IDENTIFICATION OF RICE VARIETIES

Term End Milestone-1 (Project -1)

CODE

Prashant Raghuwanshi (2223-1)

DSC630-T301 Predictive Analytics (2223-1)

Professor Catie Williams

DSC, Bellevue University

04/03/2022

Each phase of the process:

1. Business understanding
   - A. Assess the Current Situation
     - a. Inventory of resources
     - b. Requirements, assumptions and constraints
     - c. Risks and contingencies
     - d. Terminology
     - e. Costs and benefits
   - B. What are the Desired Outputs
   - C. What Questions Are We Trying to Answer?
2. Data Understanding
   - A. Initial Data Report
   - B. Describe Data
   - C. Initial Data Exploration
   - D. Verify Data Quality
     - a. Missing Data
     - b. Outliers
   - E. Data Quality Report
3. Data Preparation
   - A. Select Your Data
   - B. Cleanse the Data
     - a. Label Encoding
     - b. Drop Unnecessary Columns
     - c. Altering Datatypes
     - d. Dealing With Zeros

C. Construct Required Data
D. Integrate Data
4. Exploratory Data Analysis
5. Modelling
A. Modelling Technique
B. Modelling Assumptions
C. Build Model
D. Assess Model
6. Evaluation
7. Deployment

Template Source : https://www.sv-europe.com/crisp-dm-methodology/

Abstract:

This term-end project-1 aims to evaluate the Students should be able to identify a business problem to address through predictive analytics. The goal is to select appropriate models and model specifications and apply the respective methods to enhance data-driven decision-making related to the business problem. Students will identify the potential use of predictive analytics, formulate the problem, identify the right sources of data, analyze data and prescribe actions to improve not only the process of decision making but also the outcome of decisions. In this project, we are using datasets that contain, five different varieties of rice belonging to the same trademark were selected to carry out classification operations using morphological, shape, and color features. A total of 75 thousand rice grain images, including 15 thousand for each variety, were obtained. The images were pre-processed using MATLAB software and prepared for feature extraction. Using a combination of 12 morphological, 4 shape features, and 90 color features obtained from five different color spaces, a total of 106 features were extracted from the images. For classification, models were created with algorithms using machine learning techniques of k-nearest neighbor, decision tree, logistic regression, multilayer perceptron, random forest, and support vector machines. With these models, performance measurement values were obtained for feature sets of 12, 16, 90, and 106. Among the models, the success of the algorithms with the highest average classification accuracy was achieved 97.99% with random forest for morphological features. 98.04% were obtained with random forest for morphological and shape features. It was achieved with logistic regression as 99.25% for color features. Finally, 99.91% was obtained with multilayer perceptron for morphological, shape, and color features. When the results are examined, it is observed that with the addition of each new feature, the success of classification increases. Based on the performance measurement values obtained, it is possible to say that the study achieved success in classifying rice varieties.

1. Stage One - Determine Business Objectives and Assess the Situation

1.1 Assess the Current Situation

The modern Food Processing industry is importing grains (rice) from various international grains distributors. Their producted packed foods mostly depend on quality and varieties of imported raw gains. Even though placing the same type of rice variety order by food processing companies, most of the time Multiple gains distributors supplies the adulterated mix with the ordered rice varieties, and it results in causing the quality degradation and inconsistent taste of packed food and at last, it impacts the sales of processed food product in the competitive market place. At present food processing companies are using random sampling and manual grain monitoring techniques to make sure the procured variety of rice is good. However, this technique is not giving consistent results, since it depends on the individual human eye for identifying the quality of the grain from a single sample. Most of the time due to lack of expertise human eyes are not able to detect the ambiguities in a sample.

1.1.1. Inventory of resources

List the resources available to the project including:

- Personnel: Prashant Raghuwanshi
- Data: Rice_MSC_Dataset
- Computing resources: Personal computers
- Software: Jupyter Notebook(Python)

1.1.2. Requirements, assumptions and constraints -

Requirements : This Project is trying to make use of machine learning techniques to automatically identify the variety of the rice in the given rice sample. Here I am planning to feed the data for the rice to the ML model and the ML model will detect the rice sample and send a signal to the grain sampling machine’s sensors to pick out the ambiguous rice variety from the sample.

assumption: Here I am assuming the Preprocessing operations were applied to the rice images was applied successfully and made available data for feature extraction is not having any issue.

constraints : Major Constraints are related to used datasets and processed images, here the used datasets contain a total of 75 thousand rice grain images, including 15 thousand for each variety however due to the rapidly advancing Seed development process, we might not have all full collections of grain records under each gain varieties.

1.1.3.Risks and contingencies

- Risks include potential PII issues with respondents, which can largely be mitigated by anonymized respondent IDs. As long as a particular individual is never personally idenitified the risk of PII information leaking is mitigated.

- Potential for respondents to misrepresent their individual situations in their survey replies. Survey responses are to some extent unverifiable. For example, if a respondent indicates that the reason they remain unemployed is a long term medical issue, we can't verify that claim and that could potentially skew results.

- Lack of suitable candidates for a given application based on survey responses. This can be mitigated by the model indicating viability within the candidate pool.

1.1.4.Terminology

CRISP-DM The Cross-Industry Standard Process for Data-Mining – CRISP-DM is a model of a data mining process used to solve problems by experts. The model identifies the different stages in implementing a data mining project, as described bellow The model proposes the following steps: • Business Understanding – to understand the rules and business objectives of the company. • Understanding Data – to collect and describe data. • Data Preparation – to prepare data for import into the software. • Modeling – to select the modeling technique to be used. • Evaluation – Evaluate the process to see if the technique solves modeling and creating rules. • Deployment – to deploy the system and train its users

1.1.5.Costs and benefits

Costs: • The primary cost associated with this project is the time of the people working on it. • Computing resources for modeling • Data collection and processing computing costs

Benefits: • Social benefit of this model is helping the food processing companies to automatically detect the variety of rice in the provided sample of rice and helping the authority to stop low-cost mixing and adulteration practices of grain traders • Financial benefit to the company from the ability to maintain the brand quality of processed food which results in increasing the brand loyalty for consumers and its sales.

1.2 What Questions Are We Trying To Answer?

- Targeted Parameters:
- Can we identify survey respondent segments that are candidates for potential packaging as demographic data to sell to our customers?
- How can we determine how non technical professional time use data impacts ability of customers to find candidates for employment?
- How can we determine targeted parameters based on employer skill set requirements?
- How can we determine how technical professional time use data impacts ability of customers to find candidates for employment?
- Does a respondent's family status have an impact on their employment?
- Does the working status of a respondent's spouse impact their employment?
- What impact does non work time usage (Free time/Spending time with Children, etc.) have on respondent employment choices?

2. Stage Two - Data Understanding

The second stage of the CRISP-DM process requires you to acquire the data listed in the project resources. This initial collection includes data loading, if this is necessary for data understanding. For example, if you use a specific tool for data understanding, it makes perfect sense to load your data into this tool. If you acquire multiple data sources then you need to consider how and when you're going to integrate these.

A total of 75 thousand pieces of rice grain were obtained, including 15 thousand pieces of each variety of rice (Arborio, Basmati, Ipsala, Jasmine, Karacadag). Preprocessing operations were applied to the images and made available for feature extraction. A total of 106 features were inferred from the images; 12 morphological features and 4 shape features were obtained using morphological features and 90 color features were obtained from five different color spaces (RGB, HSV, Lab*, YCbCr, XYZ).

