# Machine Learning Report

**Prepared for Dr. Yehia, Eng. Mohammed Shawkey.**

- Contents
  - Exploratory Data Analysis
  - Logistic Regression Classifier
  - Random Forest Classifier
  - SVM Classifier

- Work Load

**Exploratory Data Analysis**→Salma Ragab Hassan, Aya Ahmed Musad

**Modeling**→Shredan Abdullah Kamal, Nada Osman AbdElAziz

## Team #14

| Name | SEC | BN |
|---|---|---|
| Nada Osman AbdElAziz | 2 | 30 |
| Salma Ragab Hassan | 1 | 30 |
| Shredan Abdullah Kamal | 1 | 33 |
| Aya Ahmed Musad | 1 | 14 |

# I. Problem Definition and Motivation:

The problem is to develop a classification model that accurately identifies different varieties of date fruits based on their characteristics. Given the attributes such as size, color, texture, taste, and other features, the model should classify each date fruit into one of the **seven classes: Barhee, Deglet Nour, Sukkary, Rotab Mozafati, Ruthana, Safawi, and Sagai.**

1. Agricultural Industry Improvement: Date fruits are a significant agricultural product in many regions worldwide. Accurate classification can help in better management of date palm orchards, optimizing harvesting, and improving overall agricultural practices.

2. Market Segmentation: Different varieties of date fruits have distinct flavors and textures, appealing to different consumer preferences. Accurate classification can aid in market segmentation, allowing businesses to target specific consumer demographics more effectively.

### Evaluation Metrics:

**1. Precision:** Precision measures the proportion of correctly identified instances of a specific class among all instances classified as that class. It helps assess the model's accuracy in identifying true positives while minimizing false positives.

**2. Recall:** Recall calculates the proportion of correctly identified instances of a specific class among all instances belonging to that class in the dataset. It evaluates the model's ability to identify all positive instances without missing any.

**3. F1-score:** F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, making it a useful metric for binary and multiclass classification tasks.

### Link for dataset from Kaggle:

https://www.kaggle.com/datasets/muratkokludataset/date-fruit-datasets/data

# II. Exploratory Data Analysis

### About Dataset

7 Class; Barhee, Deglet Nour, Sukkary, Rotab Mozafati, Ruthana, Safawi, Sagai.

| Name | Data Types | Default Task | Attribute Types | # Instances | # Attributes | Year | Download |
|---|---|---|---|---|---|---|---|
| **Date Fruit Datasets** | 7 Class | Classification Clustering | Integer, Real | 898 | 34 | 2021 | Download 5453 downloaded |
| **Citation Request** | KOKLU, M., KURSUN, R., TASPINAR, Y. S. and CINAR, I. (2021). Classification of Date Fruits into Genetic Varieties Using Image Analysis. *Mathematical Problems in Engineering*, Vol.2021, Article ID: 4793293. **DOI:** https://doi.org/10.1155/2021/4793293 | | | | | | |

