GEBZE TECHNICAL UNIVERSITY

Department of Computer Engineering

CSE 557 | Data Mining | Fall 2023

Project Report

Lecturer : Dr. Burcu Yılmaz

Student : Bülent Karadeniz (225023016010)

Date : 02/02/2024

Project Name

Comparison of Machine Learning Models such as Logistic Regression (LR), Support Vector Classification (SVC), Decision Tree (DT), Random Forest (RF) in the Classification of Raisin Grains.

Content

1. Introduction

a. Aim of the Project :

In this study, a study on the classification of raisins obtained from Besi and Keçimen grapes belonging to Adıyaman region, which is an endemic grape growing in our country, has been utilised in Selçuk University, Faculty of Technology, Department of Computer Engineering. In addition to the machine learning models used in the study, it is aimed to perform the classification process with different models with additional models. My aim is to test the machine learning used in the authors' study on the same data set and to observe different model results by trying different additional methods.

b. Importance and challenges of classifying raisin grains:

Raisins are a concentrated source of carbohydrates and a nutritious snack, containing antioxidants, potassium, fiber and iron [2]. Turkey is one of the countries that ranks top in the world's grape production. Approximately 30% of the grapes produced in Turkey are considered as table, 37% as dried, 3% as wine and 30% as other products [3].

There are many applications of traditional methods for assessing and determining the quality of foods.However, these can be time consuming and expensive. In addition, human-made procedures from traditional methods can be inconsistent and more inefficient, as well as physical conditions such as fatigue and even people's psychological mood can affect the outcome of the work. These negative situations and problems are the main reasons for developing alternative methods to quickly and precisely evaluate the basic features of products such as raisins.

In the process of classification and sorting of this kind of raisins, which is an important agricultural product for our country, perhaps in the future, a more advanced automated system and a process in which machine learning and computer vision models are involved may contribute to the agriculture of our country. The product can be utilised in the best way and contribute to the economy.

c. Previous study and results:

The authors summarise how the dataset was created and the results of some of the classification methods developed. In this study, machine vision system was developed in order to distinguish between two different variety of raisins (Kecimen and Besni) grown in Turkey. Firstly, a total of 900 pieces raisin grains were obtained, from an equal number of both varieties. These images were subjected to various preprocessing steps and 7 morphological feature extraction operations were performed using image processing techniques. In addition, minimum, mean, maximum and standard deviation statistical information was calculated for each feature. The distributions of both raisin varieties on the features were examined and these distributions were shown on the graphs. Later, models were created using LR, MLP, and SVM machine learning techniques and performance measurements were performed. The classification achieved 85.22% with LR, 86.33% with MLP and 86.44% with the highest classification accuracy obtained in the study with SVM. Considering the number of data available, it is possible to say that the study was successful.

2. Dataset

a. Data source :

In this study, the dataset shared by the authors in excel format was used. The dataset can be accessed from this link. (https://archive.ics.uci.edu/dataset/850/raisin) . The method of creating the data set is as follows as mentioned by the authors in the article. Firstly, raisin sample images were obtained and images were processed by using various image processing techniques. The images obtained were first converted to grayscale images and then converted to binary images. Using the imcomplement function on binary images, black areas are converted to white and white areas to black. Later, the images were cleared of noise. In the next phase, various morphological feature inference operations were applied on the obtained images.

b. Data Preprocessing

Briefly, firstly, 450 equal amounts of raisins of Besni Keçimen breeds are photographed in a closed system. Then these images are cleaned from noise by image processing. Then they produce excel data by creating numerical values according to 7 features they determine themselves.

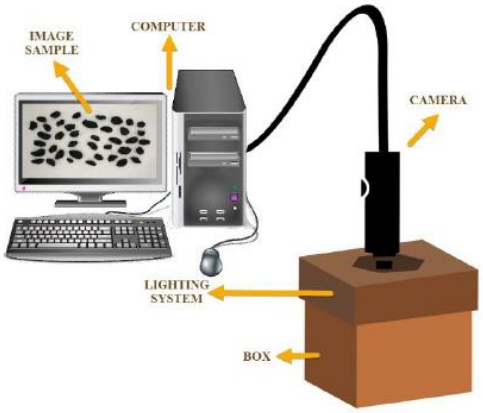
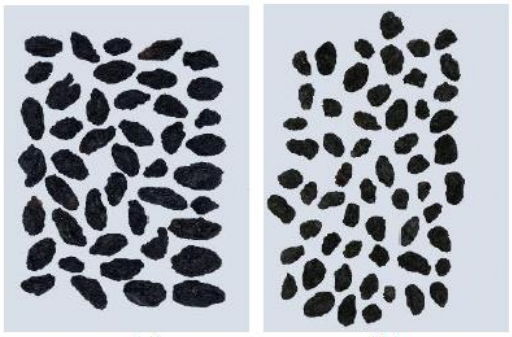


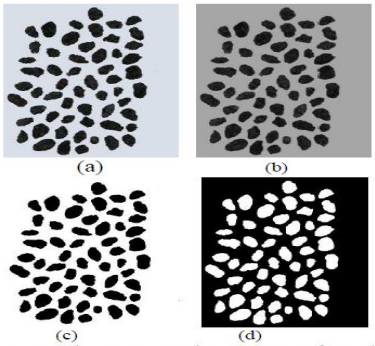
Figure 2. The computer vision system used to acquisition images *(Görüntü elde etmek için kullanılan sistem)*



(a) (b)

Figure 3. Sample image of raisin varieties used in the study ((a) Besni, (b) Kecimen) *(Çalışmada kullanılan kuru üzüm çeşitlerine ait örnek görüntü ((a) Besni, (b) Keçimen))*

In the process of image processing, the pre-processes required for the most accurate way to perform feature extraction and classification are explained. The image processing phase has a critical importance as it directly affects feature inference and thus the classification outcome. All these were taken into account during the processing of the image. Finally, the images are free of the noise on them. The preprocessing steps performed on raisin images are given in Figure 4.



