score : ",accuracy\_score(y\_train,rf\_train.predict(x\_train))\*100,"%")  #Performance Metrics  confusion\_matrix(y\_test,rf\_train.predict(x\_test))  fig, ax = plt.subplots(figsize=(8, 8))  plt.rcParams.update({'font.size': 16})  plot\_confusion\_matrix(rf\_train,x\_test,y\_test,cmap='plasma',ax=ax)  plt.title('Confusion matrix')  plt.savefig('confusion\_matrix.jpg')  plt.show()  print("F1-score : ",f1\_score(y\_test,rf\_train.predict(x\_test)))  ######F1\_score balances the percision and recall from confusion matrix  print("precision, recall, fscore, support :",precision\_recall\_fscore\_support(y\_test,rf\_train.predict(x\_test),average ='binary'))  print("Log Loss :",log\_loss(y\_test, rf\_train.predict\_proba(x\_test)))  print("Testing Accuracy Score : ",accuracy\_score(y\_test, rf\_train.predict(x\_test))\*100,"%") #Accuracy of prediction in %  roc\_auc\_score(y\_test, rf\_train.predict(x\_test))  #Observations  ##Plotting no. of decision trees vs LogLoss  preds = np.stack([t.predict\_proba(x\_test) for t in rf\_train.estimators\_])  lgls=[log\_loss(y\_test,np.mean(preds[:i+1], axis=0)) for i in range(100)]  plt.figure(figsize=(10,6))  plt.plot(lgls,color='b',linewidth=3)  plt.xlabel('Number of trees')  plt.ylabel('log loss')  plt.title('Log Loss')  plt.show()  ##Plotting ROC curve for True Positives VS False Positives  fpr, tpr, thresholds = roc\_curve(y\_test, rf\_train.predict(x\_test))  roc\_auc = auc(fpr, tpr)  display =RocCurveDisplay(fpr=fpr, tpr=tpr, roc\_auc=roc\_auc,estimator\_name='Random Forest')  display.plot(color='r')  plt.plot(list(np.arange(0,2,0.1)),list(np.arange(0,2,0.1)),color='k')  plt.title('ROC curve')  plt.xlim([0,1])  plt.ylim([0,1])  plt.show()  #Sample Decision tree in Random forest model  ##visualise in one of the decision tree in random forest with best Hyper parameters  dot\_data = export\_graphviz(rf\_train.estimators\_[0],  feature\_names=x.columns.values,  class\_names=['Cammeo','Osmancik'],  filled=True, impurity=True,  rounded=True)  graph = graphviz.Source(dot\_data, format='png')  graph.render('figure\_name')  graph |