A logo for university of canberra

Description automatically generated

***Faculty of Science and Technology***

**Assignment Coversheet**

|  |  |
| --- | --- |
| **Student ID number &**  **Student Name** | U3059571  Yanlong Su |
| **Unit name** | Software Technology 1 |
| **Unit number** | 4483 |
| **Unit Tutor** | Mr. Pranav Gupta |
| **Assignment name** | ST1 Capstone Project – Semester 1 2023 |
| **Due date** | 29 Oct 2023 |
| **Date submitted** | 29 Oct 2023 |

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source, and other bibliographic details.

**Signature of student: Yanlong Su Date: 29/10/2023**

Table of Contents

[Introduction 3](#_Toc149463714)

[Rice MSC data set: 3](#_Toc149463715)

[Questions: 3](#_Toc149463716)

[Methodology 3](#_Toc149463717)

[Exploratory Data Analysis: 3](#_Toc149463718)

[Predictive Data Analytics: 12](#_Toc149463719)

[Model Preparation and Development: 14](#_Toc149463720)

[Implementation and deployment 22](#_Toc149463721)

[GitHub Link: 25](#_Toc149463722)

[Conlcusions 26](#_Toc149463723)

[References 27](#_Toc149463724)

[Appendix 1: Log Book 28](#_Toc149463725)

# Introduction

This report is about the capstone project for ST1 unit. It contains the codes that can be run on Python and Google Colab as well as the output from running the code. The structure of the report is organised as follows: firstly, I will introduce the purpose of this report and the pre-process to the provided date set. Secondly, I will perform the exploratory data analysis and predictive data analysis. Then, the project will be set up for deployment of web service via Streamlit. Link to the GitHub repository is also a crucial part of the phase stage, it will be shared in the implement section. The conclusion is presented at last.

## Rice MSC data set:

The project uses the data set downloaded from the Kaggle website [1], [2], [3], [4], [5]. It is called Rice MSC data set. The data set contains a CSV file with more than 8 million data cells. It contains 5 classes of rice varieties, they are Arborio, Basmati, Jasmine, Ipsala, and Karacadag. A total of 75,000 pieces of rice grain were obtained from same brand, including 15,000 pieces of each variety of rice. A total of 106 features are identified, including 12 morphological features, 4 shape features and 90 colour features. Roundness, compactness, shape factor 3, aspect ratio and eccentricity are 5 most important features among all 106 features. The purpose of this data set is to develop a model that trains the AI to categorise and differentiate between 5 rice varieties based on their image features.

However, 90 colour features in the data set are seemed to be a noise factor. Since the model should be emphasising on shape attributes rather than colour attributes. Thus, 90 colour features will not be used for model development and training in this report.

## Questions:

1. How many rows and columns are used for EDA?
2. what is the sample size for each attribute?
3. How many outliers are in the sample?
4. How many samples left after removal of outlier for each class?
5. What is the correlation between attributes?

# Methodology

## Exploratory Data Analysis:

The first set for the EDA is to set up the working environment. Google Colab was chosen as it can run the Jupyter notebook and is compatible with Python which is the main language we are using for the coding.

# Import Required Python Packages and read data  
import os  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import missingno as msno # To visualize missing value  
import plotly.graph\_objects as go # To Generate Graphs  
import plotly.express as px # To Generate box plot for statistical representation  
%matplotlib inline  
import warnings  
warnings.filterwarnings('ignore')

# Read dataset

df = pd.read\_csv(r"https://raw.githubusercontent.com/jacklong233/ST1/main/Rice\_MSC\_Dataset\_Trimmed.csv")

# Checking description: first 5 rows

df.head()

A screenshot of a computer

Description automatically generated

Continue:

A screenshot of a computer

Description automatically generated

# Checking description: last 5 rows

df.tail()

A screenshot of a computer

Description automatically generated

Continue:

A screenshot of a computer

Description automatically generated

# Rows and columns-data shape (attributes & samples)

df.shape

(75000, 17)

Answer for question 1: there are 75000 rows and 17 columns used for EDA.