(a) (b) (c) (d)

Figure 4. The preprocessing steps performed on images ((a) Real image, (b) Grayscale image, (c) Binary image, (d) Imcomplement image) *(Görüntüler üzerinde gerçekleştirilen ön işlem aşamaları ((a) Gerçek görüntü, (b) Gri tonlamalı görüntü, (c) İkili görüntü, (d) Terslenmiş görüntü))*

Since the data in the data set was clean and cleaned from noise, no extra work was done. In addition, since the data set is small, no data deletion was applied.

c. Features Ekstraction

At this stage, numerical values are generated from the obtained images according to their pixel values for machine learning. A description of these features is given below.

Area: Gives the number of pixels within the boundaries of the raisin grain.
Perimeter: It measures the environment by calculating the distance between the boundaries of the raisin grain and the pixels around it.

MajorAxisLength: Gives the length of the main axis, which is the longest line that can be drawn on the raisin grain.
MinorAxisLength: Gives the length of the small axis, which is the shortest line that can be drawn on the raisin grain.
Eccentricity: It gives a measure of the eccentricity of the ellipse, which has the same moments as raisins.
ConvexArea: Gives the number of pixels of the smallest convex shell of the region formed by the raisin grain.
Extent: Gives the ratio of the region formed by the raisin grain to the total pixels in the bounding box.

Data set general status sample rows

```
df = pd.read_excel("Raisin_Dataset.xlsx")
```

```
df.head()
```

	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	Extent	Perimeter	Class
0	87524	442.246011	253.291155	0.819738	90546	0.758651	1184.040	Kecimen
1	75166	406.690687	243.032436	0.801805	78789	0.684130	1121.786	Kecimen
2	90856	442.267048	266.328318	0.798354	93717	0.637613	1208.575	Kecimen
3	45928	286.540559	208.760042	0.684989	47336	0.699599	844.162	Kecimen
4	79408	352.190770	290.827533	0.564011	81463	0.792772	1073.251	Kecimen

```
df.info()
```

When we look at the column and row numbers of the data set, it is seen that there are no missing values. There are also no duplicate values. As we have stated above, it is seen that the data set is a cleaned data.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Area             900 non-null    int64
 1   MajorAxisLength  900 non-null    float64
 2   MinorAxisLength  900 non-null    float64
 3   Eccentricity     900 non-null    float64
 4   ConvexArea       900 non-null    int64
 5   Extent           900 non-null    float64
 6   Perimeter        900 non-null    float64
 7   Class            900 non-null    object
dtypes: float64(5), int64(2), object(1)
memory usage: 56.4+ KB
```

In general, when we look at the mean, standard deviation and quartiles of the dataset, we see that the stds in the dataset are smaller than the mean, and that the size between the quartiles is normal.

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Area	900.0	87804.127778	39002.111390	25387.000000	59348.000000	78902.000000	105028.250000	235047.000000
MajorAxisLength	900.0	430.929950	116.035121	225.629541	345.442898	407.803951	494.187014	997.291941
MinorAxisLength	900.0	254.488133	49.988902	143.710872	219.111126	247.848409	279.888575	492.275279
Eccentricity	900.0	0.781542	0.090318	0.348730	0.741766	0.798846	0.842571	0.962124
ConvexArea	900.0	91186.090000	40769.290132	26139.000000	61513.250000	81651.000000	108375.750000	278217.000000
Extent	900.0	0.699508	0.053468	0.379856	0.670869	0.707367	0.734991	0.835455
Perimeter	900.0	1165.906636	273.764315	619.074000	966.410750	1119.509000	1308.389750	2697.753000

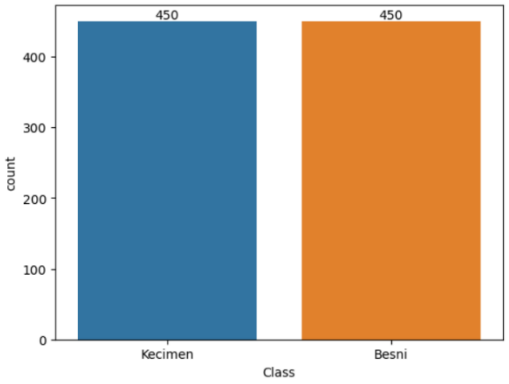
d. Data visualization

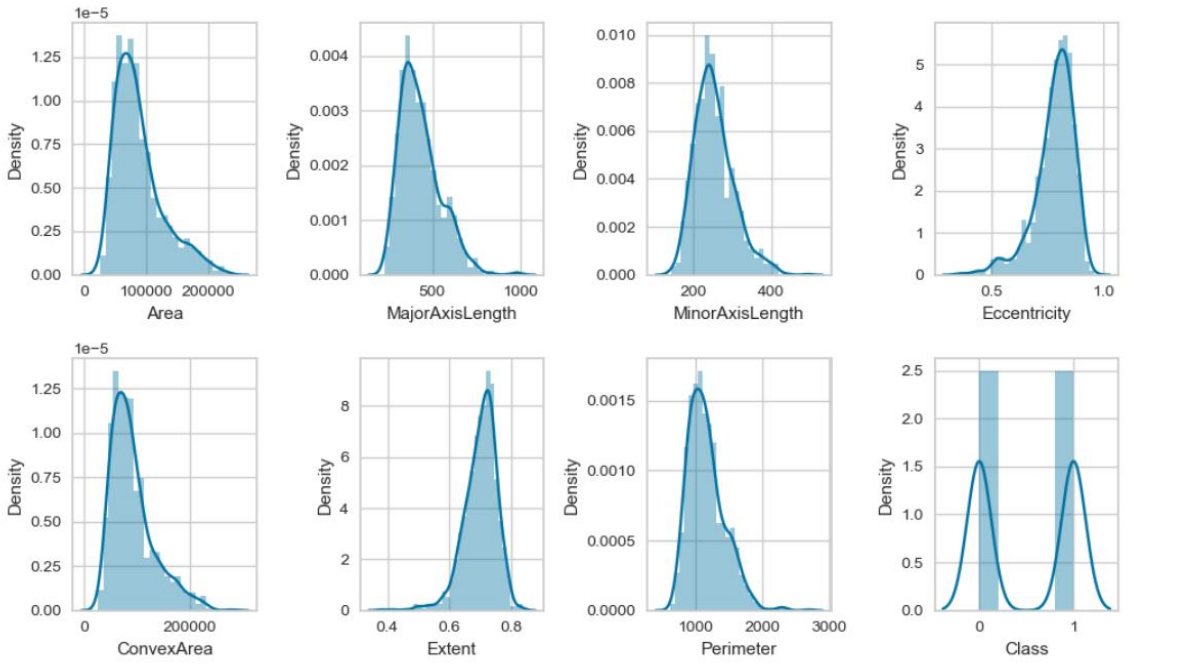
In this graph, we see that the data set is evenly distributed. In addition, in the data set, we assigned the string values in our target feature to the numeric values of the class names. Because we can only give numeric values to the model.