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 898 entries, 0 to 897
Data columns (total 35 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   AREA           898 non-null    int64
 1   PERIMETER      898 non-null    float64
 2   MAJOR_AXIS     898 non-null    float64
 3   MINOR_AXIS     898 non-null    float64
 4   ECCENTRICITY   898 non-null    float64
 5   EQDIASQ        898 non-null    float64
 6   SOLIDITY       898 non-null    float64
 7   CONVEX_AREA    898 non-null    int64
 8   EXTENT         898 non-null    float64
 9   ASPECT_RATIO   898 non-null    float64
 10  ROUNDNESS      898 non-null    float64
 11  COMPACTNESS    898 non-null    float64
 12  SHAPEFACTOR_1  898 non-null    float64
 13  SHAPEFACTOR_2  898 non-null    float64
 14  SHAPEFACTOR_3  898 non-null    float64
 15  SHAPEFACTOR_4  898 non-null    float64
 16  MeanRR         898 non-null    float64
 17  MeanRG         898 non-null    float64
 18  MeanRB         898 non-null    float64
 19  StdDevRR       898 non-null    float64
...
 33  ALLdaub4RB     898 non-null    float64
 34  Class          898 non-null    object
dtypes: float64(29), int64(5), object(1)
memory usage: 245.7+ KB
```

### Dataset Checking:

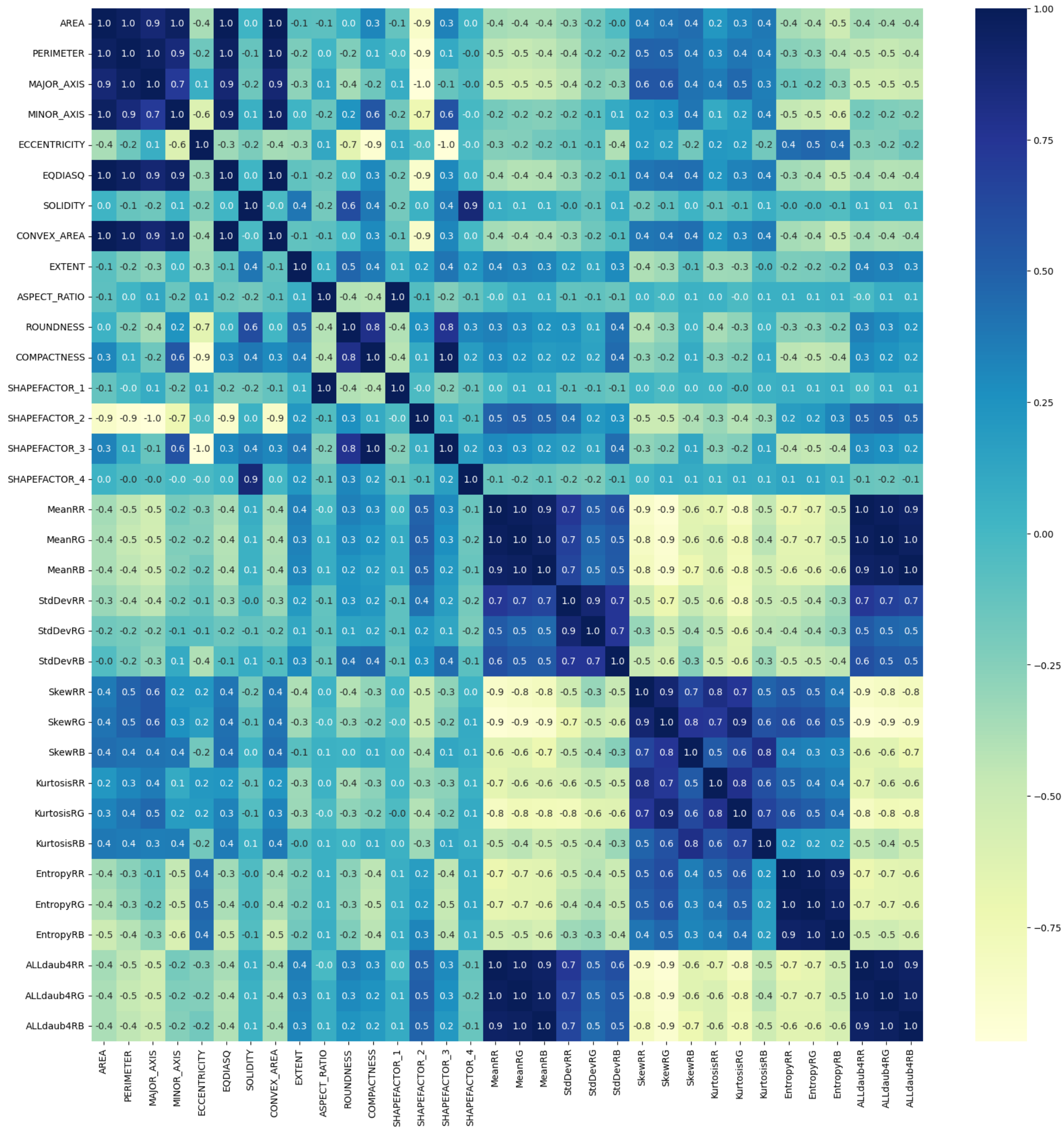
Checking the duplicate values in the dataset