# Name of the attributes

df.columns

Index(['AREA', 'PERIMETER', 'MAJOR\_AXIS', 'MINOR\_AXIS', 'ECCENTRICITY',

'EQDIASQ', 'SOLIDITY', 'CONVEX\_AREA', 'EXTENT', 'ASPECT\_RATIO',

'ROUNDNESS', 'COMPACTNESS', 'SHAPEFACTOR\_1', 'SHAPEFACTOR\_2',

'SHAPEFACTOR\_3', 'SHAPEFACTOR\_4', 'CLASS'],

dtype='object')

# unique values for each attribute

df.nunique()

AREA 10793

PERIMETER 57459

MAJOR\_AXIS 71629

MINOR\_AXIS 67873

ECCENTRICITY 3026

EQDIASQ 10793

SOLIDITY 588

CONVEX\_AREA 11069

EXTENT 5464

ASPECT\_RATIO 26437

ROUNDNESS 5247

COMPACTNESS 4291

SHAPEFACTOR\_1 247

SHAPEFACTOR\_2 84

SHAPEFACTOR\_3 5411

SHAPEFACTOR\_4 544

CLASS 5

dtype: int64

Answer to question 2: see above sample numbers for each unique attributes

# Complete info about data frame

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 75000 entries, 0 to 74999

Data columns (total 17 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 AREA 75000 non-null int64

1 PERIMETER 75000 non-null float64

2 MAJOR\_AXIS 75000 non-null float64

3 MINOR\_AXIS 75000 non-null float64

4 ECCENTRICITY 75000 non-null float64

5 EQDIASQ 75000 non-null float64

6 SOLIDITY 75000 non-null float64

7 CONVEX\_AREA 75000 non-null int64

8 EXTENT 75000 non-null float64

9 ASPECT\_RATIO 75000 non-null float64

10 ROUNDNESS 75000 non-null float64

11 COMPACTNESS 75000 non-null float64

12 SHAPEFACTOR\_1 75000 non-null float64

13 SHAPEFACTOR\_2 75000 non-null float64

14 SHAPEFACTOR\_3 75000 non-null float64

15 SHAPEFACTOR\_4 75000 non-null float64

16 CLASS 75000 non-null object

dtypes: float64(14), int64(2), object(1)

memory usage: 9.7+ MB

# Visualising data  distribution in detail

fig = plt.figure(figsize =(30,30))  
ax=fig.gca()  
df.hist(ax=ax,bins =30)  
plt.show()

A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

# Detecting outliers

df.plot(kind='box', subplots=True,  
 layout=(20,8),sharex=False,sharey=False, figsize=(30, 150), color='deeppink');

A screenshot of a computer screen

Description automatically generated

# Identify the outliers  
continous\_features = ['AREA', 'PERIMETER', 'MAJOR\_AXIS', 'MINOR\_AXIS', 'ECCENTRICITY', 'EQDIASQ', 'SOLIDITY', 'CONVEX\_AREA', 'EXTENT', 'ASPECT\_RATIO', 'ROUNDNESS', 'COMPACTNESS', 'SHAPEFACTOR\_1','SHAPEFACTOR\_2','SHAPEFACTOR\_3','SHAPEFACTOR\_4']  
def outliers(df\_out, drop = False):  
 for each\_feature in df\_out.columns:  
 feature\_data = df\_out[each\_feature]  
 Q1 = np.percentile(feature\_data, 25.) # 25th percentile of the data of the given feature  
 Q3 = np.percentile(feature\_data, 75.) # 75th percentile of the data of the given feature  
 IQR = Q3-Q1 #Interquartile Range  
 outlier\_step = IQR \* 1.5   
 outliers = feature\_data[~((feature\_data >= Q1 - outlier\_step) & (feature\_data <= Q3 + outlier\_step))].index.tolist()  
 if not drop:  
 print('For the feature {}, No of Outliers is {}'.format(each\_feature, len(outliers)))  
 if drop:  
 df.drop(outliers, inplace = True, errors = 'ignore')  
 print('Outliers from {} feature removed'.format(each\_feature))  
  
outliers(df[continous\_features])

For the feature AREA, No of Outliers is 11986

For the feature PERIMETER, No of Outliers is 0

For the feature MAJOR\_AXIS, No of Outliers is 0

For the feature MINOR\_AXIS, No of Outliers is 0

For the feature ECCENTRICITY, No of Outliers is 140

For the feature EQDIASQ, No of Outliers is 9165

For the feature SOLIDITY, No of Outliers is 722

For the feature CONVEX\_AREA, No of Outliers is 11569

For the feature EXTENT, No of Outliers is 49

For the feature ASPECT\_RATIO, No of Outliers is 113

For the feature ROUNDNESS, No of Outliers is 0

For the feature COMPACTNESS, No of Outliers is 0

For the feature SHAPEFACTOR\_1, No of Outliers is 0

For the feature SHAPEFACTOR\_2, No of Outliers is 0

For the feature SHAPEFACTOR\_3, No of Outliers is 0

For the feature SHAPEFACTOR\_4, No of Outliers is 1716

Answer to question 3: see above for the outlier amount for each attribute

# Remove outliers:

outliers(df[continous\_features], drop = True)

Outliers from AREA feature removed

Outliers from PERIMETER feature removed

Outliers from MAJOR\_AXIS feature removed

Outliers from MINOR\_AXIS feature removed

Outliers from ECCENTRICITY feature removed

Outliers from EQDIASQ feature removed

Outliers from SOLIDITY feature removed

Outliers from CONVEX\_AREA feature removed

Outliers from EXTENT feature removed

Outliers from ASPECT\_RATIO feature removed

Outliers from ROUNDNESS feature removed

Outliers from COMPACTNESS feature removed

Outliers from SHAPEFACTOR\_1 feature removed

Outliers from SHAPEFACTOR\_2 feature removed

Outliers from SHAPEFACTOR\_3 feature removed

Outliers from SHAPEFACTOR\_4 feature removed

# Rows and columns-data shape after removal of outliers

df.shape

(60827, 17)

Answer to question 4: only 60,827 samples left.