```
# Target değişkenine nümeric dönüşüm gerçekleştirelim
df["Class"] = df["Class"].map({"Kecimen": 0, "Besni": 1})
```

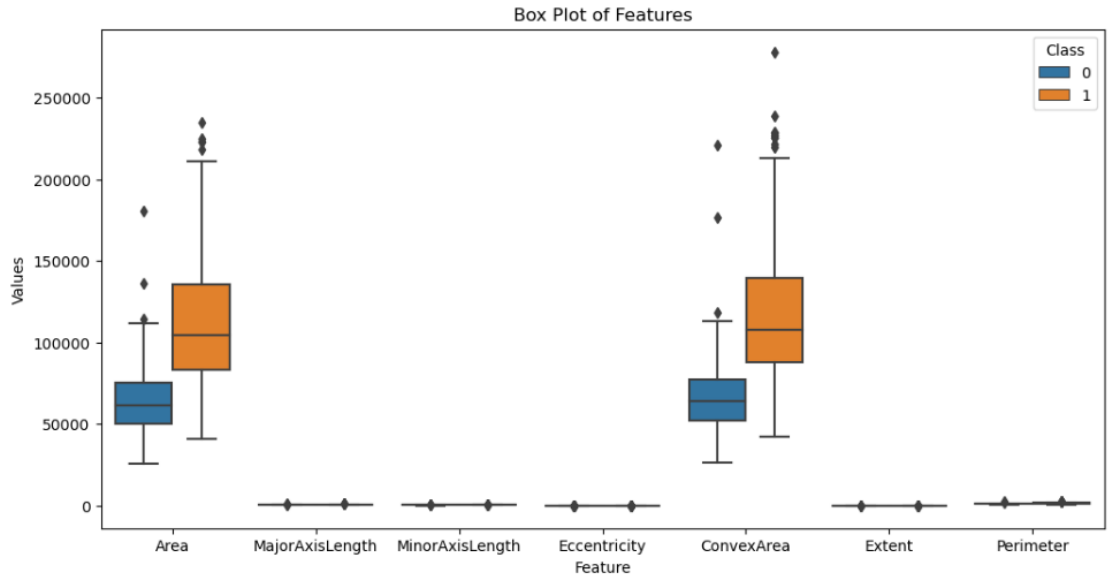
```
df.sample(5)
```

	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	Extent	Perimeter	Class
376	70977	362.063575	256.383025	0.706096	73417	0.650223	1033.870	0
398	82886	424.822709	253.171548	0.803024	85879	0.648281	1163.528	0
96	79661	360.073447	282.739032	0.619209	81032	0.779157	1045.658	0
458	124579	439.960503	371.026214	0.537416	128307	0.698250	1373.537	1
840	48551	302.812835	205.963380	0.733057	50748	0.742824	862.001	1





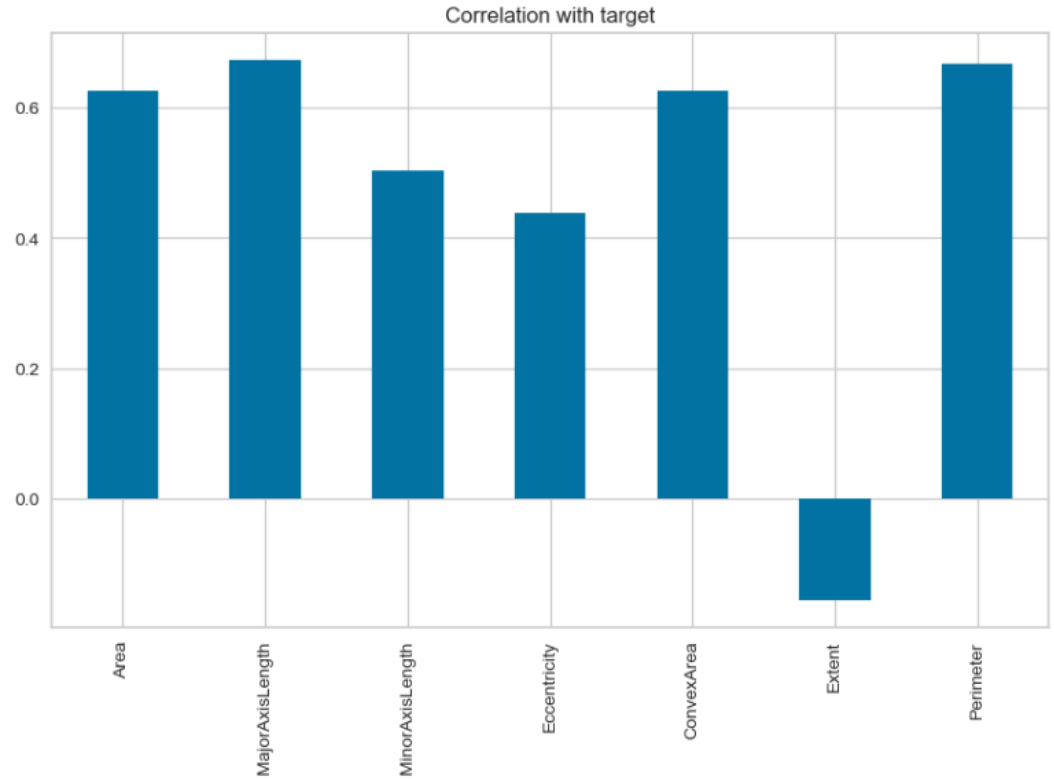
The box plot graph shows some outlier values in the data set, but due to the small size of the data set, no downgrading was performed.



This graph shows which feature is most affected by the target feature and which feature has a positive or negative correlation.

Major axis and perimeter show the highest positive correlation with class feature, while extent feature (this is calculated by a special formula, authors' own)