2.1 Initial Data Report

Initial data collection report - List the data sources acquired together with their locations, the methods used to acquire them and any problems encountered. Record problems you encountered and any resolutions achieved. This will help both with future replication of this project and with the execution of similar future projects.

In [29]:
```
# Import Libraries Required
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns
pd.set_option("display.max_columns", None)
import warnings
warnings.filterwarnings(action='ignore')
from matplotlib.pyplot import xticks
from sklearn import preprocessing
from kmodes.kmodes import KModes
```

In [2]:
```
#Data source:
#Source Query location:
path =  'C:/Users/21313711/Documents/DSC680/Rice_MSC_Dataset/Rice_MSC_Dataset.xlsx'
# reads the data from the file - denotes as CSV, it has no header, sets column headers
df =  pd.read_excel(path)
```

In [3]:
```
df.shape
```

Out[3]:
```
(75000, 107)
```

2.2 Describe Data

Attribute Information:

1.) Area: Returns the number of pixels 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.) Major Axis Length: The longest line that can be drawn on the rice grain, i.e. the main axis distance, gives.

4.) Minor Axis Length: 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.) Convex Area: Returns the pixel count of the smallest convex shell of the region formed by the rice grain.

7.) Extent: Returns the ratio of the regionformed by the rice grain to the bounding box pixels.

8.) Class: Cammeo and Osmancik rices

In [4]:
```
df.columns
```

Out[4]:
```
Index(['AREA', 'PERIMETER', 'MAJOR_AXIS', 'MINOR_AXIS', 'ECCENTRICITY',
       'EQDIASQ', 'SOLIDITY', 'CONVEX_AREA', 'EXTENT', 'ASPECT_RATIO',
       ...
       'ALLdaub4L', 'ALLdaub4a', 'ALLdaub4b', 'ALLdaub4Y', 'ALLdaub4Cb',
       'ALLdaub4Cr', 'ALLdaub4XX', 'ALLdaub4YY', 'ALLdaub4ZZ', 'CLASS'],
      dtype='object', length=107)
```

In [5]:
```
df.shape
```

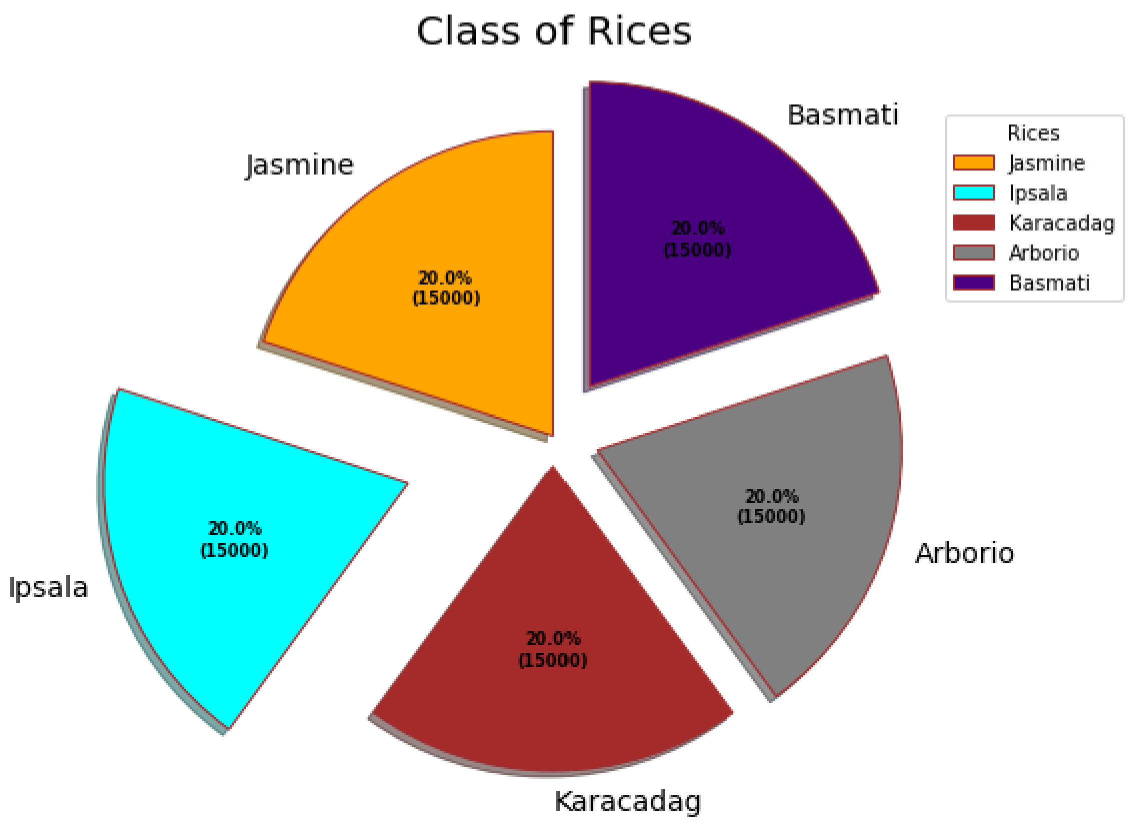
Out[5]:
```
(75000, 107)
```

In [6]:
```
label = df["CLASS"].value_counts().index
value = df["CLASS"].value_counts().values
explode = (0.0, 0.5, 0.1, 0.15, 0.2)
colors = ( "orange", "cyan", "brown","grey", "indigo")
wp = { 'linewidth' : 1, 'edgecolor' : "brown" }

def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    return "{:.1f}%\n({:d})".format(pct, absolute)

fig, ax = plt.subplots(figsize =(10, 7))
wedges, texts, autotexts = ax.pie(value,
                                  autopct = lambda pct: func(pct, value),
                                  explode = explode,
                                  labels = label,
                                  shadow = True,
                                  colors = colors,
                                  startangle = 90,
                                  wedgeprops = wp,
                                  textprops = dict(color ="k",fontsize=14))

ax.legend(wedges, label,
          title ="Rices",
          loc ="center left",
          bbox_to_anchor =(1, 0, 0.8, 1.6))
plt.setp(autotexts, size = 8, weight ="bold")
ax.set_title("Class of Rices",fontsize=20)
plt.show()
```



In [7]:

```
df.describe()
```

Out[7]:

	AREA	PERIMETER	MAJOR_AXIS	MINOR_AXIS	ECCENTRICITY	EQDIASQ	SOLIDI
count	75000.000000	75000.000000	75000.000000	75000.000000	75000.000000	75000.000000	75000.000
mean	8379.197507	378.169453	161.805540	66.829335	0.886077	101.731251	0.975
std	3119.209274	70.597008	36.461005	16.689269	0.071906	17.874070	0.007
min	3929.000000	261.040000	96.968300	34.673000	0.627700	70.728800	0.877
25%	6259.000000	316.431500	132.623500	49.650200	0.846100	89.270400	0.970
50%	7345.000000	351.261000	149.343950	69.183900	0.885600	96.705500	0.976
75%	8901.000000	444.986000	197.462025	75.814125	0.950800	106.457100	0.982
max	21019.000000	593.698000	255.647200	113.441100	0.986800	163.591600	0.992

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75000 entries, 0 to 74999
Columns: 107 entries, AREA to CLASS
```