# Check if outliers got removed

df.plot(kind='box', subplots=True,  
 layout=(20,8),sharex=False,sharey=False, figsize=(30, 150), color='deeppink');

A screenshot of a computer

Description automatically generated

# Checking target value distribution

print(df.CLASS.value\_counts())  
fig, ax = plt.subplots(figsize=(5,4))  
name = ["Aborio", "Basmati", "Jasmine", "Karacadag", "Ipsala"]  
ax = df.CLASS.value\_counts().plot(kind='bar')  
ax.set\_title("Rice MSC", fontsize = 13, weight = 'bold')  
ax.set\_xticklabels (name, rotation = 0)  
  
# To calculate the percentage  
totals = []  
for i in ax.patches:  
 totals.append(i.get\_height())  
total = sum(totals)  
for i in ax.patches:  
 ax.text(i.get\_x()+.09, i.get\_height()-50,  
 str(round((i.get\_height()/total)\*100, 2))+'%', fontsize=14,  
 color='white', weight = 'bold')  
plt.tight\_layout()

A screenshot of a graph

Description automatically generated

# Check correlation between variables

sns.set(style="white")  
plt.rcParams['figure.figsize'] = (20, 10)  
sns.heatmap(df.iloc[:, :-1].corr(), annot = True, linewidths=1, cmap="Blues")  
plt.title('Correlation Between Variables', fontsize = 30)  
plt.show()

A screenshot of a computer

Description automatically generated

Answer to question 5: the correlation is shown in above picture

# Obtain full profiler report

!pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip

#restart kernel

#re-run import libraries and data

import pandas as pd  
import numpy as np  
from pandas\_profiling import ProfileReport

profile = ProfileReport(df,title="Rice MSC",  
 html={'style':{'full\_width':True}})

profile.to\_notebook\_iframe()

A screenshot of a computer

Description automatically generated

## Predictive Data Analytics:

#pre-processing

from sklearn.exceptions import DataDimensionalityWarning  
#encode object columns to integers  
from sklearn import preprocessing  
from sklearn.preprocessing import OrdinalEncoder  
  
for col in df:  
 if df[col].dtype =='object':  
 df[col]=OrdinalEncoder().fit\_transform(df[col].values.reshape(-1,1))  
df

A screenshot of a computer

Description automatically generated

There are too many rows and columns which can’t be fully written in the report. Please check python file for more details.

class\_label =df['CLASS']  
df = df.drop(['CLASS'], axis =1)  
df = (df-df.min())/(df.max()-df.min())  
df['CLASS']=class\_label  
df

A screenshot of a computer

Description automatically generated

There are too many rows and columns which can’t be fully written in the report. Please check python file for more details.

#Data Preprocessing

import sklearn  
from sklearn import linear\_model, preprocessing  
from sklearn.model\_selection import KFold  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import GridSearchCV  
from sklearn.metrics import classification\_report  
from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay  
from sklearn.metrics import accuracy\_score  
from sklearn.pipeline import Pipeline  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
from sklearn.ensemble import GradientBoostingClassifier  
from sklearn.ensemble import RandomForestClassifier

rice\_data = df.copy()  
le = preprocessing.LabelEncoder()  
area = le.fit\_transform(list(rice\_data["AREA"]))  
perimete = le.fit\_transform(list(rice\_data["PERIMETER"]))  
major\_axis = le.fit\_transform(list(rice\_data["MAJOR\_AXIS"]))   
minor\_axis = le.fit\_transform(list(rice\_data["MINOR\_AXIS"]))   
eccentricity = le.fit\_transform(list(rice\_data["ECCENTRICITY"]))   
eqdiasq = le.fit\_transform(list(rice\_data["EQDIASQ"]))   
solidity = le.fit\_transform(list(rice\_data["SOLIDITY"]))   
convex\_area = le.fit\_transform(list(rice\_data["CONVEX\_AREA"]))   
extent = le.fit\_transform(list(rice\_data["EXTENT"]))   
aspect\_ratio = le.fit\_transform(list(rice\_data["ASPECT\_RATIO"]))   
roundness = le.fit\_transform(list(rice\_data["ROUNDNESS"]))   
compactness = le.fit\_transform(list(rice\_data["COMPACTNESS"]))   
shapefactor\_1 = le.fit\_transform(list(rice\_data["SHAPEFACTOR\_1"]))   
shapefactor\_2 = le.fit\_transform(list(rice\_data["SHAPEFACTOR\_2"]))   
shapefactor\_3 = le.fit\_transform(list(rice\_data["SHAPEFACTOR\_3"]))   
shapefactor\_4 = le.fit\_transform(list(rice\_data["SHAPEFACTOR\_4"]))   
Class = le.fit\_transform(list(rice\_data["CLASS"]))