```
# Correlation with target
df.drop('Class', axis=1).corrwith(df.Class).plot(kind='bar', grid=True, figsize=(10, 6), title="Correlation with target");
```



3. Metrics and Machine Learning Models

a. Metrics

- **Confusing matrix (CM)**

These metrics can be used to evaluate model performance. In this study, since the dataset was balanced, I evaluated the model success mostly on the accuracy value. If the data series were balanced data, it would be more accurate to evaluate it on recall and F1.

In Table 1, the confusion matrix used in the classification of raisins is given. There are four parameters in the confusion matrix. These are named as tp: true positives, fp: false positives, fn: false negatives, tn: true negatives. Examples correctly classified into the positive class are called true positives. Examples correctly classified into the negative class are called true negatives. Examples of positive classes that are falsely classified as negative are called false negatives, and examples of negative classes that are falsely classified as positive are called false positives [11].

The performance metrics for two-class classification are calculated benefiting from Table 1.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$specificity = \frac{TN}{TN + FP}$$

		Predicted		
		Yes	No	
Actual	Yes	2 (True +ve)	1 (False -ve)	2/(2+1)=⅔ Recall (Sensitivity)
	No	2 (False +ve)	3 (True -ve)	3/(3+2)=⅗ (Specificity)
		2/(2+2)=50% (Precision)		Accuracy= (2+3)/(2+1+2+3)= 5/8

Table 1. Confusion matrix used in raisin classification
(Kuru üzüm sınıflandırmasında kullanılan karmaşıklık matrisi)

		Predicted	
		Kecimen	Besni
Actual	Kecimen	tp	fp
	Besni	fn	tn

- **Cross validation (CV)**

Cross-validation is a method of error prediction developed with the aim of improving the security of classification. Cross-validation divides the dataset such that it is random to a determined number of subsets for training and testing. It accepts one of the subsets as a test set, and the system is trained with the remaining sets. This process is repeated up to the number of data sets and the system is tested [13]. Figure 5 shows the cross-validation method.

The number of cross-validation repeats was selected as k=10. 1/10 of the data set is divided for testing and 9/10 for training. For the entire dataset, these operations are repeated and the system test is completed.

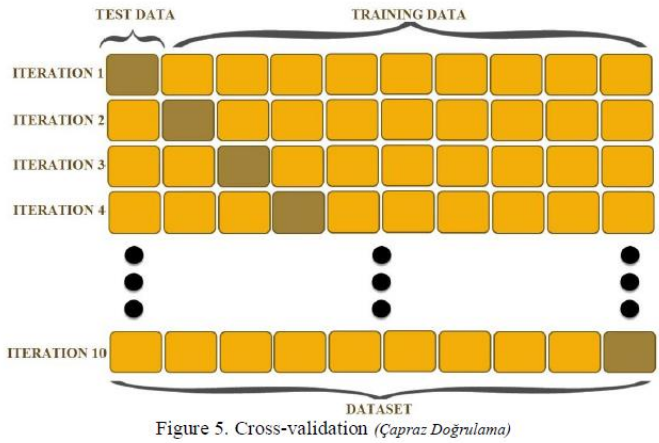


Figure 5. Cross-validation *(Çapraz Doğrulama)*

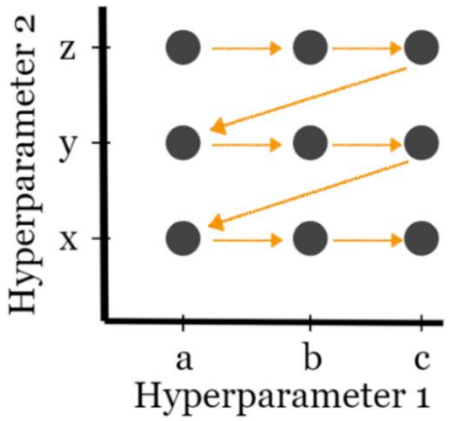
- **Grid Search (GR)**

Grid search is a technique used in machine learning to find the optimal hyperparameters for a model. Hyperparameters are parameters that are not learned during the training process but rather are set beforehand and affect the learning process of the algorithm. Examples of hyperparameters include the learning rate in neural networks or the depth of a decision tree.

In grid search, you specify a grid of hyperparameter values to explore. For example, if you're training a support vector machine (SVM), you might want to try different combinations of C (the regularization parameter) and kernel type (e.g., linear, polynomial, or radial basis function). You define a grid of C values and kernel types, and then grid search trains a model for each combination of hyperparameters and evaluates their performance using cross-validation.

The combination of hyperparameters that yields the best performance on the validation set (or through cross-validation) is then selected as the optimal set of hyperparameters for the model.

I applied grid search, cross validation, confusing matrix on LR model in this study. I used default parameter for other models such as SVM, DT, RF.



- **ROC (Receiver Operating Characteristic) curve and AUC (Area Under the ROC Curve)**

These are evaluation metrics commonly used in binary classification tasks to assess the performance of a classification model.

ROC Curve:

The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier as its discrimination threshold is varied.

It plots the True Positive Rate (Sensitivity) against the False Positive Rate (1 - Specificity) for different threshold values.

The True Positive Rate (TPR) is the ratio of correctly predicted positive instances to all actual positive instances, while the False Positive Rate (FPR) is the ratio of incorrectly predicted positive instances to all actual negative instances.

A diagonal line (the line of no-discrimination) represents the scenario where the classifier performs no better than random guessing, while a perfect classifier would have its ROC curve passing through the upper-left corner (TPR = 1, FPR = 0).

AUC (Area Under the ROC Curve):

The AUC is a single scalar value representing the area under the ROC curve.

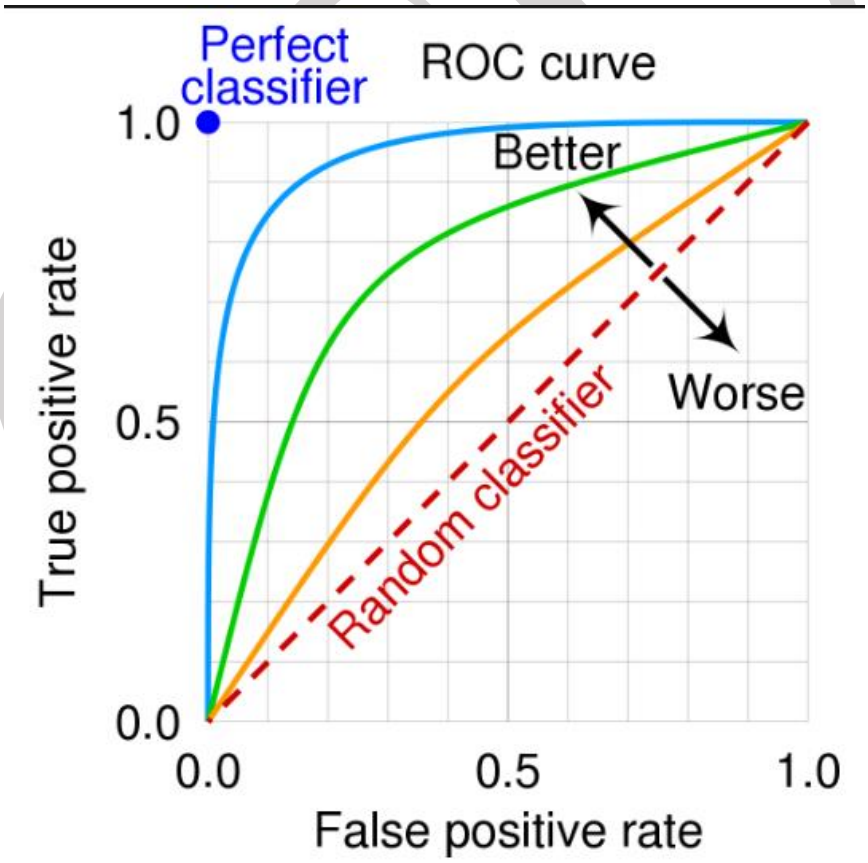
It provides an aggregate measure of the classifier's performance across all possible classification thresholds.

AUC ranges from 0 to 1, where a higher AUC value indicates better classifier performance.

An AUC of 0.5 indicates that the classifier performs no better than random guessing, while an AUC of 1 represents a perfect classifier.

AUC is useful for comparing the overall performance of different models without fixing a specific decision threshold.

In summary, ROC curve and AUC are valuable tools for evaluating the performance of binary classification models, providing insights into their ability to discriminate between positive and negative instances across different threshold settings



b. Importing Libraries

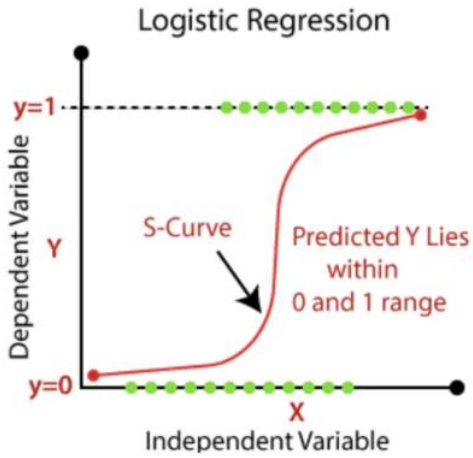
In this study, default models available in Scikit learn libraries were used. Comparison was made with the results of the authors. Since the hyperparameter values used by the authors were not shared in the publication, they were compared with the results given in the publication.

In this study, in all models, the standard scaler process was applied to the entire data set before the data was taken into the model and the target dependent variable was converted into a numerical value by applying the map function.

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.express as px
import cufflinks as cf

from plotly.offline import download_plotlyjs, init_notebook_mode,plot,iplot
init_notebook_mode(connected=True)
cf.go_offline()

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV, cross_validate

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier


from sklearn.metrics import log_loss, recall_score, accuracy_score, precision_score, f1_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import roc_curve, precision_recall_curve, roc_auc_score, auc,average_precision_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


import warnings
warnings.filterwarnings("ignore")
```