Checking Is there any null values in the dataset

Checking the dataset is balanced or not

## Data Analysis:

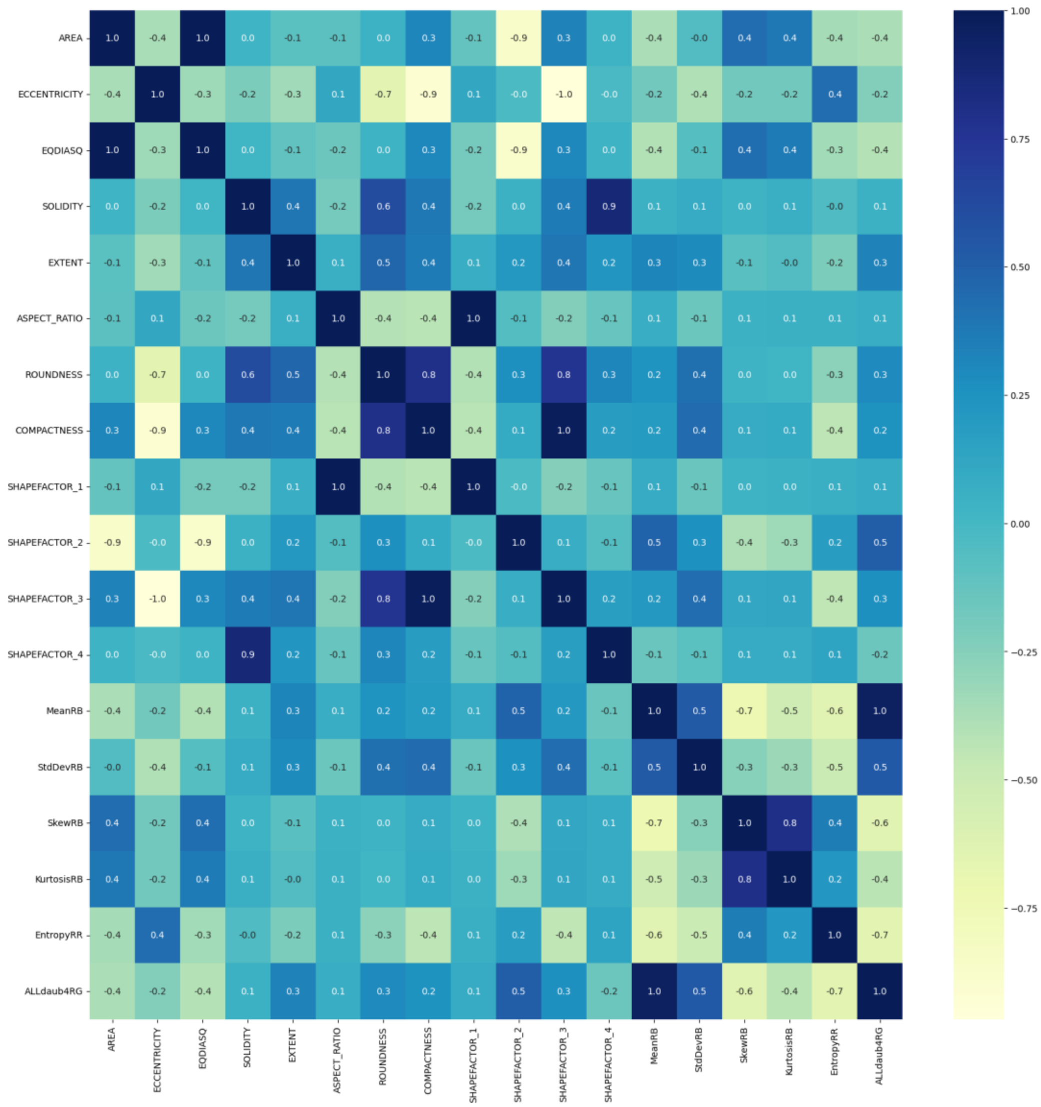
### The correlation between features/columns.



## Observations :

Dropping the columns that has Correlation equal to 1(Multicolinearity)

"PERIMETER", "MAJOR_AXIS", "MINOR_AXIS", "CONVEX_AREA", "MeanRR", "ALLdaub4RR", "EntropyRG", "MeanRG","StdDevRR","StdDevRG","ALLdaub4RB","EntropyRB","SkewRG","SkewRR","KurtosisRR","KurtosisRG".



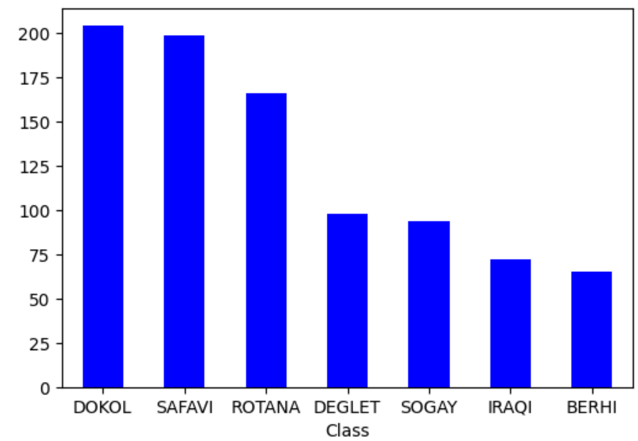

Before

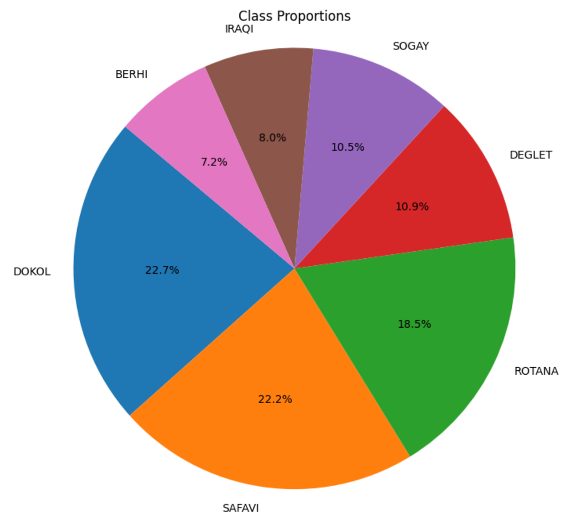
| | Recall | Precision | F1_Score |
|---|---|---|---|
| LogisticRegression | 0.902778 | 0.902778 | 0.902778 |
| SVC | 0.902778 | 0.902778 | 0.902778 |
| RandomForestClassifier | 0.881944 | 0.881944 | 0.881944 |

After

| | Recall | Precision | F1_Score |
|---|---|---|---|
| RandomForestClassifier | 0.909722 | 0.909722 | 0.909722 |
| LogisticRegression | 0.888889 | 0.888889 | 0.888889 |
| SVC | 0.868056 | 0.868056 | 0.868056 |

**Check the dataset is balanced or not.**



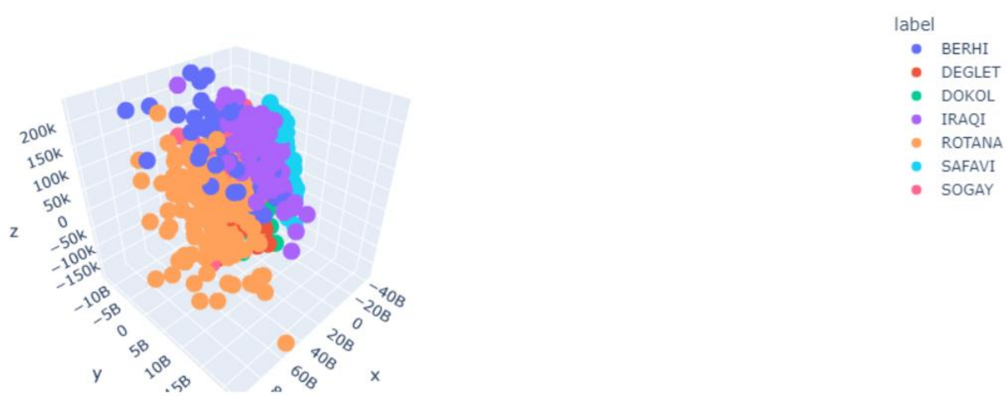




**Observations :**

The dataset is not perfectly balanced as the proportions of samples for each class are not exactly equal.

**visualize the data in 3D space**

to visualize the data in 3D space we perform dimensionality reduction using PCA to the original features.