## Model Preparation and Development:

x = list(zip(area, perimete, major\_axis, minor\_axis, eccentricity, eqdiasq, solidity, convex\_area, extent, aspect\_ratio, roundness, compactness, shapefactor\_1, shapefactor\_2, shapefactor\_3, shapefactor\_4))  
y = list(Class)  
# Test options and evaluation metric  
num\_folds = 5  
seed = 7  
scoring = 'accuracy'  
  
# Model Test/Train  
# Splitting what we are trying to predict into 4 different arrays -  
# X train is a section of the x array(attributes) and vise versa for Y(features)  
# The test data will test the accuracy of the model created  
x\_train, x\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size = 0.20, random\_state=seed)  
# Splitting 20% of our data into test samples. If we train the model with higher data it already has seen that information and knows

# Size of train and test subsets after splitting  
np.shape(x\_train), np.shape(x\_test)

((48661, 16), (12166, 16))

# Predictive analytics model development by comparing different Scikit-learn classification algorithms

models = []  
models.append(('DT', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', SVC()))  
models.append(('GBM', GradientBoostingClassifier()))  
models.append(('RF', RandomForestClassifier()))

# Evaluate each model in turn  
results = []  
names = []  
  
print("Performance on Training set")  
  
for name, model in models:  
 kfold = KFold(n\_splits=num\_folds,shuffle=True,random\_state=seed)  
 cv\_results = cross\_val\_score(model, x\_train, y\_train, cv=kfold, scoring='accuracy')  
 results.append(cv\_results)  
 names.append(name)  
 msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())  
 msg += '\n'  
 print(msg)

Performance on Training set

DT: 0.964797 (0.002743)

NB: 0.965537 (0.002400)

SVM: 0.970161 (0.001902)

GBM: 0.976552 (0.003089)

RF: 0.976203 (0.002804)

# Compare Algorithms' Performance

fig = plt.figure()  
fig.suptitle('Algorithm Comparison')  
ax = fig.add\_subplot(111)  
plt.boxplot(results)  
ax.set\_xticklabels(names)  
plt.show()

A screenshot of a computer

Description automatically generated

# Model Evaluation of best performing model, by testing with Independent/external test data set.

# Make predictions on validation/test dataset

models.append(('DT', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', SVC()))  
models.append(('GBM', GradientBoostingClassifier()))  
models.append(('RF', RandomForestClassifier()))  
dt = DecisionTreeClassifier()  
nb = GaussianNB()  
gb = GradientBoostingClassifier()  
rf = RandomForestClassifier()  
  
best\_model = rf  
best\_model.fit(x\_train, y\_train)  
y\_pred = best\_model.predict(x\_test)  
print("Best Model Accuracy Score on Test Set:", accuracy\_score(y\_test, y\_pred))

Best Model Accuracy Score on Test Set: 0.9750123294427092

# Model Performance Evaluation Metric 1 - Classification Report

print(classification\_report(y\_test, y\_pred))

A screenshot of a computer screen

Description automatically generated

# Model Performance Evaluation Metric 2

# Confusion matrix

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay  
cm = confusion\_matrix(y\_test, y\_pred)  
disp = ConfusionMatrixDisplay(confusion\_matrix=cm)  
disp.plot(cmap='Blues')  
plt.show()

A screenshot of a computer

Description automatically generated

# Model Evaluation Metric 3- ROC-AUC curve [6]

from sklearn.metrics import roc\_auc\_score  
from sklearn.metrics import roc\_curve  
best\_model.fit(x\_train, y\_train)  
n\_classes = len(set(y\_train))  
fpr = dict()  
tpr = dict()  
roc\_auc = dict()  
for i in range(n\_classes):  
 y\_test\_bin = pd.Series(y\_test).map(lambda x: 1 if x == i else 0)  
 y\_score\_bin = best\_model.predict\_proba(x\_test)[:, i]   
 fpr[i], tpr[i], \_ = roc\_curve(y\_test\_bin, y\_score\_bin)  
 roc\_auc[i] = roc\_auc\_score(y\_test\_bin, y\_score\_bin)  
plt.figure()  
colors = ['blue', 'red', 'green', 'yellow', 'purple', 'cyan']   
for i, color in zip(range(n\_classes), colors):  
 plt.plot(fpr[i], tpr[i], color=color, label='ROC curve of class {0} (area = {1:0.2f})'.format(i, roc\_auc[i]))  
  
plt.plot([0, 1], [0, 1], 'k--')  
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) for Multi-class')  
plt.legend(loc="lower right")  
plt.savefig('LOC\_ROC')  
plt.show()