```
dtypes: float64(95), int64(11), object(1)
memory usage: 61.2+ MB
```

In [9]:
```
df.head(5)
```

Out[9]:

	AREA	PERIMETER	MAJOR_AXIS	MINOR_AXIS	ECCENTRICITY	EQDIASQ	SOLIDITY	CONVEX_AREA
0	7805	437.915	209.8215	48.0221	0.9735	99.6877	0.9775	7985
1	7503	340.757	138.3361	69.8417	0.8632	97.7400	0.9660	7767
2	5124	314.617	141.9803	46.5784	0.9447	80.7718	0.9721	5271
3	7990	437.085	201.4386	51.2245	0.9671	100.8622	0.9659	8272
4	7433	342.893	140.3350	68.3927	0.8732	97.2830	0.9831	7561

2.3 Verify Data Quality

Examine the quality of the data, addressing questions such as:

- Is the data complete (does it cover all the cases required)?
- Is it correct, or does it contain errors and, if there are errors, how common are they?
- Are there missing values in the data? If so, how are they represented, where do they occur, and how common are they?

2.3.1. Missing Data

In addition to incorrect datatypes, another common problem when dealing with real-world data is missing values. These can arise for many reasons and have to be either filled in or removed before we train a machine learning model. First, let’s get a sense of how many missing values are in each column

While we always want to be careful about removing information, if a column has a high percentage of missing values, then it probably will not be useful to our model. The threshold for removing columns should depend on the problem

In [10]:
```
# checking null value
df.isnull().sum()
```

Out[10]:
```
AREA            0
PERIMETER       0
MAJOR_AXIS      0
MINOR_AXIS      0
ECCENTRICITY    0
               ..
ALLdaub4Cr      0
ALLdaub4XX      0
ALLdaub4YY      0
ALLdaub4ZZ      0
CLASS           0
Length: 107, dtype: int64
```

In [11]:
```
# This function take a dataframe
# as a parameter and returning list
# of column names whose contents
# are duplicates.
def getDuplicateColumns(df):

    # Create an empty set
    duplicateColumnNames = set()

    # Iterate through all the columns
    # of dataframe
    for x in range(df.shape[1]):

        # Take column at xth index.
        col = df.iloc[:, x]

        # Iterate through all the columns in
        # DataFrame from (x + 1)th index to
        # last index
        for y in range(x + 1, df.shape[1]):

            # Take column at yth index.
            otherCol = df.iloc[:, y]

            # Check if two columns at x & y
            # index are equal or not,
            # if equal then adding
            # to the set
            if col.equals(otherCol):
                duplicateColumnNames.add(df.columns.values[y])

    # Return list of unique column names
    # whose contents are duplicates.
    return list(duplicateColumnNames)
```

In [12]:
```
drop_columns2= getDuplicateColumns(df)
# list duplicate data columns
drop_columns2
```

Out[12]:
```
[]
```

In [13]:
```
# drop columns from df whose of values are duplicate
df.drop(drop_columns2, axis=1, inplace=True)
```

In [14]:
```
# filling out missing values
df = df.fillna(method="ffill")
df = df.fillna(method="bfill")
```

3. Stage Three - Data Preperation

This is the stage of the project where you decide on the data that you're going to use for analysis. The criteria you might use to make this decision include the relevance of the data to your data mining goals, the quality of the data, and also technical constraints such as limits on data volume or data types. Note that data selection covers selection of attributes (columns) as well as selection of records (rows) in a table.

3.1 Label Encoding

In [15]:

```
df.dtypes
```

Out[15]:

```
AREA              int64
PERIMETER       float64
MAJOR_AXIS      float64
MINOR_AXIS      float64
ECCENTRICITY    float64
                 ...
ALLdaub4Cr      float64
ALLdaub4XX      float64
ALLdaub4YY      float64
ALLdaub4ZZ      float64
CLASS            object
Length: 107, dtype: object
```

In [26]:

```
#dataset is not balanced. define a function that finds the minimum number of samples in
def trim(df2, column):
    df3=df2.copy()
    sample_list=[]
    balance=list(df3[column].value_counts())
    min_samples=np.min(balance) # least samples in any class
    print ('the minimum number of samples in any class is ', min_samples)
    min_size = 0
    groups=df3.groupby(column)
    for label in df3[column].unique():
        group=groups.get_group(label)
        sample_count=len(group)
        if sample_count> min_samples :
            samples=group.sample(min_samples, replace=False, weights=None, random_state
            sample_list.append(samples)
        elif sample_count>= min_size:
            sample_list.append(group)
    df=pd.concat(sample_list, axis=0).reset_index(drop=True)
    return df3
```

- All fields are already label encoded. No need to change data types.
- May potentially update to One Hot Encoding.

3.2.2 Drop Unnecessary Columns

Sometimes we may not need certain columns. We can drop to keep only relevent data

3.2 Dealing With Zeros

Replacing all the zeros from cols. **Note** You may not want to do this - add / remove as required

- Zero values were previously addressed using mean value imputation.

3.3 Construct Required Data

This task includes constructive data preparation operations such as the production of derived attributes or entire new records, or transformed values for existing attributes.

Derived attributes - These are new attributes that are constructed from one or more existing attributes in the same record, for example you might use the variables of length and width to calculate a new variable of area.

Generated records - Here you describe the creation of any completely new records. For example you might need to create records for customers who made no purchase during the past year. There was no reason to have such records in the raw data, but for modelling purposes it might make sense to explicitly represent the fact that particular customers made zero purchases.

- Do not have derivative records. No construction required.

4. Stage Four - Exploratory Data Analysis

In [16]:
```
df.head()
```

Out[16]:

	AREA	PERIMETER	MAJOR_AXIS	MINOR_AXIS	ECCENTRICITY	EQDIASQ	SOLIDITY	CONVEX_AREA
0	7805	437.915	209.8215	48.0221	0.9735	99.6877	0.9775	7985
1	7503	340.757	138.3361	69.8417	0.8632	97.7400	0.9660	7767
2	5124	314.617	141.9803	46.5784	0.9447	80.7718	0.9721	5271
3	7990	437.085	201.4386	51.2245	0.9671	100.8622	0.9659	8272
4	7433	342.893	140.3350	68.3927	0.8732	97.2830	0.9831	7561

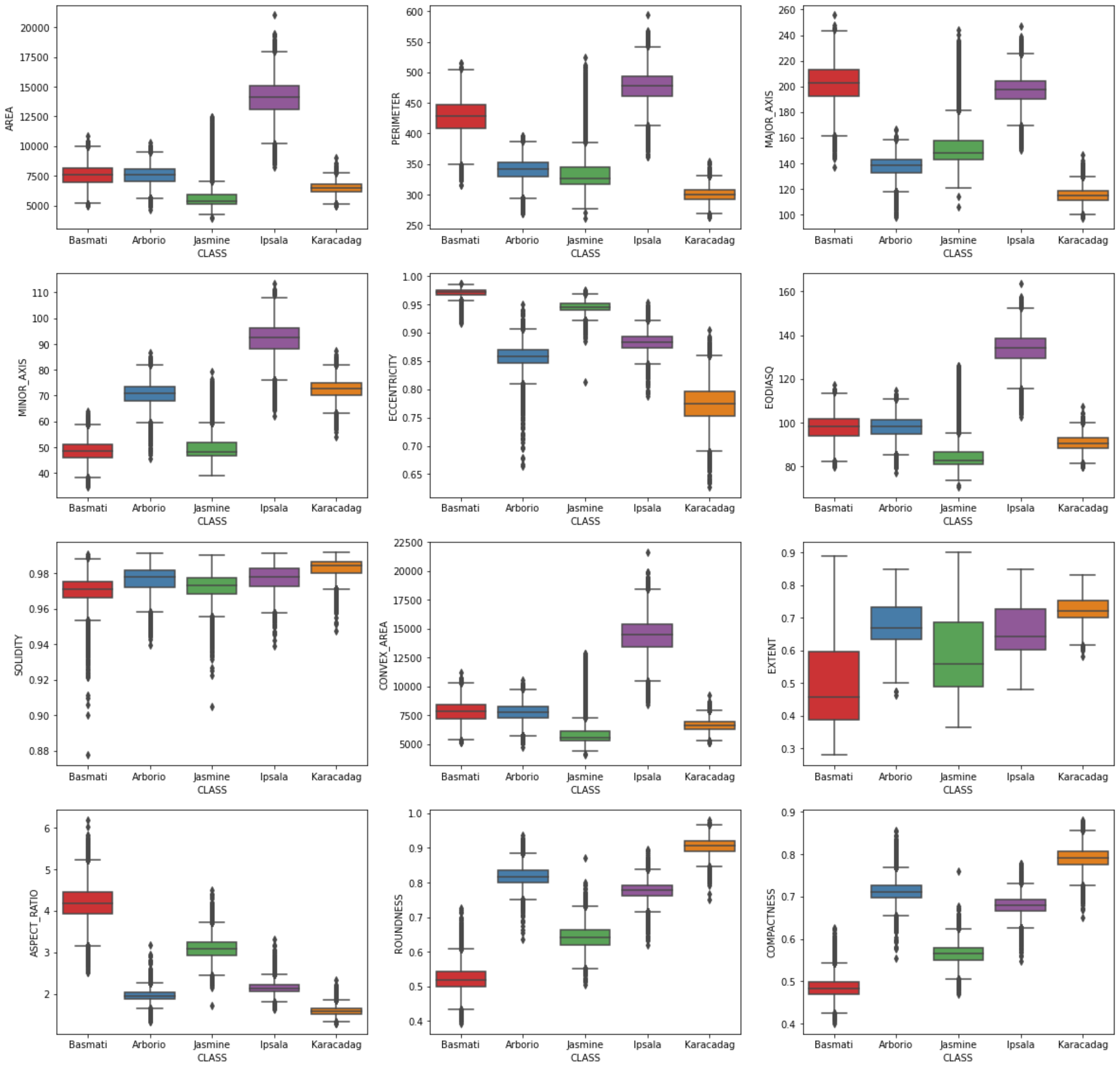
4.1. Outliers

At this point, we may also want to remove outliers. These can be due to typos in data entry, mistakes in units, or they could be legitimate but extreme values. For this project, we will remove anomalies based on the definition of extreme outliers:

https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm

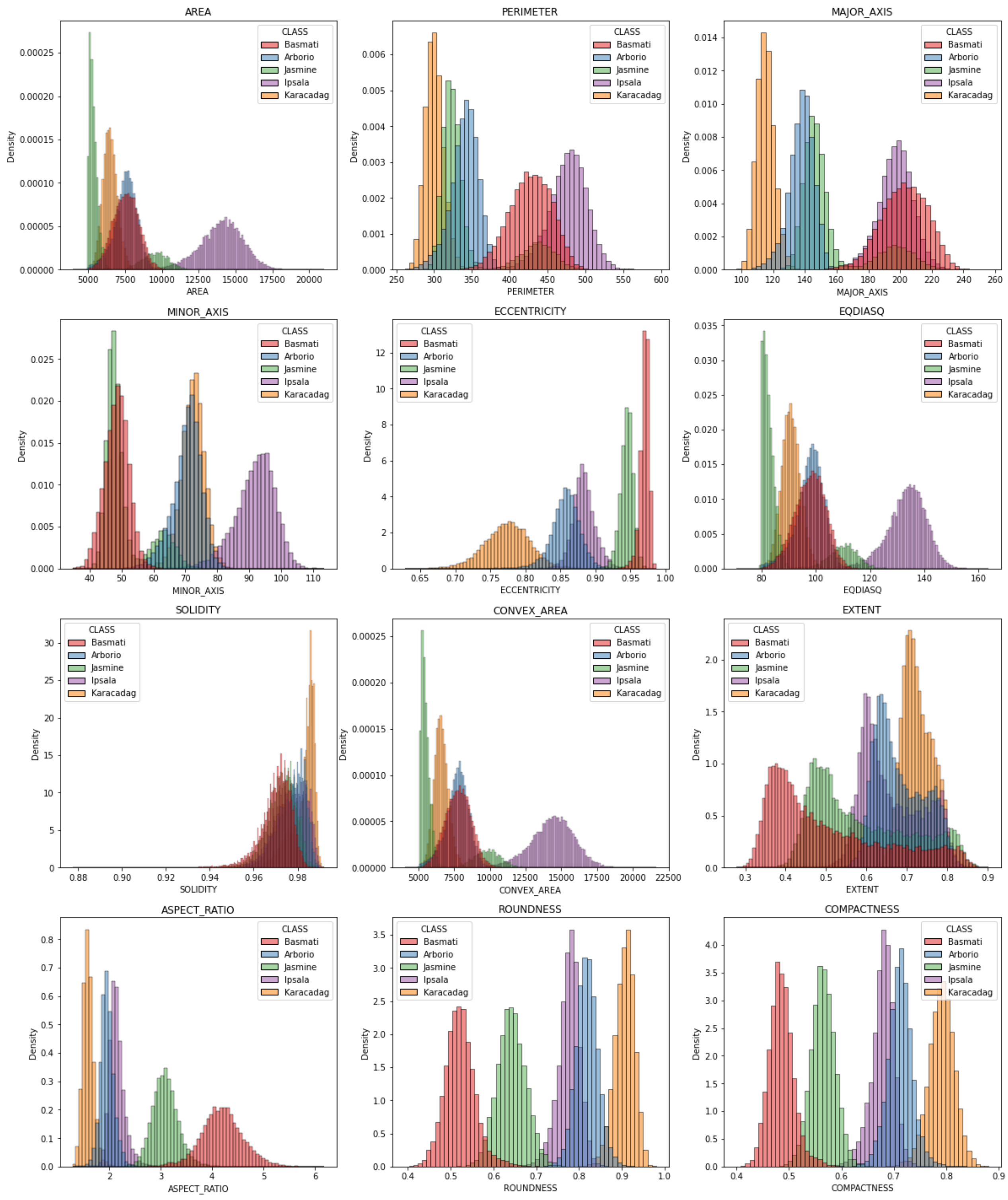
- Below the first quartile − 3 ∗ interquartile range
- Above the third quartile + 3 ∗ interquartile range

In [19]:
```
plt.figure(figsize=(20,20))
for i in range(12):
    plt.subplot(4, 3, i + 1)
    sns.boxplot(x="CLASS", y=df.columns[i], data=df, palette="Set1")
plt.show()
```



In [20]:

```
plt.figure(figsize=(20,25))
for i in range(12):
    plt.subplot(4, 3, i + 1)
    sns.histplot(data=df, x=df.columns[i],hue="CLASS", stat="density", palette="Set1")
    plt.title(df.columns[i])
plt.show()
```



4.2 Initial Data Exploration

During this stage you'll address data mining questions using querying, data visualization and reporting techniques. These may include:

- **Distribution** of key attributes (for example, the target attribute of a prediction task)
- **Relationships** between pairs or small numbers of attributes
- Results of **simple aggregations**
- **Properties** of significant sub-populations

- **Simple** statistical analyses

These analyses may directly address your data mining goals. They may also contribute to or refine the data description and quality reports, and feed into the transformation and other data preparation steps needed for further analysis.

- **Data exploration report** - Describe results of your data exploration, including first findings or initial hypothesis and their impact on the remainder of the project. If appropriate you could include graphs and plots here to indicate data characteristics that suggest further examination of interesting data subsets.

4.2.1 Distributions

In [21]:

```
# distrubutions of varieties class records looks ideal
print(df['CLASS'].value_counts())
```

```
Jasmine      15000
Ipsala       15000
Karacadag    15000
Arborio      15000
Basmati      15000
Name: CLASS, dtype: int64
```

4.2.2 Correlations

Can we derive any correlation from this data-set. Pairplot chart gives us correlations, distributions and regression path Correlogram are awesome for exploratory analysis. It allows to quickly observe the relationship between every variable of your matrix. It is easy to do it with seaborn: just call the pairplot function

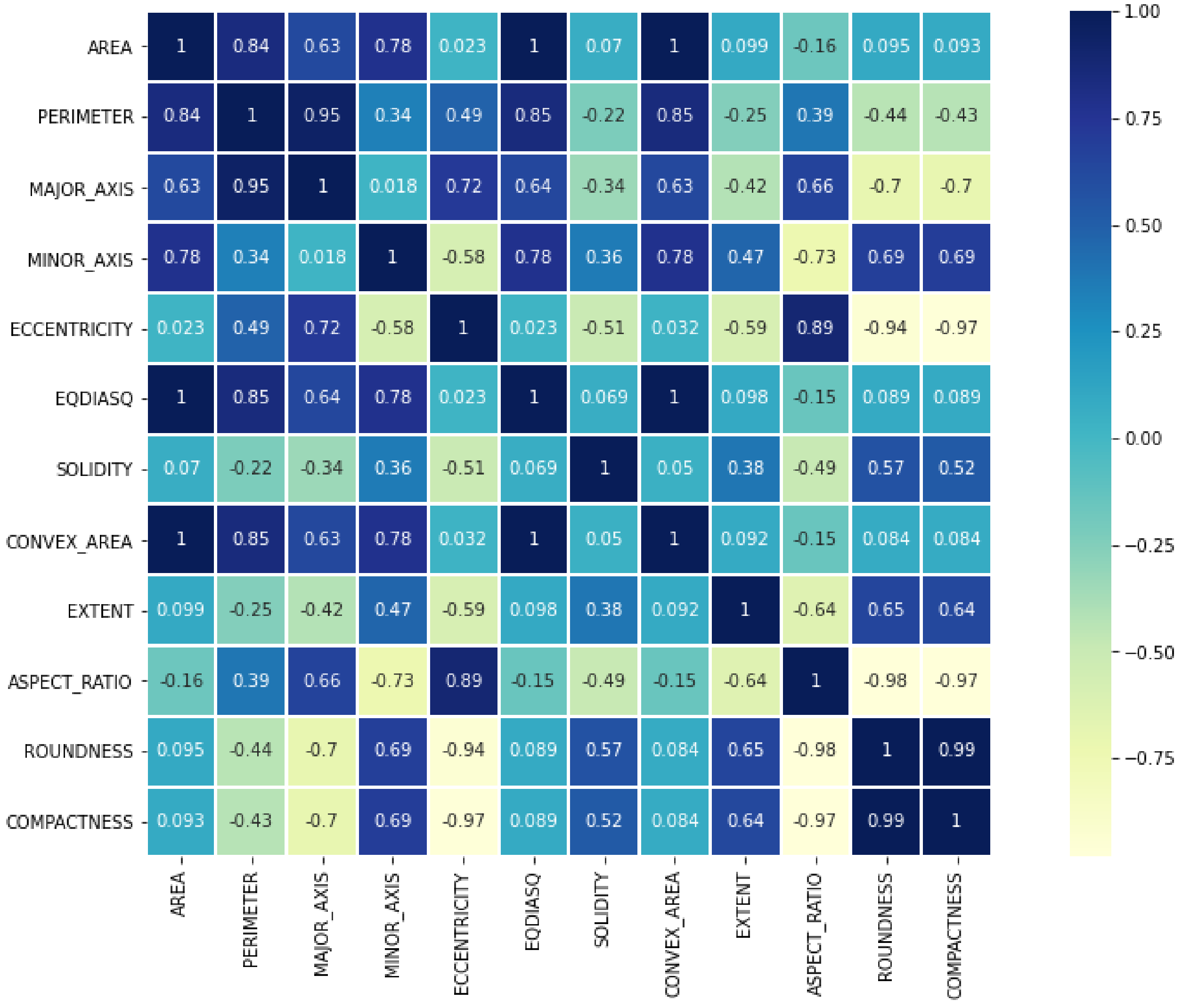
Pairplot Documentation cab be found here: https://seaborn.pydata.org/generated/seaborn.pairplot.html

In [22]:

```
plt.figure(figsize=(16,9))
sns.heatmap(df.iloc[:,:12].corr(), cmap="YlGnBu",annot=True, fmt=".2g", linewidths = 1,
```

Out[22]:

```
<AxesSubplot:>
```



	AREA	PERIMETER	MAJOR_AXIS	MINOR_AXIS	ECCENTRICITY	EQDIASQ	SOLIDITY	CONVEX_AREA	EXTENT	ASPECT_RATIO	ROUNDNESS	COMPACTNESS
AREA	1	0.84	0.63	0.78	0.023	1	0.07	1	0.099	-0.16	0.095	0.093
PERIMETER	0.84	1	0.95	0.34	0.49	0.85	-0.22	0.85	-0.25	0.39	-0.44	-0.43
MAJOR_AXIS	0.63	0.95	1	0.018	0.72	0.64	-0.34	0.63	-0.42	0.66	-0.7	-0.7
MINOR_AXIS	0.78	0.34	0.018	1	-0.58	0.78	0.36	0.78	0.47	-0.73	0.69	0.69
ECCENTRICITY	0.023	0.49	0.72	-0.58	1	0.023	-0.51	0.032	-0.59	0.89	-0.94	-0.97
EQDIASQ	1	0.85	0.64	0.78	0.023	1	0.069	1	0.098	-0.15	0.089	0.089
SOLIDITY	0.07	-0.22	-0.34	0.36	-0.51	0.069	1	0.05	0.38	-0.49	0.57	0.52
CONVEX_AREA	1	0.85	0.63	0.78	0.032	1	0.05	1	0.092	-0.15	0.084	0.084
EXTENT	0.099	-0.25	-0.42	0.47	-0.59	0.098	0.38	0.092	1	-0.64	0.65	0.64
ASPECT_RATIO	-0.16	0.39	0.66	-0.73	0.89	-0.15	-0.49	-0.15	-0.64	1	-0.98	-0.97
ROUNDNESS	0.095	-0.44	-0.7	0.69	-0.94	0.089	0.57	0.084	0.65	-0.98	1	0.99
COMPACTNESS	0.093	-0.43	-0.7	0.69	-0.97	0.089	0.52	0.084	0.64	-0.97	0.99	1

4.3 Data Quality Report

List the results of the data quality verification. If quality problems exist, suggest possible solutions. Solutions to data quality problems generally depend heavily on both data and business knowledge.

- Primary data quality issue is missing values in certain parts of the data. To correct for this, we imputed the mean value of the columns into those missing fields in order to have a more complete approximation of the missing data.

5. Stage Four - Modelling

As the first step in modelling, you'll select the actual modelling technique that you'll be using. Although you may have already selected a tool during the business understanding phase, at this stage you'll be selecting the specific modelling technique e.g. decision-tree building with C5.0, or neural network generation with back propagation. If multiple techniques are applied, perform this task separately for each technique.

In [27]:
```
def preprocess(df):
    df1=df.copy()
    # balance the data set by having samples in each class equal to smallest samples fo
    df1=trim(df1, 'CLASS')
    print (df1['CLASS'].value_counts())
    # partition into target y and data x
    y=df1['CLASS']
    X=df1.drop(['CLASS'], axis=1)
    #split into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, shuffle=T
    # scale the X data
    scaler = StandardScaler()
    scaler.fit(X_train) # fit only on train data
    X_train = pd.DataFrame(scaler.transform(X_train), index=X_train.index, columns=X_tr
    X_test = pd.DataFrame(scaler.transform(X_test), index=X_test.index, columns=X_test.