c. Logistic Regression (LR)

LR is one of the widely used statistical models. In LR, the dependent variable is estimated from one or more variables. LR clarifies the relationship between dependent variables and independent variables. In LR, variables do not need to require a normal distribution . Because the predicted values in LR are probabilities, they were bounded by 0 and 1. The reason for this is that LR predicts its probability and not itself in the results .

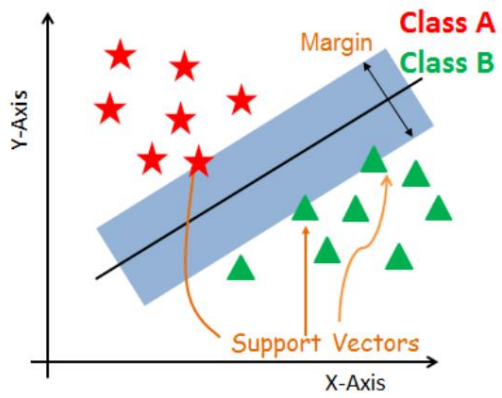


Three models were created for LR. The first model was used for forecasting. The 2nd and 3rd models were only checked for accuracy.

d. Support Vector Machine (SVM)

SVM is a core-based method of forming a hyperplane for classification and regressions. SVM has the ability to classify data as linear in two-dimensional space, planar in three-dimensional space and hyperplane in multi-dimensional space with separation mechanisms. SVM finds the best hyper plane separating the data of the classes and performs the classification process. The best hyperplane for an SVM is the one with the largest margine between the two classes .

I used default parameter for SVM.



e. Decision Tree (DT)

A decision tree is a popular supervised learning algorithm used for both classification and regression tasks in machine learning. It models decisions as a tree structure, where each internal node represents a decision based on a feature, each branch represents the outcome of that decision, and each leaf node represents the final decision or outcome.

Tree Construction: The decision tree algorithm starts with the entire dataset at the root node and recursively splits it into subsets based on the feature that provides the best split (i.e., the feature that best separates the data into classes or minimizes impurity for classification tasks, or maximizes information gain for regression tasks).

Splitting Criteria: The algorithm selects the splitting criteria at each node based on various metrics such as Gini impurity, entropy, or variance reduction, depending on whether it's a classification or regression problem.