A screen shot of a graph

Description automatically generated

# Model Evaluation Metric 4-prediction report

for x in range(len(y\_pred)):  
 print("Predicted: ", y\_pred[x], "Actual: ", y\_test[x], "Data: ", x\_test[x],)

There are too many results that can’t fit in the report entirely. So only 3 pages of results are shown:

dicted: 0 Actual: 0 Data: (2945, 25072, 27839, 39087, 1691, 2945, 129, 3113, 4330, 5626, 3579, 2688, 47, 38, 3033, 146)

Predicted: 4 Actual: 4 Data: (1713, 7582, 8907, 44632, 782, 1713, 234, 1764, 4544, 1681, 4486, 3563, 41, 58, 4341, 193)

Predicted: 4 Actual: 4 Data: (1966, 8504, 12530, 43104, 1003, 1966, 340, 1947, 3947, 2400, 4681, 3421, 41, 53, 4118, 277)

Predicted: 0 Actual: 0 Data: (3400, 26079, 25204, 50486, 1414, 3400, 242, 3483, 4664, 4034, 3856, 2994, 35, 39, 3473, 150)

Predicted: 4 Actual: 4 Data: (2142, 7473, 5423, 52463, 100, 2142, 325, 2136, 4003, 133, 5072, 4049, 26, 60, 5117, 263)

Predicted: 0 Actual: 0 Data: (2731, 21504, 19064, 45277, 1397, 2731, 267, 2779, 4490, 3953, 3898, 3006, 42, 43, 3490, 138)

Predicted: 0 Actual: 0 Data: (2175, 20887, 17782, 34980, 1536, 2175, 160, 2293, 3115, 4676, 3371, 2832, 53, 46, 3238, 60)

Predicted: 0 Actual: 0 Data: (3116, 26056, 29338, 41276, 1684, 3116, 107, 3309, 3025, 5582, 3578, 2704, 45, 37, 3055, 170)

Predicted: 0 Actual: 0 Data: (3619, 26986, 31434, 48417, 1596, 3619, 256, 3695, 4491, 5029, 3877, 2833, 36, 35, 3238, 244)

Predicted: 3 Actual: 3 Data: (241, 11333, 22659, 13808, 2412, 241, 144, 308, 2288, 10370, 2133, 1552, 133, 41, 1590, 100)

Predicted: 4 Actual: 4 Data: (1558, 5821, 11141, 34726, 1123, 1558, 337, 1536, 3588, 2837, 4567, 3312, 49, 55, 3949, 263)

Predicted: 1 Actual: 1 Data: (3585, 41568, 55743, 18732, 2774, 3585, 176, 3732, 752, 20261, 863, 499, 120, 7, 461, 143)

Predicted: 3 Actual: 3 Data: (4926, 42043, 52768, 24866, 2592, 4926, 219, 5061, 4930, 14598, 1721, 1105, 74, 10, 1083, 97)

Predicted: 3 Actual: 3 Data: (197, 12901, 32274, 5681, 2551, 197, 187, 239, 1352, 13499, 1932, 1212, 150, 36, 1201, 84)

Predicted: 3 Actual: 3 Data: (504, 14733, 32015, 13119, 2486, 504, 221, 535, 1536, 11840, 2138, 1384, 134, 36, 1395, 108)

Predicted: 3 Actual: 3 Data: (387, 18178, 36977, 4724, 2599, 387, 235, 407, 1371, 14772, 1690, 1096, 151, 32, 1073, 138)

Predicted: 3 Actual: 3 Data: (649, 11792, 22701, 20302, 2311, 649, 215, 687, 1850, 9292, 2600, 1777, 113, 41, 1858, 152)

Predicted: 0 Actual: 0 Data: (2479, 20884, 18068, 40561, 1447, 2479, 115, 2642, 4154, 4199, 3705, 2960, 46, 44, 3423, 149)

Predicted: 3 Actual: 3 Data: (61, 9891, 22150, 10732, 2437, 61, 125, 132, 3645, 10822, 2047, 1470, 141, 42, 1493, 12)

Predicted: 0 Actual: 0 Data: (2460, 24850, 32312, 27097, 1999, 2460, 263, 2506, 2755, 7854, 3121, 2294, 64, 35, 2493, 188)