    return X_train, X_test, y_train, y_test
```

In [30]:

```
X_train, X_test, y_train, y_test=preprocess(df)
print ('X_train length: ', len(X_train), '  X_test length: ', len(X_test))
```

```
the minimum number of samples in any class is  15000
Jasmine      15000
Ipsala       15000
Karacadag    15000
Arborio      15000
Basmati      15000
Name: CLASS, dtype: int64
X_train length:  60000   X_test length:  15000
```

In [31]:

```
print (y_train.value_counts())
```

```
Ipsala       12025
Jasmine      12015
Karacadag    11992
Arborio      11990
Basmati      11978
Name: CLASS, dtype: int64
```

5.1. Modelling technique

Document the actual modelling technique that is to be used.

Import Models below:

In [34]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,AdaBoos
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

5.2. Modelling assumptions

Many modelling techniques make specific assumptions about the data, for example that all attributes have uniform distributions, no missing values allowed, class attribute must be symbolic etc. Record any assumptions made.

-

5.3. Build Model

Run the modelling tool on the prepared dataset to create one or more models.

Parameter settings - With any modelling tool there are often a large number of parameters that can be adjusted. List the parameters and their chosen values, along with the rationale for the choice of parameter settings.

Models - These are the actual models produced by the modelling tool, not a report on the models.

Model descriptions - Describe the resulting models, report on the interpretation of the models and document any difficulties encountered with their meanings.

In [32]:
```
models = {
    "Logistic Regression": LogisticRegression(),
    "      Decision Tree": DecisionTreeClassifier(),
    "     Neural Network": MLPClassifier(),
    "      Random Forest": RandomForestClassifier(),
    "  Gradient Boosting": GradientBoostingClassifier(),
    " AdaBoostClassifier": AdaBoostClassifier(),
    "KNeighborsClassifier": KNeighborsClassifier()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    print(name + " trained.")
```