Stopping Criteria: The tree-building process continues until a stopping criterion is met, which could be reaching a maximum depth, having a minimum number of samples at a node, or other user-defined conditions.

Prediction: Once the tree is constructed, it can be used to make predictions. For classification, each instance traverses the tree from the root to a leaf node based on the feature values, and the majority class of the instances in that leaf node is assigned as the predicted class. For regression, the predicted value is typically the mean or median of the target values in the leaf node.

Decision trees have several advantages, including their interpretability, ease of understanding, and ability to handle both numerical and categorical data. However, they can be prone to overfitting, especially when the tree grows too large or when the dataset is noisy. Techniques such as pruning and

using ensemble methods like Random Forests or Gradient Boosted Trees are often employed to address these limitations.

f. Random Forest (RF)

Random Forest is a popular ensemble learning technique in machine learning. It's a type of supervised learning algorithm used for classification and regression tasks.

In Random Forest, multiple decision trees are trained during the learning phase. Each tree is trained on a random subset of the training data and considers only a random subset of the features when making a split at each node. This randomness helps to reduce overfitting and makes the model more robust.

During prediction, each tree in the forest independently predicts the target variable, and the final prediction is determined by aggregating the predictions of all the trees. For classification tasks, the final prediction is typically the mode (most frequent class) of all the individual tree predictions, while for regression tasks, it's usually the mean or median of the individual tree predictions.

Random Forest has several advantages, including:

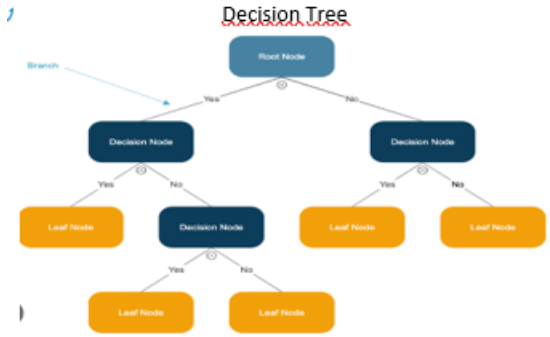
Robustness to overfitting: The randomness in the training process helps to reduce overfitting compared to individual decision trees.

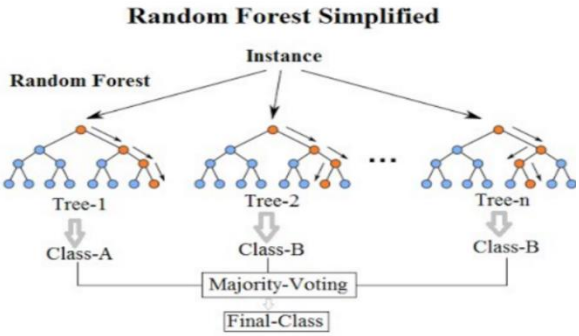
High accuracy: Random Forest often achieves high accuracy on various types of datasets.

Ability to handle large datasets: It can efficiently handle large datasets with high-dimensional feature spaces.

Feature importance: Random Forest can provide insight into the importance of different features in making predictions.

Overall, Random Forest is a versatile and powerful machine learning algorithm that is widely used in practice for a variety of classification and regression tasks.





4. Results

a. Logistic Regression (LR) Results

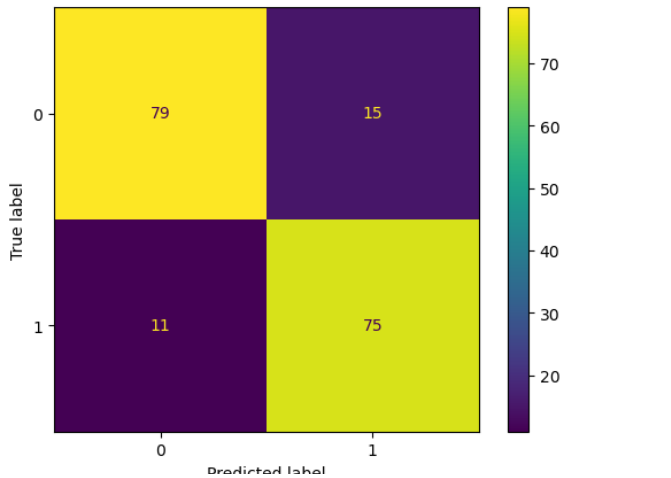
I. LR Default model accuracy score : Train set : 0.85 - Test set : 0.86
II. LR Cross validate method accuracy score: Train set : 0.86
III. LR Grid Search method accuracy score : Train set: 0.87 – Test set : 0.86

Looking at the LR train set and test set scores, we can say that our model is not overfitting. Because train and test results are close to each other.

```
: eval_metric(grid_model, X_train_scaled, y_train, X_test_scaled, y_test)

Test_Set
[[79 15]
 [11 75]]
              precision    recall  f1-score   support

           0       0.88      0.84      0.86        94
           1       0.83      0.87      0.85        86

    accuracy                           0.86       180
   macro avg       0.86      0.86      0.86       180
weighted avg       0.86      0.86      0.86       180


Train_Set
[[317  39]
 [ 55 309]]
              precision    recall  f1-score   support

           0       0.85      0.89      0.87       356
           1       0.89      0.85      0.87       364

    accuracy                           0.87       720
   macro avg       0.87      0.87      0.87       720
weighted avg       0.87      0.87      0.87       720
```

The figure on the right shows the train and test scores of the grid model in aggregate.

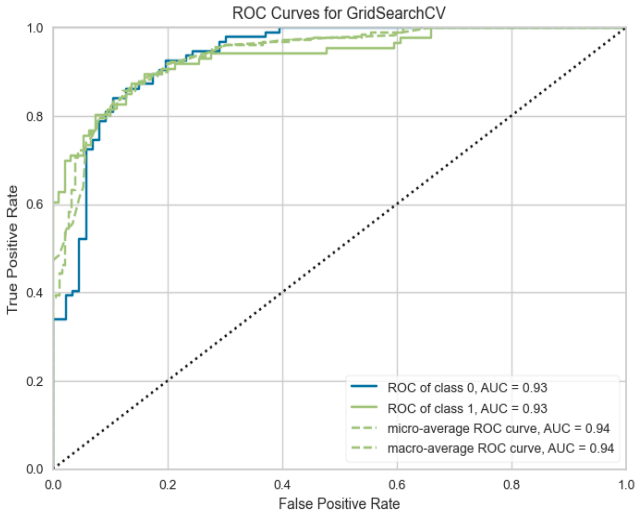
```
disp = ConfusionMatrixDisplay(cm_grid, display_labels=None)
disp.plot()

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2972088b590>
```



The figure on the right shows the CM obtained from our grid model. It is seen that our model correctly predicts 154 grape species on the test data and incorrectly predicts 26 grape species. Here, it is evaluated whether we can get better scores by looking at our model with different parameters.