Predicted: 3 Actual: 3 Data: (152, 9953, 25364, 8771, 2479, 152, 205, 183, 3744, 11685, 2158, 1409, 142, 39, 1424, 139)

Predicted: 1 Actual: 1 Data: (1696, 33886, 50584, 1145, 2834, 1696, 175, 1788, 175, 23512, 364, 211, 171, 11, 190, 77)

Predicted: 4 Actual: 4 Data: (1467, 5927, 9200, 36590, 995, 1467, 299, 1469, 4065, 2374, 4424, 3395, 49, 57, 4077, 194)

Predicted: 0 Actual: 0 Data: (2961, 25751, 30690, 35716, 1787, 2961, 216, 3056, 2849, 6303, 3480, 2583, 50, 36, 2886, 183)

Predicted: 1 Actual: 1 Data: (2934, 35936, 49889, 18417, 2732, 2934, 229, 3018, 690, 18222, 1009, 653, 122, 12, 612, 97)

Predicted: 1 Actual: 1 Data: (2818, 39296, 54112, 10975, 2799, 2818, 105, 3001, 563, 21714, 595, 381, 138, 9, 348, 103)

Predicted: 1 Actual: 1 Data: (1709, 31893, 47300, 1980, 2805, 1709, 207, 1779, 2413, 22064, 641, 381, 160, 13, 348, 218)

Predicted: 4 Actual: 4 Data: (1727, 6740, 7989, 45150, 725, 1727, 310, 1726, 3926, 1516, 4638, 3633, 40, 58, 4452, 261)

Predicted: 4 Actual: 4 Data: (1374, 4558, 9244, 33405, 1084, 1374, 320, 1361, 4524, 2693, 4544, 3336, 51, 57, 3986, 235)

Predicted: 1 Actual: 1 Data: (2797, 37967, 52530, 12864, 2781, 2797, 104, 2980, 465, 20653, 705, 466, 134, 10, 428, 129)

Predicted: 1 Actual: 1 Data: (2228, 33522, 47635, 8737, 2767, 2228, 127, 2373, 1449, 19887, 794, 544, 140, 13, 504, 210)

Predicted: 1 Actual: 1 Data: (1384, 32194, 47478, 435, 2836, 1384, 232, 1429, 950, 23577, 347, 256, 176, 12, 232, 305)

Predicted: 4 Actual: 4 Data: (1130, 2882, 8707, 29009, 1218, 1130, 345, 1098, 3684, 3203, 4535, 3222, 58, 57, 3812, 253)

Predicted: 1 Actual: 1 Data: (3390, 40784, 54663, 18749, 2765, 3390, 78, 3619, 718, 19766, 820, 516, 122, 8, 478, 64)

Predicted: 0 Actual: 0 Data: (1958, 17442, 24128, 25216, 1971, 1958, 298, 1969, 4458, 7658, 3555, 2372, 69, 38, 2597, 297)

Predicted: 4 Actual: 4 Data: (1292, 4319, 4650, 39259, 733, 1292, 249, 1324, 3772, 1537, 4472, 3591, 47, 62, 4386, 171)

Predicted: 1 Actual: 1 Data: (1839, 34139, 46733, 3405, 2790, 1839, 129, 1969, 367, 21157, 439, 457, 153, 13, 420, 244)

Predicted: 1 Actual: 1 Data: (2682, 44925, 57594, 1853, 2867, 2682, 142, 2830, 126, 24479, 9, 79, 159, 4, 71, 261)

Predicted: 1 Actual: 1 Data: (3940, 45490, 58221, 16150, 2820, 3940, 180, 4090, 750, 22877, 489, 314, 124, 3, 285, 231)

Predicted: 1 Actual: 1 Data: (2380, 34883, 49976, 10175, 2776, 2380, 158, 2506, 4226, 20373, 743, 470, 141, 12, 433, 61)

Predicted: 4 Actual: 4 Data: (1701, 6808, 8518, 44057, 780, 1701, 263, 1732, 3705, 1677, 4590, 3583, 41, 58, 4373, 240)

Predicted: 4 Actual: 4 Data: (1736, 7090, 10246, 41365, 926, 1736, 339, 1715, 3779, 2136, 4592, 3483, 43, 56, 4215, 273)

Predicted: 4 Actual: 4 Data: (1846, 7445, 7781, 48891, 580, 1846, 285, 1864, 4561, 1117, 4684, 3717, 36, 59, 4587, 209)

Predicted: 4 Actual: 4 Data: (1108, 2272, 946, 42312, 367, 1108, 252, 1134, 4244, 605, 4646, 3856, 43, 67, 4814, 201)