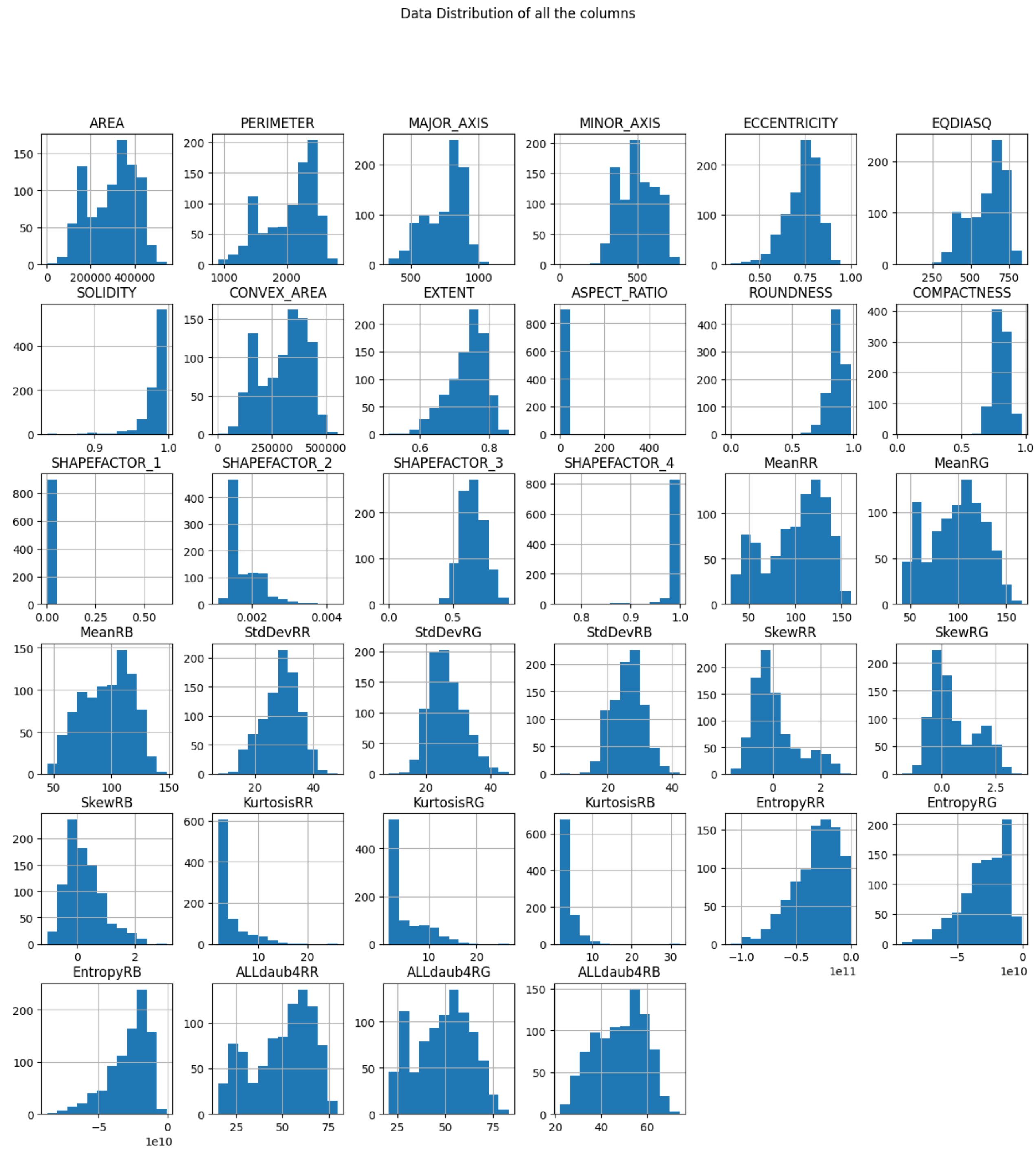

**ZeroR Algorithm**

we use the zero-r algorithm to detect the baseline for this problem.

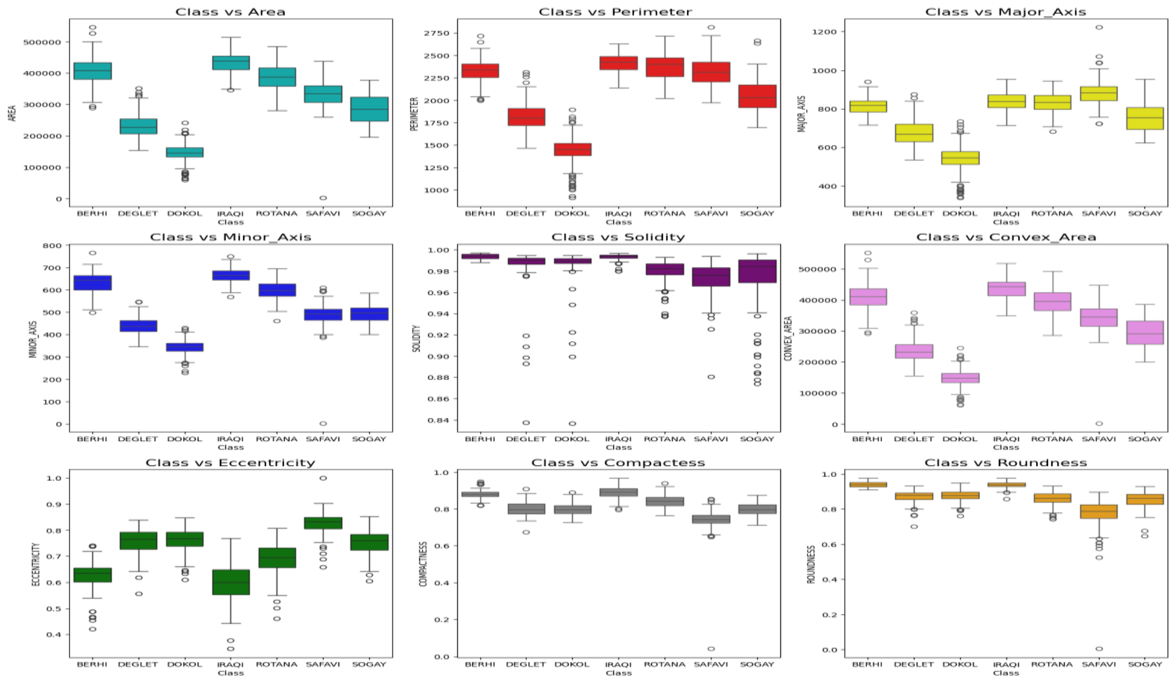
Baseline Class: DOKOL

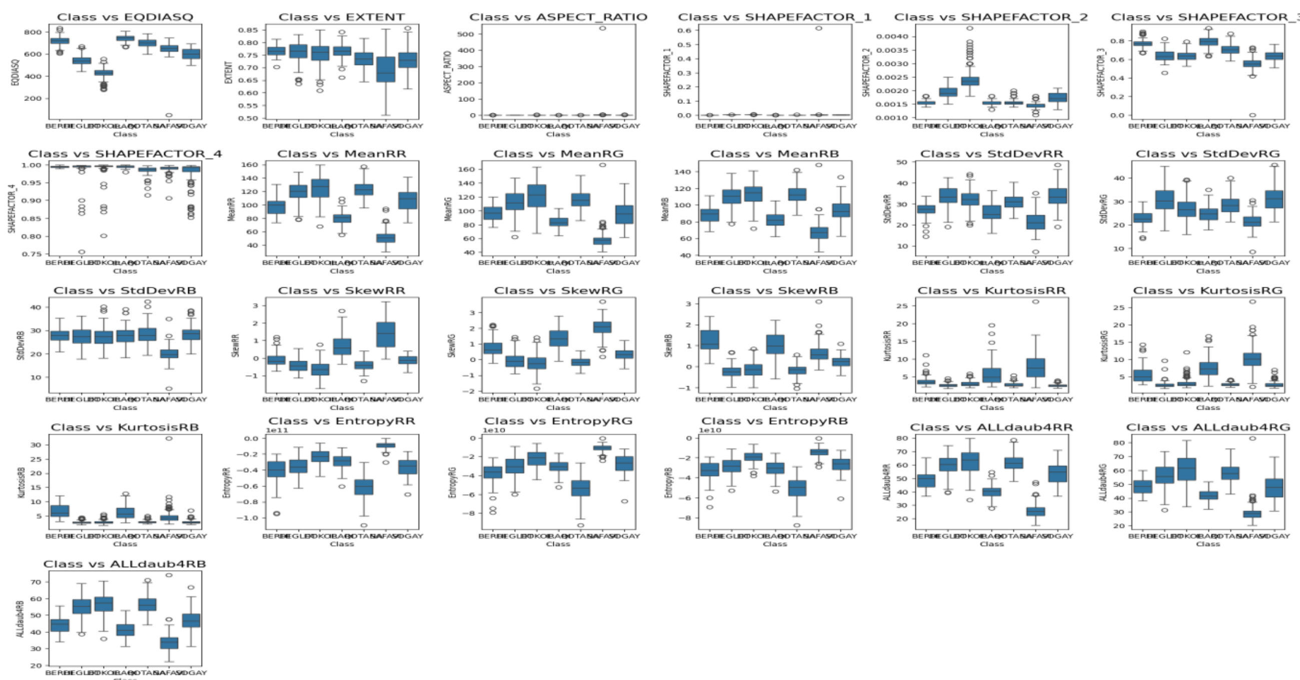
Baseline Proportion: 0.22717149220489977

# Data Distribution of all the columns



## Detect Outliers Using Box Plot




## Detect the outliers in the Training Data And Remove it: (Ratio of outliers:11.32)

Appling the Winsorization method:

Winsorization is a data preprocessing technique used to handle outliers in a dataset by either capping or flooring extreme values at a specified percentile. Instead of removing outliers completely.

**Divide the dataset into training, validation, and test sets:**

→ the initial dataset is split into 80% training and 20% test, and then the training set is further split into 80% training and 20% validation.

→ the initial dataset is split into 70% training and 30% test, and then the training set is further split into 70% training and 30% validation.

**Observations :**

The 80-20 gives better results than that of 70-30.

**Data Preprocessing:**

All the features are numerical except the class, we do Numerical Encoding for numerical features using RobustScaler().

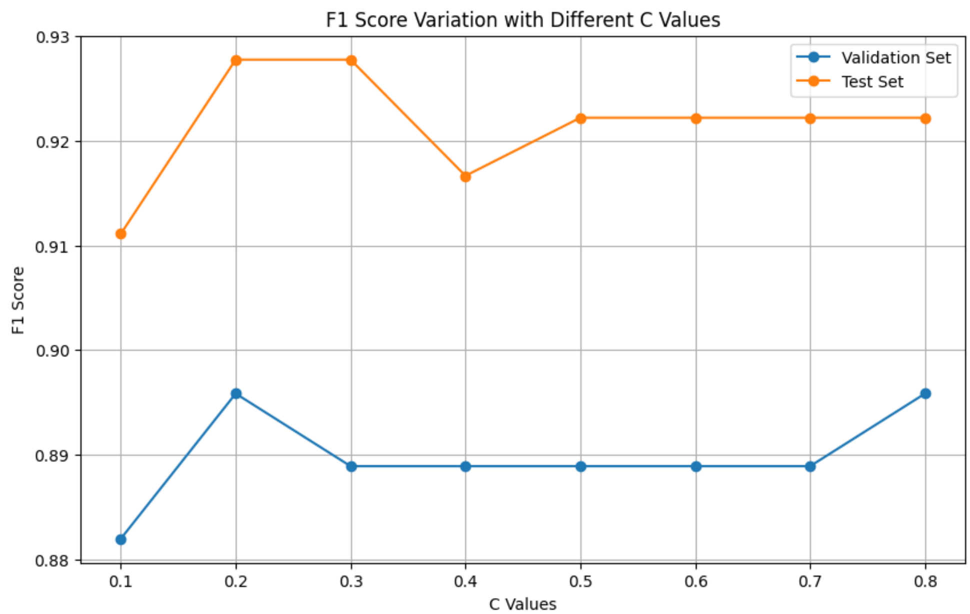