```
Logistic Regression trained.
      Decision Tree trained.
     Neural Network trained.
      Random Forest trained.
  Gradient Boosting trained.
 AdaBoostClassifier trained.
KNeighborsClassifier trained.
```

5.4. Assess Model

Interpret the models according to your domain knowledge, your data mining success criteria and your desired test design. Judge the success of the application of modelling and discovery techniques technically, then contact business analysts and domain experts later in order to discuss the data mining results in the business context. This task only considers models, whereas the evaluation phase also takes into account all other results that were produced in the course of the project.

At this stage you should rank the models and assess them according to the evaluation criteria. You should take the business objectives and business success criteria into account as far as you can here. In most data mining projects a single technique is applied more than once and data mining results are generated with several different techniques.

Model assessment - Summarise the results of this task, list the qualities of your generated models (e.g.in terms of accuracy) and rank their quality in relation to each other.

Revised parameter settings - According to the model assessment, revise parameter settings and tune them for the next modelling run. Iterate model building and assessment until you strongly believe that you have found the best model(s). Document all such revisions and assessments.

In [43]:

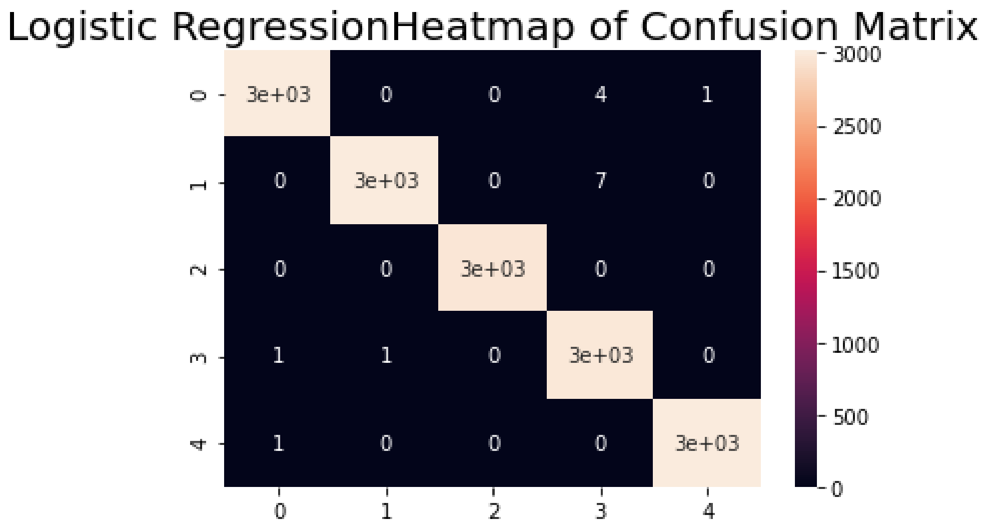
```
for name, model in models.items():
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, pos_label=1, average='weighted')
    print(name + " Accuracy: {:.2f}%".format(acc * 100) + " "+" F1-Score: {:.5f}".forma
```

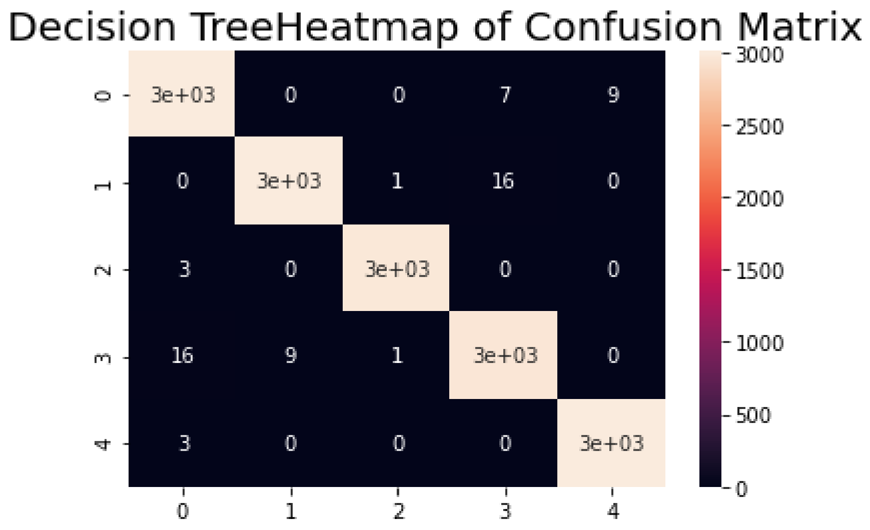
```
Logistic Regression Accuracy: 99.90%  F1-Score: 0.99900
      Decision Tree Accuracy: 99.57%  F1-Score: 0.99567
     Neural Network Accuracy: 99.93%  F1-Score: 0.99927
      Random Forest Accuracy: 99.86%  F1-Score: 0.99860
  Gradient Boosting Accuracy: 99.83%  F1-Score: 0.99833
 AdaBoostClassifier Accuracy: 60.71%  F1-Score: 0.48542
KNeighborsClassifier Accuracy: 99.83%  F1-Score: 0.99827
```

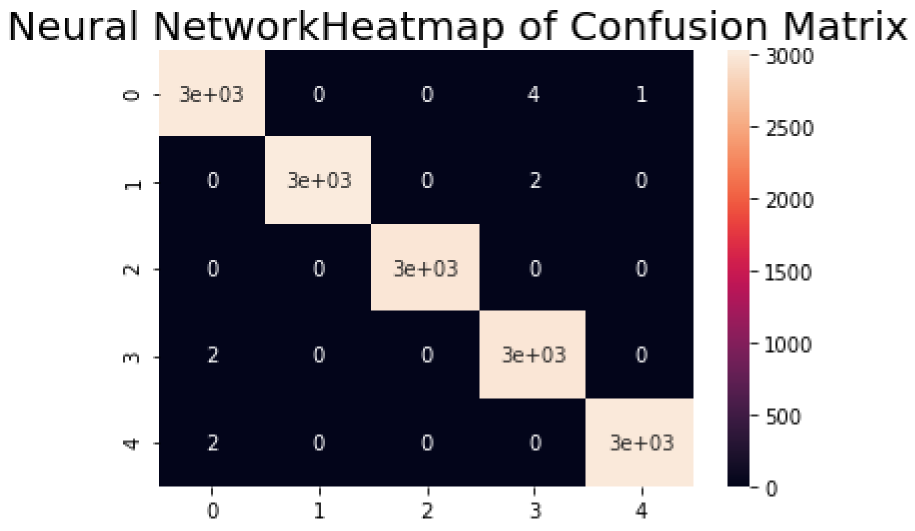
Classification performance measurement results are pasted below. all models except AdaBoostClassifier have been achieved a classification success of over 99%. The 99.93% accuracy achieved in the LR model has the highest value among other models. Also for F1, recall and precision look good for the LR model.

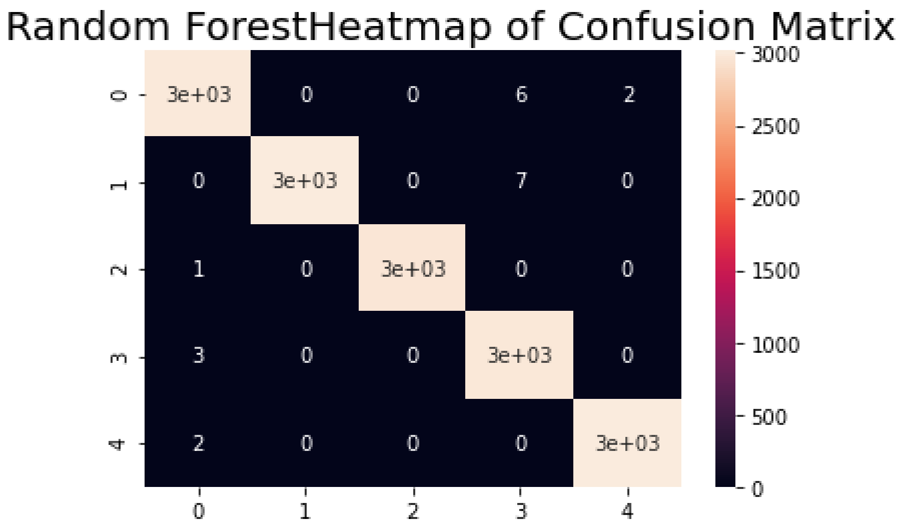
In [37]:

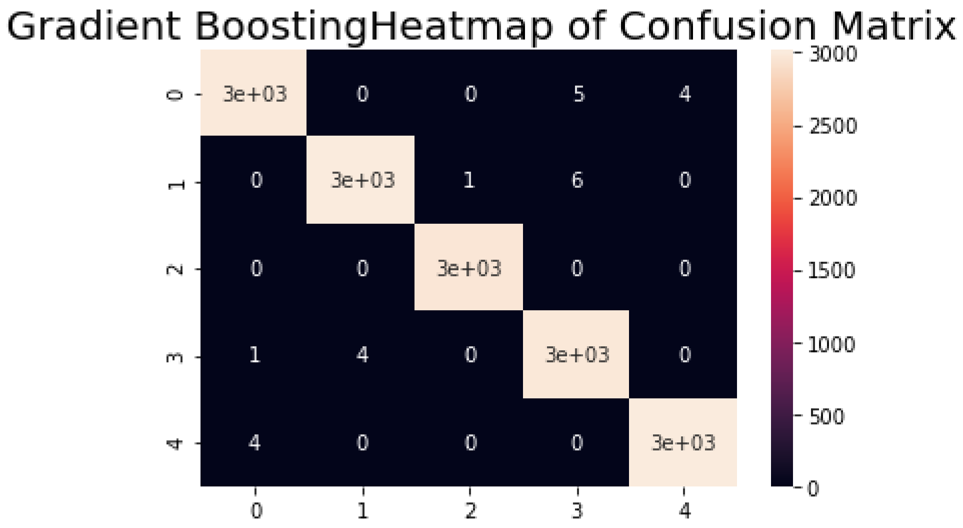
```
for name, model in models.items():
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    ttl = name + 'Heatmap of Confusion Matrix'
    plt.title(ttl, fontsize = 20)
    sns.heatmap(cm, annot = True)
    plt.show()
```

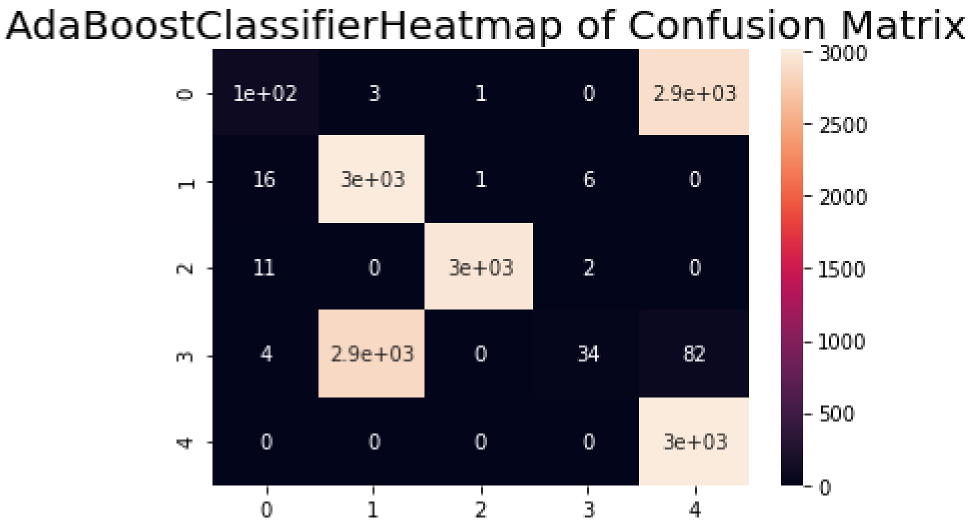

Logistic RegressionHeatmap of Confusion Matrix
3e+03 0 0 4 1
0 3e+03 0 7 0
0 0 3e+03 0 0
1 1 0 3e+03 0
1 0 0 0 3e+03
0 1 2 3 4
3000 2500 2000 1500 1000 500 0

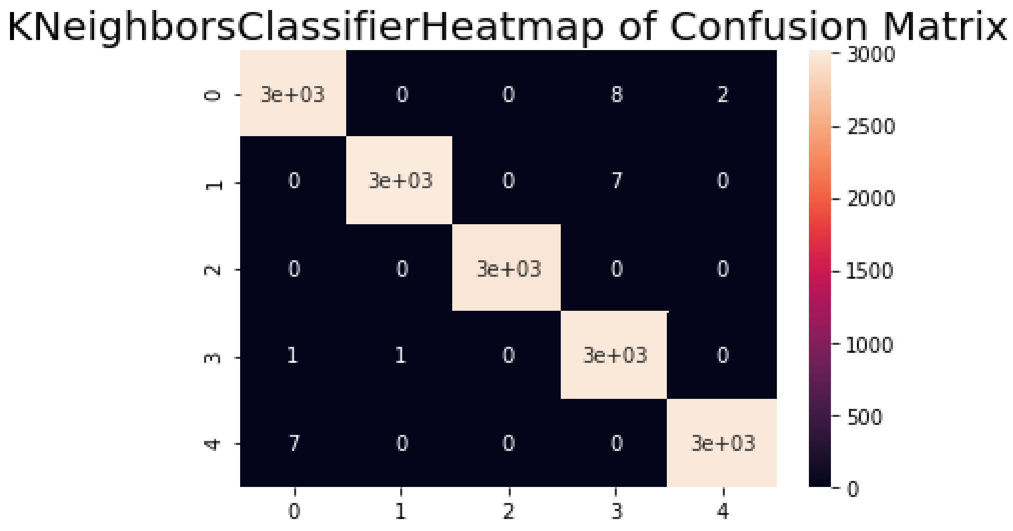

Decision TreeHeatmap of Confusion Matrix
3e+03 0 0 7 9
0 3e+03 1 16 0
3 0 3e+03 0 0
16 9 1 3e+03 0
3 0 0 0 3e+03
0 1 2 3 4
3000 2500 2000 1500 1000 500 0


Neural NetworkHeatmap of Confusion Matrix
3e+03 0 0 4 1
0 3e+03 0 2 0
0 0 3e+03 0 0
2 0 0 3e+03 0
2 0 0 0 3e+03
0 1 2 3 4
3000 2500 2000 1500 1000 500 0


Random ForestHeatmap of Confusion Matrix
3e+03 0 0 6 2
0 3e+03 0 7 0
1 0 3e+03 0 0
3 0 0 3e+03 0
2 0 0 0 3e+03
0 1 2 3 4
3000 2500 2000 1500 1000 500 0


Gradient BoostingHeatmap of Confusion Matrix
3e+03 0 0 5 4
0 3e+03 1 6 0
0 0 3e+03 0 0
1 4 0 3e+03 0
4 0 0 0 3e+03
0 1 2 3 4
3000 2500 2000 1500 1000 500 0


AdaBoostClassifierHeatmap of Confusion Matrix
1e+02 3 1 0 2.9e+03
16 3e+03 1 6 0
11 0 3e+03 2 0
4 2.9e+03 0 34 82
0 0 0 0 3e+03
0 1 2 3 4
3000 2500 2000 1500 1000 500 0



In [50]:
```
for name, model in models.items():
    y_pred = model.predict(X_test)
    cm = classification_report(y_test, y_pred)
    ttl = name + 'classification_report\n'
    print(ttl)
    print(cm)
```