Predicted: 1 Actual: 1 Data: (3422, 40321, 54677, 18629, 2766, 3422, 266, 3485, 4824, 19816, 884, 525, 121, 8, 486, 113)

Predicted: 3 Actual: 3 Data: (411, 21758, 38640, 2704, 2651, 411, 164, 472, 1145, 15953, 1378, 935, 159, 30, 901, 117)

Predicted: 0 Actual: 0 Data: (2641, 21368, 20621, 41024, 1522, 2641, 232, 2715, 4352, 4597, 3818, 2863, 46, 43, 3283, 103)

Predicted: 4 Actual: 4 Data: (863, 1154, 2089, 30805, 857, 863, 278, 868, 4064, 1913, 4605, 3529, 54, 64, 4287, 251)

Predicted: 4 Actual: 4 Data: (1460, 5424, 3991, 45803, 510, 1460, 145, 1566, 4170, 938, 4500, 3745, 41, 63, 4633, 162)

# Implementation and deployment

For this project, I’ve chosen Streamlit for the deployment of the app.

The code shown below will be implemented via Streamlit as a web based tool [6].

import streamlit as st  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.metrics import accuracy\_score  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report, confusion\_matrix, ConfusionMatrixDisplay, roc\_curve  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier  
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder  
  
URL = 'https://raw.githubusercontent.com/jacklong233/ST1/main/Rice\_MSC\_Dataset\_Trimmed.csv'

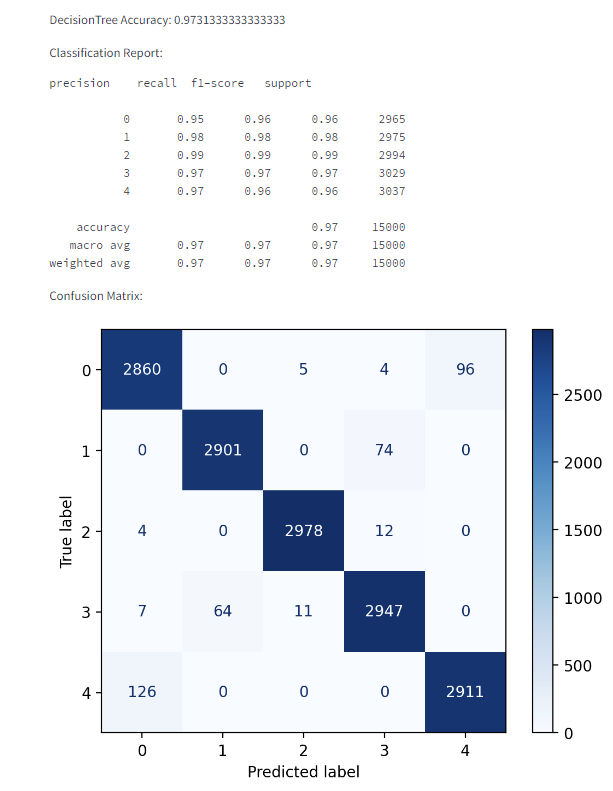
@st.cache\_resource

def load\_data():

df = pd.read\_csv(URL)  
  
 for col in df:  
 if df[col].dtype == 'object':  
 df[col] = OrdinalEncoder().fit\_transform(df[col].values.reshape(-1, 1))  
  
 df\_normalized = (df - df.min()) / (df.max() - df.min())  
  
 le = LabelEncoder()  
 labels = le.fit\_transform(df\_normalized['CLASS'])  
 df\_normalized.drop('CLASS', axis=1, inplace=True)  
 return df\_normalized, labels  
  
def main():  
 st.title("Rice Data Classifier")  
  
 data, target = load\_data()  
  
 st.sidebar.header("Model Selection")  
 model\_choice = st.sidebar.selectbox(  
 "Choose the Classifier",  
 ("DecisionTree", "NaiveBayes", "SVM", "GradientBoosting", "RandomForest"))  
  
 if st.sidebar.button("Train Model"):  
 x\_train, x\_test, y\_train, y\_test = train\_test\_split(data, target, test\_size=0.20, random\_state=7)  
  
 if model\_choice == "DecisionTree":  
 model = DecisionTreeClassifier()  
 elif model\_choice == "NaiveBayes":  
 model = GaussianNB()  
 elif model\_choice == "SVM":  
 model = SVC(probability=True)  
 elif model\_choice == "GradientBoosting":  
 model = GradientBoostingClassifier()  
 else:  
 model = RandomForestClassifier()  
  
 model.fit(x\_train, y\_train)  
 y\_pred = model.predict(x\_test)  
  
 st.write(f"{model\_choice} Accuracy: {accuracy\_score(y\_test, y\_pred)}")  
 st.write("Classification Report:")  
 st.text(classification\_report(y\_test, y\_pred))