And for class feature we use the Categorical Encoding using LabelEncoder().

# III. Modeling

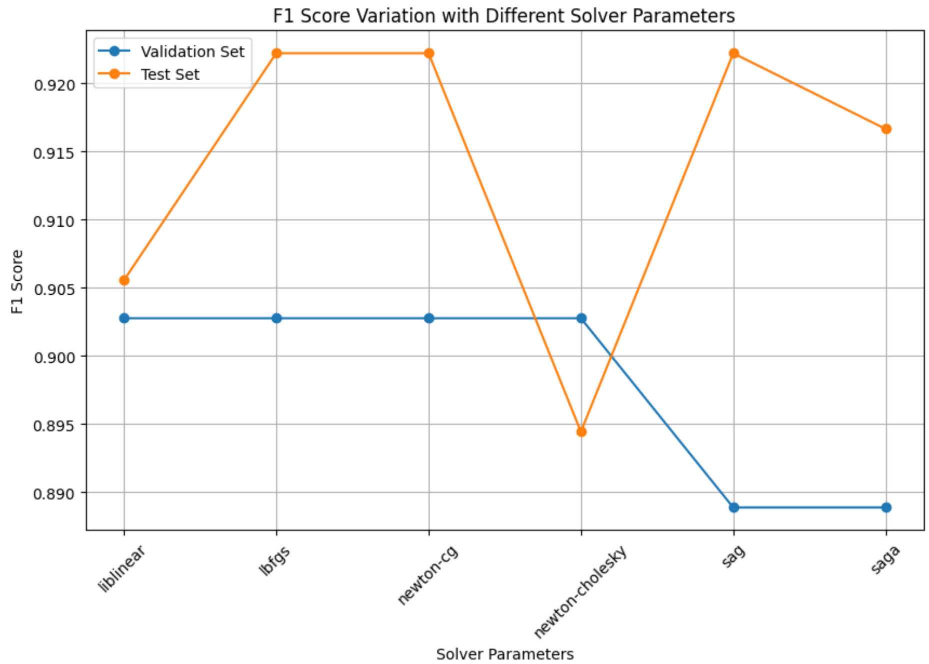
# I. Logistic Regression:

Classification algorithm in machine learning that models the probability of a binary or multi-class target variable based on one or more predictor variables.

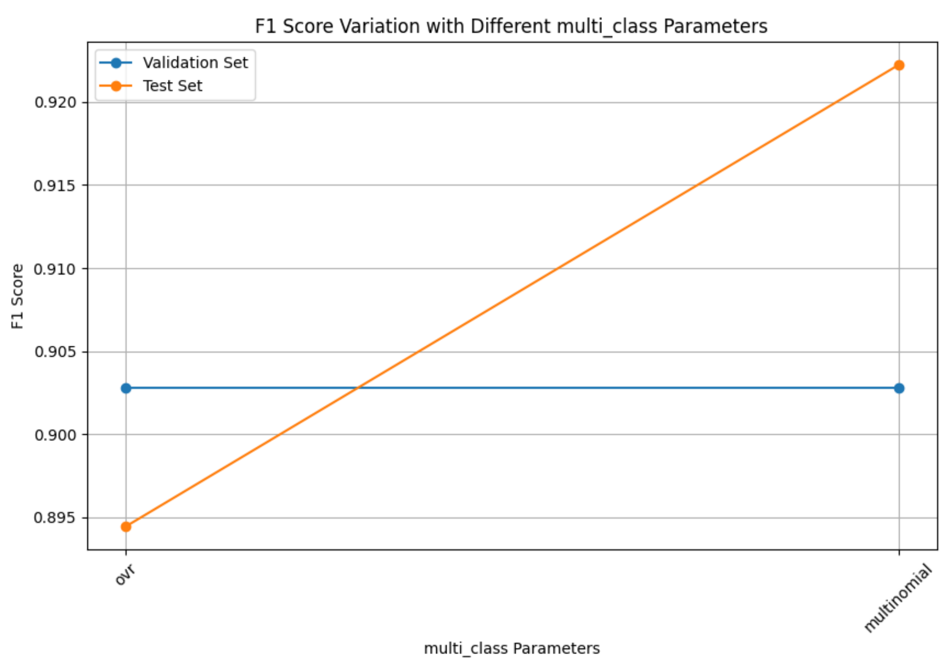
**C Parameter:** The inverse of the regularization strength. Smaller values of C specify stronger regularization.



**Solver Parameter:** The algorithm to be used for optimization.




**multi_class Parameter:** When faced with strong correlations between classes, handling the classification problem as a binary task using the One-vs-Rest (OvR) approach might not yield optimal results. In such scenarios, the multinomial logistic regression model tends to outperform OvR. This is because multinomial logistic regression considers the relationships between all classes simultaneously, making it more effective when classes are highly correlated. Therefore, when dealing with datasets exhibiting strong inter-class correlations.



**penalty Parameter:** Since the decided solver is lbfgs and the regularization types allowed for it is L2 only.

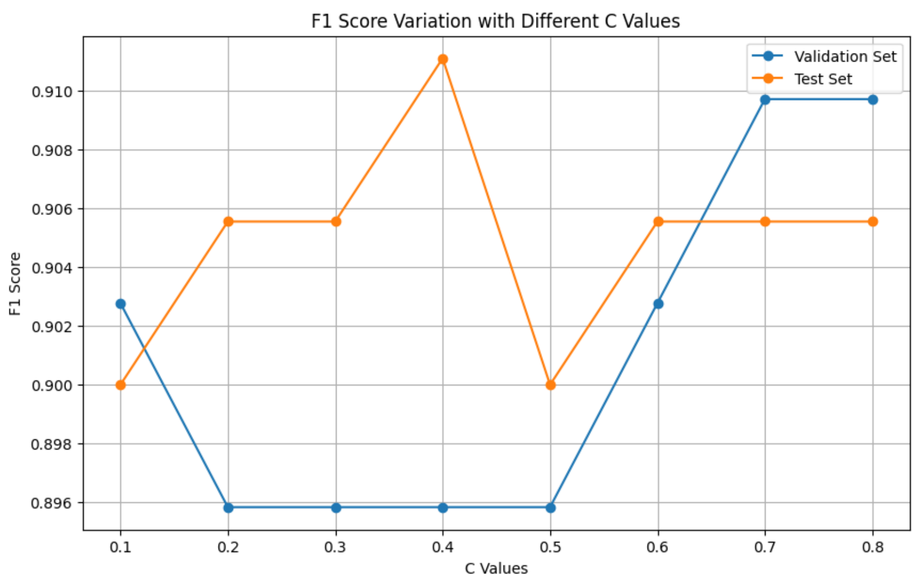
### Over Sampling to handle class imbalance:

oversampling using SMOTE.