The graph on the right shows the ROC value of the GridSearch model. As can be seen in the graph, although we obtained high results by grid search, the results did not change much with the hyperparameter adjustments. Maybe more different grid serach parameters may be needed.

In addition, the high AUC value of 0.93 indicates that our model reflects the entire data set.



b. Support Vector Macihine (SVM)

```
eval_metric(SVM_model, X_train_scaled_svm, y_train_svm, X_test_scaled_svm, y_test_svm)
```

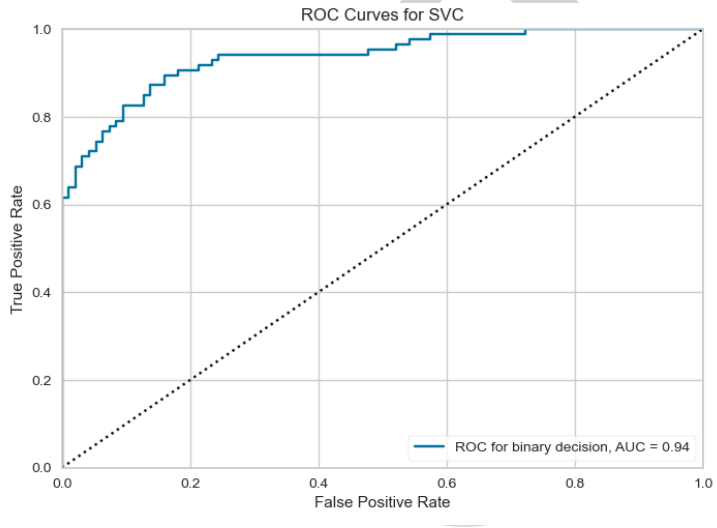
```
Test_Set
[[79 15]
 [11 75]]
              precision    recall  f1-score   support

           0       0.88      0.84      0.86        94
           1       0.83      0.87      0.85        86

    accuracy                           0.86       180
   macro avg       0.86      0.86      0.86       180
weighted avg       0.86      0.86      0.86       180


Train_Set
[[317  39]
 [ 55 309]]
              precision    recall  f1-score   support

           0       0.85      0.89      0.87       356
           1       0.89      0.85      0.87       364

    accuracy                           0.87       720
   macro avg       0.87      0.87      0.87       720
weighted avg       0.87      0.87      0.87       720
```

The figure on the right shows the train and test scores of the SVM model in aggregate.

Train accuracy score : 0.87

Test accuracy score : 0.85

It is seen.



We can see ROC graph and AUC score. Model performance is seen high.

c. Decision Tree Model (DT)

```
eval_metric(DT_model, X_train_scaled_dt, y_train_dt, X_test_scaled_dt, y_test_dt)
```

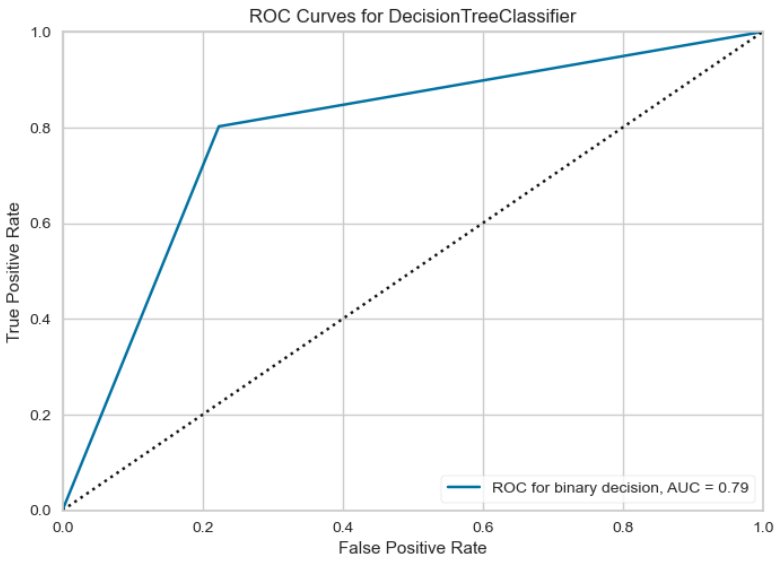
```
Test_Set
[[80 14]
 [11 75]]
              precision    recall  f1-score   support

           0       0.88      0.85      0.86        94
           1       0.84      0.87      0.86        86

    accuracy                           0.86       180
   macro avg       0.86      0.86      0.86       180
weighted avg       0.86      0.86      0.86       180


Train_Set
[[326  30]
 [ 59 305]]
              precision    recall  f1-score   support

           0       0.85      0.92      0.88       356
           1       0.91      0.84      0.87       364

    accuracy                           0.88       720
   macro avg       0.88      0.88      0.88       720
weighted avg       0.88      0.88      0.88       720
```

The figure on the right shows the train and test scores of the DT model in aggregate.

Train accuracy score : 0.88

Test accuracy score : 0.86

It is seen.