```
Logistic Regressionclassification_report

              precision    recall  f1-score   support

     Arborio       1.00      1.00      1.00      3010
     Basmati       1.00      1.00      1.00      3022
      Ipsala       1.00      1.00      1.00      2975
     Jasmine       1.00      1.00      1.00      2985
   Karacadag       1.00      1.00      1.00      3008

    accuracy                           1.00     15000
   macro avg       1.00      1.00      1.00     15000
weighted avg       1.00      1.00      1.00     15000

      Decision Treeclassification_report

              precision    recall  f1-score   support

     Arborio       0.99      0.99      0.99      3010
     Basmati       1.00      0.99      1.00      3022
      Ipsala       1.00      1.00      1.00      2975
     Jasmine       0.99      0.99      0.99      2985
   Karacadag       1.00      1.00      1.00      3008

    accuracy                           1.00     15000
   macro avg       1.00      1.00      1.00     15000
weighted avg       1.00      1.00      1.00     15000

     Neural Networkclassification_report

              precision    recall  f1-score   support

     Arborio       1.00      1.00      1.00      3010
     Basmati       1.00      1.00      1.00      3022
      Ipsala       1.00      1.00      1.00      2975
     Jasmine       1.00      1.00      1.00      2985
   Karacadag       1.00      1.00      1.00      3008
```

```
    accuracy                           1.00     15000
   macro avg       1.00      1.00      1.00     15000
weighted avg       1.00      1.00      1.00     15000

      Random Forestclassification_report

              precision    recall  f1-score   support

     Arborio       1.00      1.00      1.00      3010
     Basmati       1.00      1.00      1.00      3022
      Ipsala       1.00      1.00      1.00      2975
     Jasmine       1.00      1.00      1.00      2985
   Karacadag       1.00      1.00      1.00      3008

    accuracy                           1.00     15000
   macro avg       1.00      1.00      1.00     15000
weighted avg       1.00      1.00      1.00     15000

  Gradient Boostingclassification_report

              precision    recall  f1-score   support

     Arborio       1.00      1.00      1.00      3010
     Basmati       1.00      1.00      1.00      3022
      Ipsala       1.00      1.00      1.00      2975
     Jasmine       1.00      1.00      1.00      2985
   Karacadag       1.00      1.00      1.00      3008

    accuracy                           1.00     15000
   macro avg       1.00      1.00      1.00     15000
weighted avg       1.00      1.00      1.00     15000

 AdaBoostClassifierclassification_report

              precision    recall  f1-score   support

     Arborio       0.77      0.03      0.07      3010
     Basmati       0.51      0.99      0.67      3022
      Ipsala       1.00      1.00      1.00      2975
     Jasmine       0.81      0.01      0.02      2985
   Karacadag       0.50      1.00      0.67      3008

    accuracy                           0.61     15000
   macro avg       0.72      0.61      0.49     15000
weighted avg       0.72      0.61      0.49     15000

KNeighborsClassifierclassification_report

              precision    recall  f1-score   support

     Arborio       1.00      1.00      1.00      3010
     Basmati       1.00      1.00      1.00      3022
      Ipsala       1.00      1.00      1.00      2975
     Jasmine       0.99      1.00      1.00      2985
   Karacadag       1.00      1.00      1.00      3008

    accuracy                           1.00     15000
   macro avg       1.00      1.00      1.00     15000
weighted avg       1.00      1.00      1.00     15000
```

Conclusion: The logistic regression model is the best fit for rice gains predictions. We can use this classification model to develop Automatic systems that can be designed for many processes such as calibration of rice types and the separation of species from unwanted substances that may be present.

Assumption:
Here I am assuming the Preprocessing data extraction operations is not having any limitations and the rice grains images were extracted & preprocessed successfully by image to data conversion tool.

Limitations: This Model is limited to identifying the Rice grains only. This model is suitable to identify Arborio, Basmati, Uppsala, Jasmine, and Karacadag rice varieties and would consider other varieties as unknown gain.

Future Uses: 1) By adding additional grains datasets, we can increase the scope of this model to identify large varieties of grains. 2) Programmed in robot scanner machines and enable robots to filter adulterated gains

Recommendations: Automatic systems can be designed for many processes such as calibration of rice types and the separation of species from unwanted substances that may be present. To increase the success rate in classification, more images can be obtained from species and it is thought that success rates can be increased by using morphological features as well as color and shape features.

```
In [ ]:
```