```
def oversample_data(x_train, y_train):
    # Instantiate SMOTE
    smote = SMOTE(random_state=42)

    # Apply SMOTE to the training data
    x_train_oversampled, y_train_oversampled = smote.fit_resample(x_train, y_train)

    return x_train_oversampled, y_train_oversampled
```



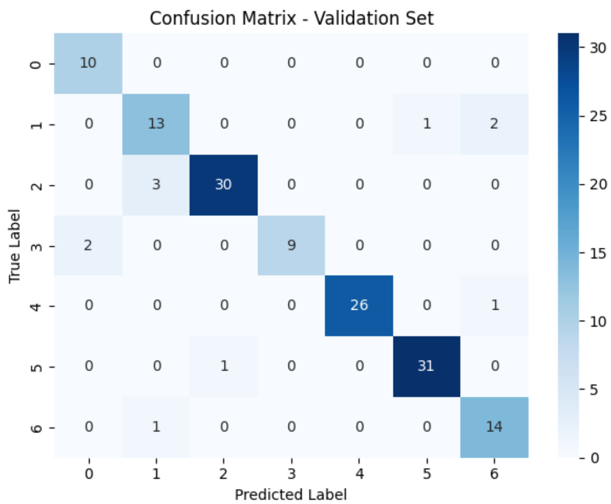

### Cross Validation:

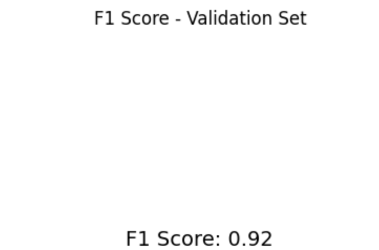
Cross Validation F1 Scores: [0.8907563  0.88235294 0.94957983 0.93277311 0.94117647]

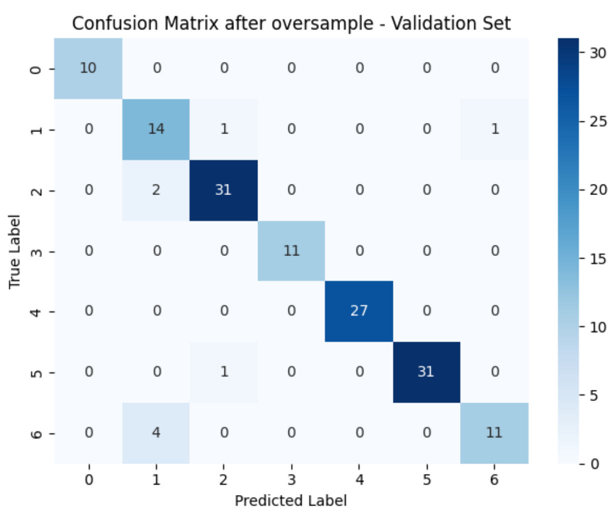
Cross-validation F1 Score: 0.92 +/- 0.03

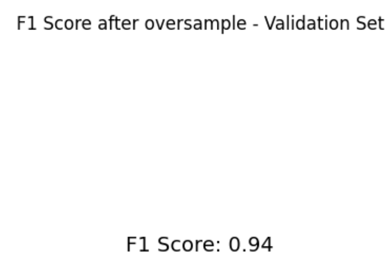
# II. Random Forest

Random Forest is a machine-learning algorithm that improves the decision-tree algorithm by fitting more than one decision-tree classifier on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. It is a bagging technique.



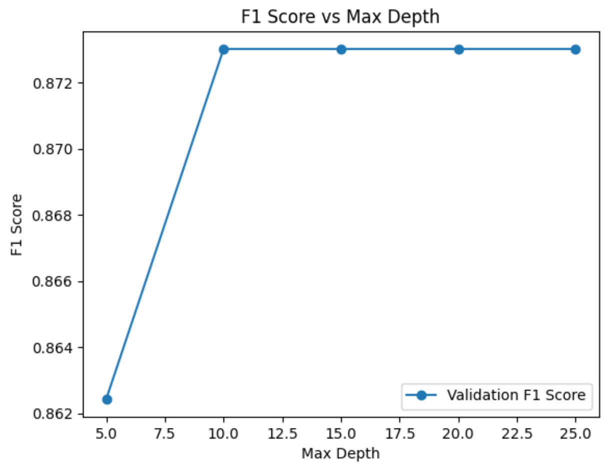



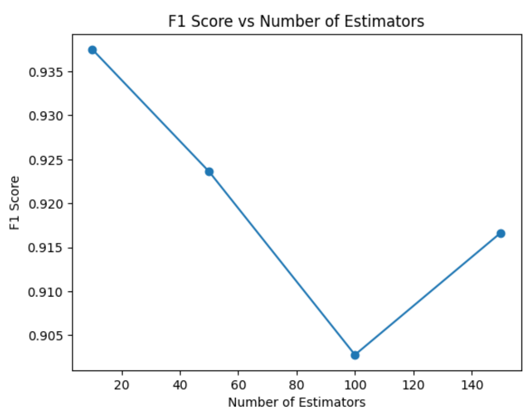



## Parameter Tuning

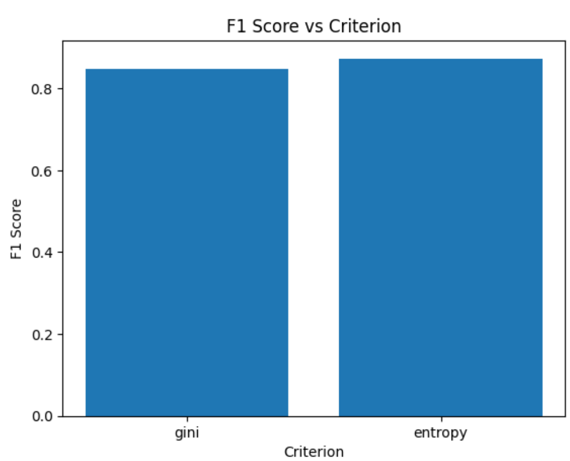
## max_depth:



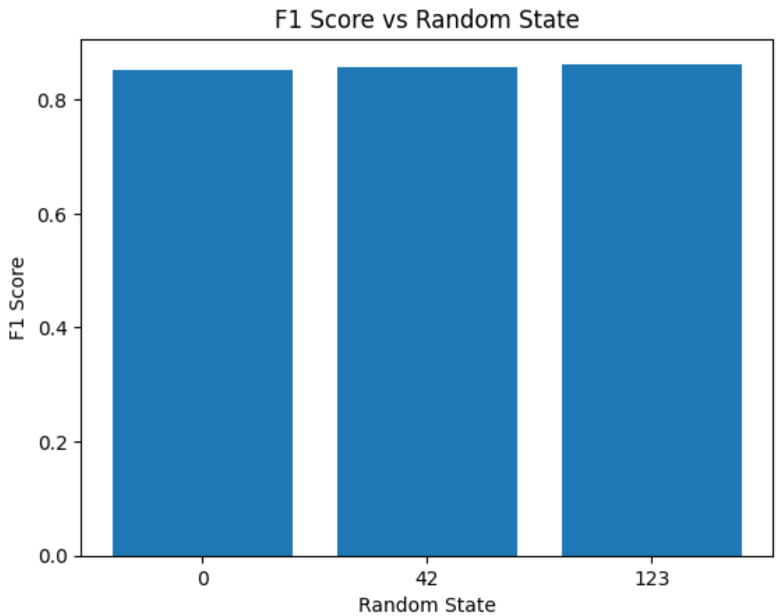

## n_estimators:



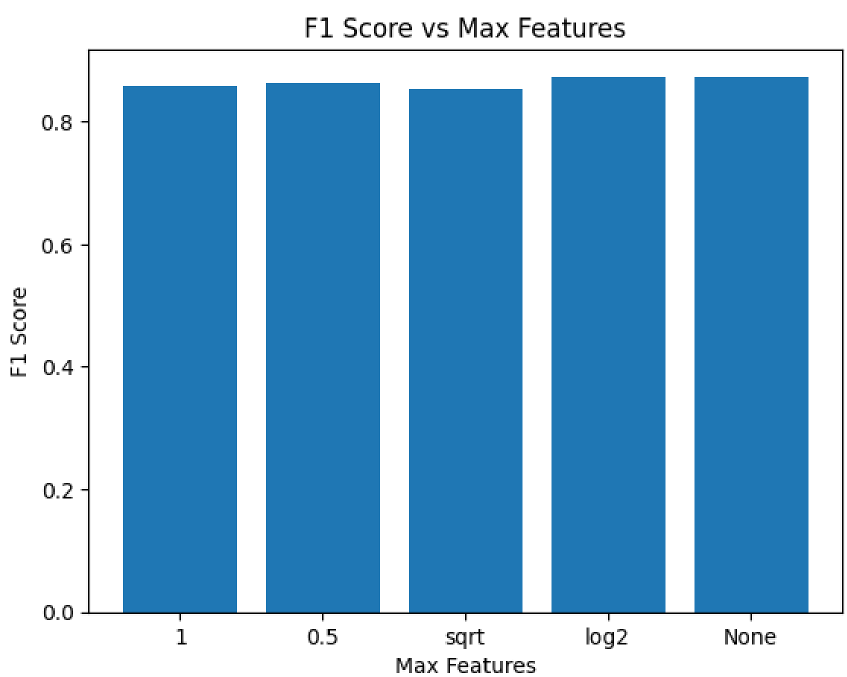

## criterion:



**random_state:**

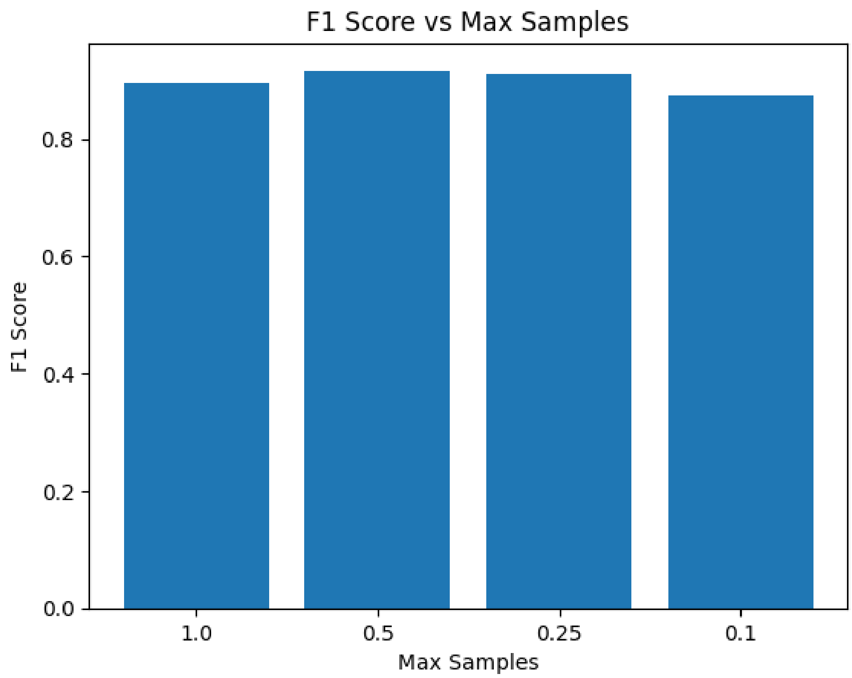



**max_features:**




**class_weight:** using balanced because the data is unbalanced.

**max_samples:**




**bootstrap:** the out-of-bag (OOB) estimation is only available when the bootstrap parameter is set to True

**oob_score:** the out-of-bag (OOB) estimation is only available when oob_score the parameter is set to True

**best tunning:** RandomForestClassifier(n_estimators=10, max_depth=12, random_state=0, criterion="entropy", class_weight="balanced", max_features=None, max_samples=0.5, oob_score=True)

**Cross Validation:**

cross-validation helps us to get the average of model accuracy, this prevents overfitting; because we trained K models and average them.