st.write("Confusion Matrix:")  
 cm = confusion\_matrix(y\_test, y\_pred)  
 fig, ax = plt.subplots()  
 ConfusionMatrixDisplay(confusion\_matrix=cm).plot(ax=ax, cmap='Blues')  
 st.pyplot(fig)  
  
 y\_prob = model.predict\_proba(x\_test)[:, 0]  
 fpr, tpr, \_ = roc\_curve(y\_test, y\_prob, pos\_label=0)  
 fig, ax = plt.subplots()  
 ax.plot(fpr, tpr)  
 ax.plot([0, 1], [0, 1], linestyle='--')  
 ax.set\_title('ROC Curve')  
 ax.set\_xlabel('False Positive Rate')  
 ax.set\_ylabel('True Positive Rate')  
 st.pyplot(fig)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

The test run is showing below:

when we choose the DT model and click ‘Train Model’, the classification report, confusion matrix, and ROC curve will show up:A graph of a curve

Description automatically generated

On the left side of screen, there are 5 different models for training purpose.

A screenshot of a computer

Description automatically generated

Every trainning model displays the accuracy of the current model, a classification report, the confusion matrix, and ROC curve.

## GitHub Link:

The GitHub repositary link is attached below:

[**https://github.com/jacklong233/ST1**](https://github.com/jacklong233/ST1)

# Conlcusions

This report is aiming to develop and deploy a Python based project to catergoise and differentiate 5 different types of rice using the Rice MSC dataset from Kaggle. To enhace the accuracy, I have decided to use 16 attributes instead of 106. Hence 90 colour features are ditched during the pre-processing data phase. With the implementation of Exploratory Data Analysis (EDA) and Predictive Data Analysis (PDA), our model has achieved an accuracy of around 98% in predicting the rice varieties. In order to enahnce the userbility and make the model more accessible, Streamlit is used to deploy the App on the cloud.

However, according to the experiment, the Ipsala type of rice has lost significant amount of samples during the outlier removal phase. There is no evidence on why this happened. Due to the constraint and limitation of this study, we can not identify the cause to the problem.

Implementation of our model can improve the accuracy and efficiency of the rice extraction process significantly. It ensures a higher output and much lower resource wastage, thus give manufactures an advantage in both economic and environmental scale.

# References

[1] https://www.kaggle.com/datasets/muratkokludataset/rice-msc-dataset

[2] K. M., C. I., and T. Y.S., “Classification of rice varieties with deep learning methods,” 2021. <https://doi.org/10.1016/j.compag.2021.106285>

[3] C. I. and K. M., “Determination of Effective and Specific Physical Features of Rice Varieties by Computer Vision In Exterior Quality Inspection,” 2021. <https://doi.org/10.15316/SJAFS.2021.252>

[4] C. I. and K. M., “Identification of Rice Varieties Using Machine Learning Algorithms,” 2022. <https://doi.org/10.15832/ankutbd.862482>

[5] C. I and K. M., “Classification of Rice Varieties Using Artificial Intelligence Methods,” 2019. <https://doi.org/10.18201/ijisae.2019355381>

[6] OpenAI, “ChatGPT,” *chat.openai.com*, Oct. 16, 2023. https://chat.openai.com/

# Appendix 1: Log Book

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Week | Planned  Activities | Tasks  Completed | Problems  Faced | Further Comments |
| Week 7 | Challenge question 7 | Challenge question 7 | N/A |  |
| Week 8 | Challenge question 9, 10 | Challenge question 9, 10 | Questions of problem 9b and 10 b are unclear to me, do we store the book name and display it, or shall we input the book name and then let the app display it? |  |
| Week 9 | Read materials, familiar with requirement of assignment, challenge 11 | Read materials, familiar with requirement of assignment, challenge 11 | N/A |  |
| Week 10 | EDA, learning the Google Colab platform | EDA, learning the Google Colab platform | Learning the use of Google Colab, Pycharm version out-dated, compatibility of code between Pycharm and Google Colab. | There is not much support materials for data set without images |
| Week 12 | PDA and PPT, prepare for the speech and interview | PDA | Code for Model Evaluation Metric 3- ROC-AUC curve. Both desktop and laptop are too slow to run the model calculations. | The code provided for metric 3 only supports binary class but I have 5 classes, I have to find external support to identify and fix errors. |
| Week 13 | Finalise PPT, interview and speech; Streamlit deployment, final report. GitHub account creation | PPT, speech and interview. Final report. GitHub account creation | Deployment of ROC curve with Streamlit. Integrating algorithm function with Streamlit.  Use of GitHub. | As I have never learned coding for machine learning. I have to find external support to fix code as it is well beyond my coding skill.  Deployment of certain function with Streamlit is much difficult than I expected. |