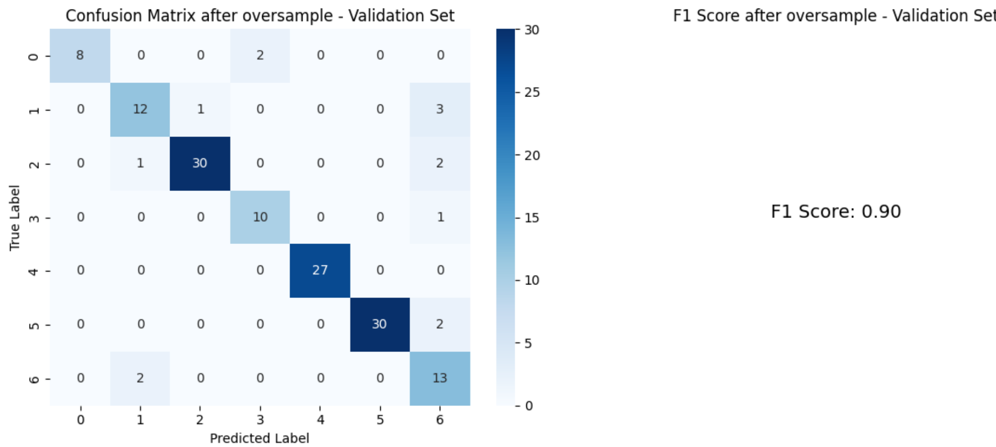
Cross Validation F1 Scores: [0.8627451  0.88235294 0.88235294 0.91447368 0.94736842]

Cross-validation F1 Score: 0.90 +/- 0.03

# III. SVM

to handle the imbalance of the data, we used Resampling using SMOTE.

Without Resampling:




With Resampling using SMOTE:

All Combinations and their F1 Scores for Test Set:

{'kernel': 'linear', 'C': 1.0, 'gamma': 0.1, 'F1 Score': 0.9097222222222222}

{'kernel': 'linear', 'C': 1.0, 'gamma': 0.01, 'F1 Score': 0.9097222222222222}

{'kernel': 'linear', 'C': 1.0, 'gamma': 0.001, 'F1 Score': 0.9097222222222222}

{'kernel': 'linear', 'C': 5.0, 'gamma': 0.1, 'F1 Score': 0.9097222222222222}

{'kernel': 'linear', 'C': 5.0, 'gamma': 0.01, 'F1 Score': 0.9097222222222222}

{'kernel': 'linear', 'C': 5.0, 'gamma': 0.001, 'F1 Score': 0.9097222222222222}

{'kernel': 'linear', 'C': 14.0, 'gamma': 0.1, 'F1 Score': 0.9166666666666666}

{'kernel': 'linear', 'C': 14.0, 'gamma': 0.01, 'F1 Score': 0.9166666666666666}

{'kernel': 'linear', 'C': 14.0, 'gamma': 0.001, 'F1 Score': 0.9166666666666666}

{'kernel': 'rbf', 'C': 1.0, 'gamma': 0.1, 'F1 Score': 0.8958333333333334}

{'kernel': 'rbf', 'C': 1.0, 'gamma': 0.01, 'F1 Score': 0.8888888888888888}

{'kernel': 'rbf', 'C': 1.0, 'gamma': 0.001, 'F1 Score': 0.8263888888888888}

{'kernel': 'rbf', 'C': 5.0, 'gamma': 0.1, 'F1 Score': 0.8888888888888888}

{'kernel': 'rbf', 'C': 5.0, 'gamma': 0.01, 'F1 Score': 0.9097222222222222}

{'kernel': 'rbf', 'C': 5.0, 'gamma': 0.001, 'F1 Score': 0.8888888888888888}

{'kernel': 'rbf', 'C': 14.0, 'gamma': 0.1, 'F1 Score': 0.8888888888888888}

{'kernel': 'rbf', 'C': 14.0, 'gamma': 0.01, 'F1 Score': 0.9097222222222222}

{'kernel': 'rbf', 'C': 14.0, 'gamma': 0.001, 'F1 Score': 0.8958333333333334}

{'kernel': 'poly', 'C': 1.0, 'gamma': 0.1, 'F1 Score': 0.8333333333333334}

{'kernel': 'poly', 'C': 1.0, 'gamma': 0.01, 'F1 Score': 0.5486111111111112}

{'kernel': 'poly', 'C': 1.0, 'gamma': 0.001, 'F1 Score': 0.16666666666666666}

{'kernel': 'poly', 'C': 5.0, 'gamma': 0.1, 'F1 Score': 0.8680555555555556}

{'kernel': 'poly', 'C': 5.0, 'gamma': 0.01, 'F1 Score': 0.5486111111111112}

{'kernel': 'poly', 'C': 5.0, 'gamma': 0.001, 'F1 Score': 0.1736111111111111}

{'kernel': 'poly', 'C': 14.0, 'gamma': 0.1, 'F1 Score': 0.875}

{'kernel': 'poly', 'C': 14.0, 'gamma': 0.01, 'F1 Score': 0.5902777777777778}

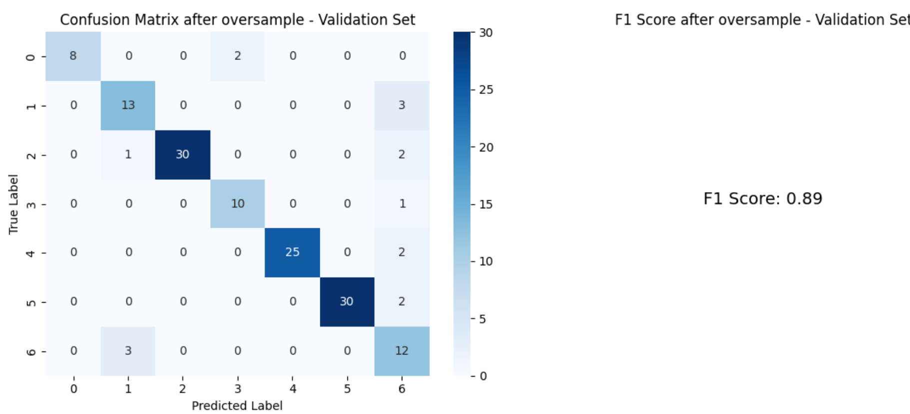
{'kernel': 'poly', 'C': 14.0, 'gamma': 0.001, 'F1 Score': 0.1736111111111111}

F1 Score for Test Set: 0.92

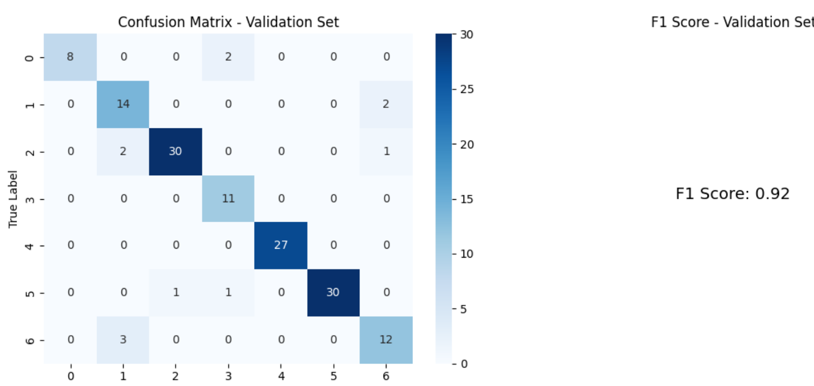
Best Parameters: {'kernel': 'linear', 'C': 14.0, 'gamma': 0.1}

Best F1 Score: 0.9166666666666666

With weighted SVM with class_weights='balanced'



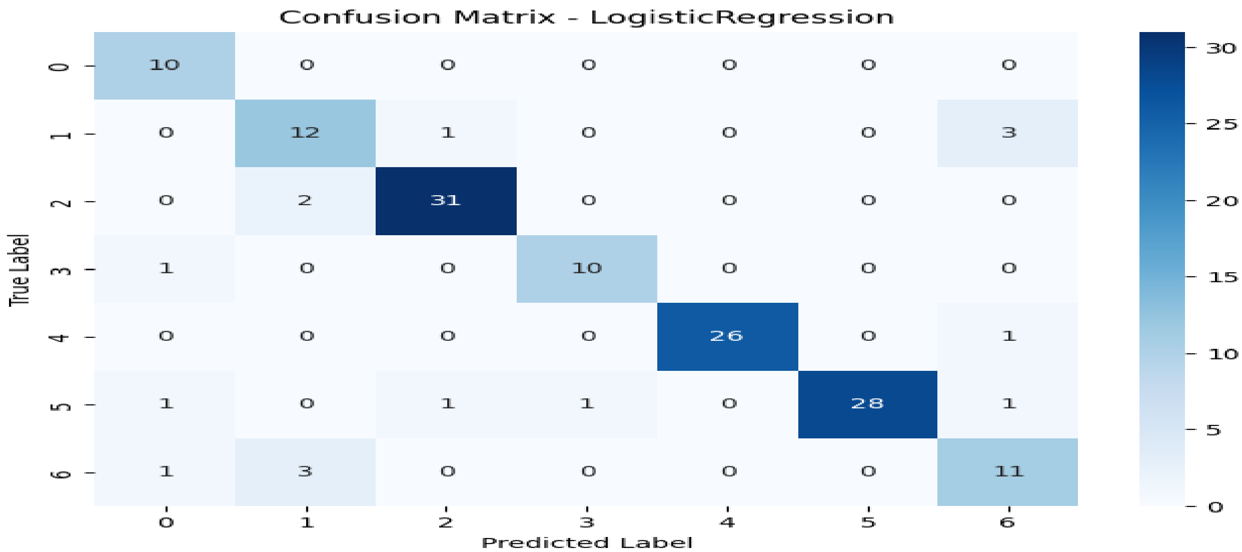

With weighted SVM with calculating inverse of the class frequencies and use them as weights

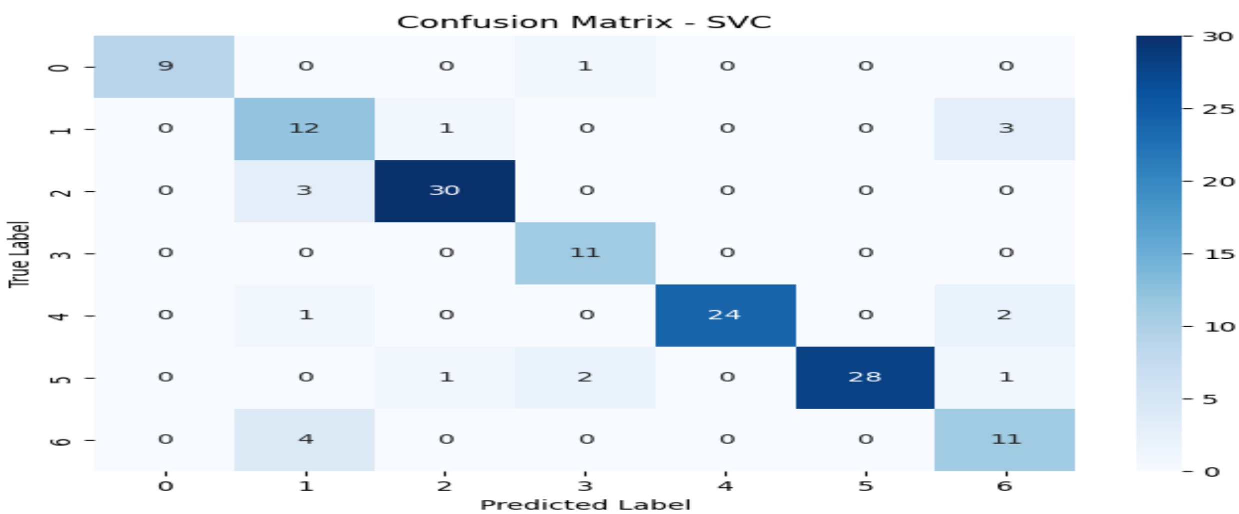


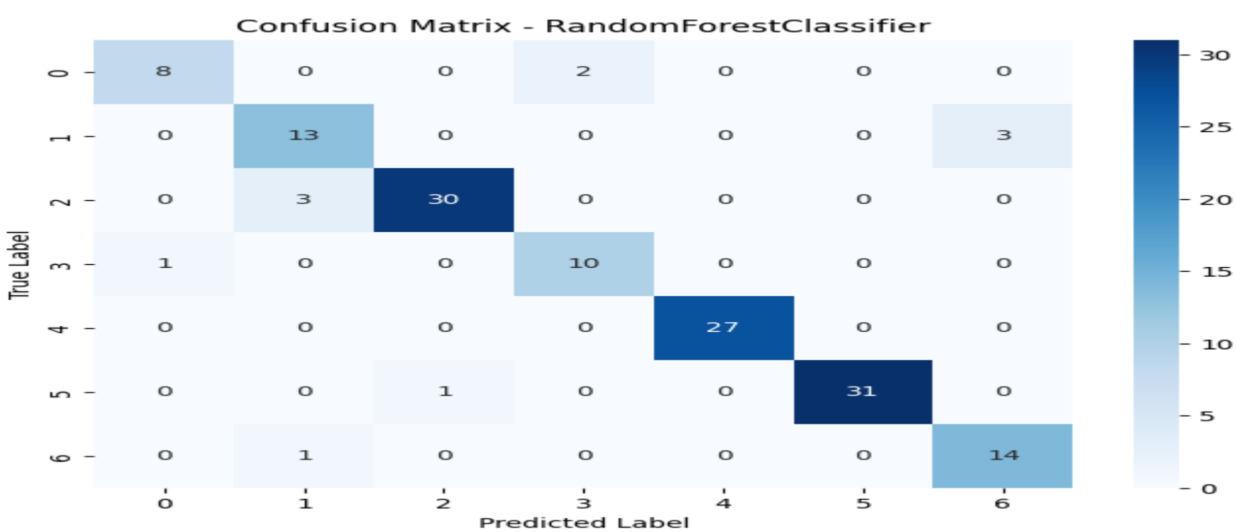

**Cross Validation Scores: [0.92810458 0.90849673 0.91503268 0.88157895 0.92763158]**

**Average Cross Validation Score: 0.9121689026487788**

**Confusion Matrix of all used models:**



**The Recall, Precision, F1_Score of all used models:**