We can see ROC graph and AUC score. Model performance is seen low, The curve in the graph is not the curve we expect. It is necessary to investigate why the result is like this. It is possible that models running on the same jupyter notebook may have common variable values.

d. Random Forest Model (RF)

```
eval_metric(RF_model, X_train_scaled_rf, y_train_rf, X_test_scaled_rf, y_test_rf)
```

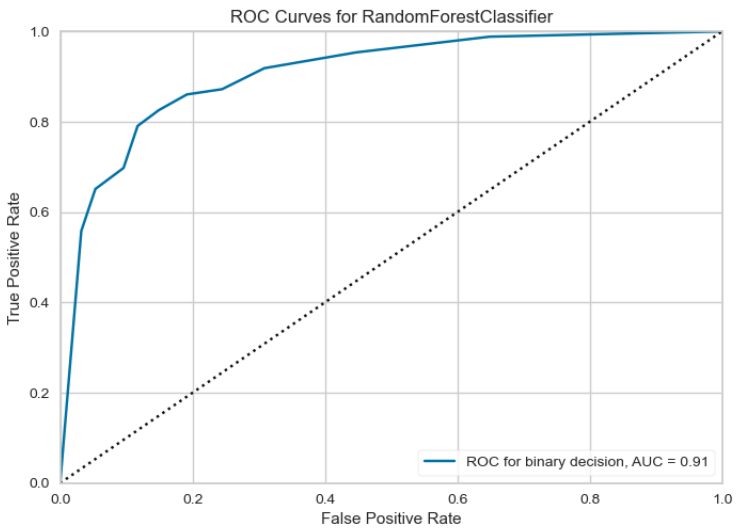
```
Test_Set
[[73 21]
 [17 69]]
              precision    recall  f1-score   support

           0       0.81      0.78      0.79        94
           1       0.77      0.80      0.78        86

    accuracy                           0.79       180
   macro avg       0.79      0.79      0.79       180
weighted avg       0.79      0.79      0.79       180


Train_Set
[[356   0]
 [  0 364]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       356
           1       1.00      1.00      1.00       364

    accuracy                           1.00       720
   macro avg       1.00      1.00      1.00       720
weighted avg       1.00      1.00      1.00       720
```

The figure on the right shows the train and test scores of the RF model in aggregate.

Train accuracy score : 1.00

Test accuracy score : 0.79

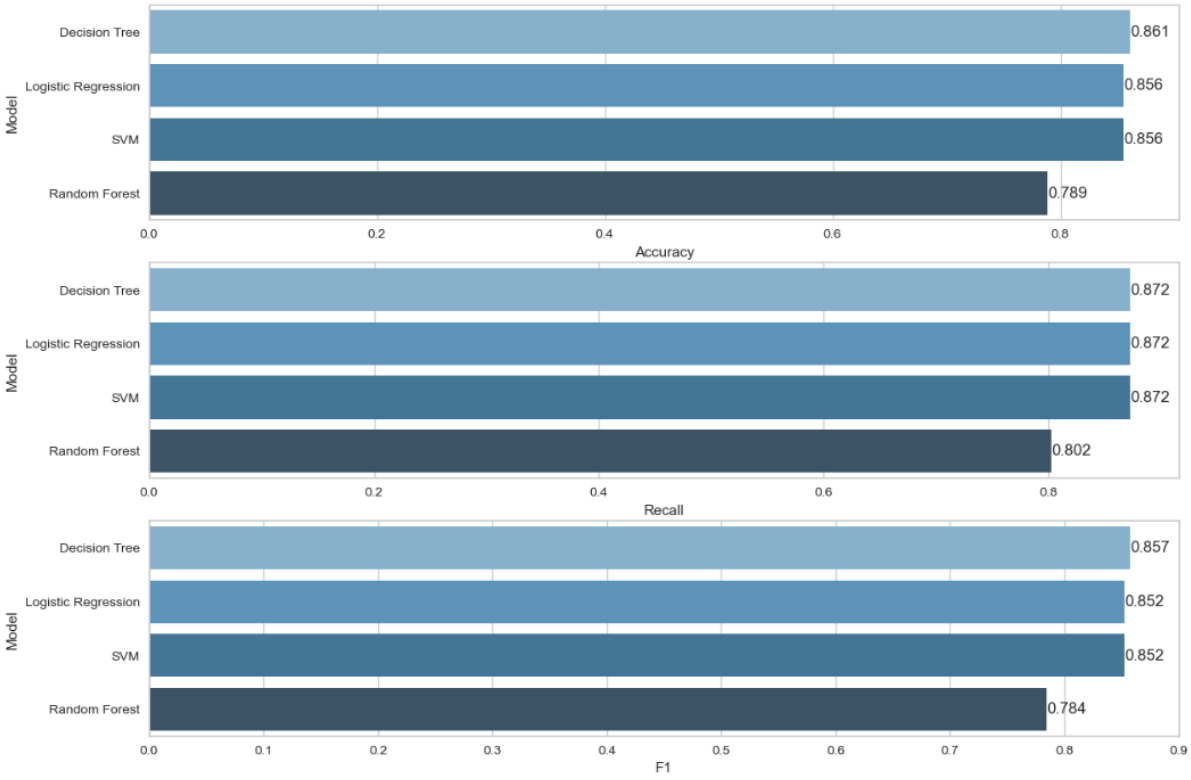
This model was seen overfit. Because while train score is high, test score is low. While the model did not make any mistakes in the train, it made many mistakes in the test set, this shows that the model has memorised the train set. However, although we know that the RF model solves the overfitting in DTs randomly, the fact that the results come in this way again suggests variable overlap. Or kernel resetting is required.



We can see ROC graph and AUC score. Model performance is seen high,

e. Comparison of classification performance of each model together and authors

We can see results of the my models in my study.



Results of the Author's Study in Table 5, Table 6.

Table 5. The confusion matrix data belonging to the algorithms *(Algoritmalara ait karmaşıklık matrisi verileri)*

Algorithms LR, MLP, SVM		**Predicted**	
		Kecimen	Besni
Actual	Kecimen	391 400 404	59 50 46
	Besni	74 73 76	376 377 374

Table 6. Classification performance measurement results *(Sınıflandırma performans ölçüm sonuçları)*

Performance Measure	LR	MLP	SVM
Accuracy	85.22	86.33	86.44
Sensivitiy	84.09	84.57	84.17
Specificity	86.44	88.29	89.05
Precision	86.89	88.89	89.78
F1-Score	85.46	86.67	86.88
Negative Predictive Value	83.56	83.78	83.11
False Positive Rate	13.56	11.71	10.95
False Discovery Rate	13.11	11.11	10.22
False Negative Rate	15.91	15.43	15.83

5. Discussion & Conlusion

According to the results of the authors in the publication that constitutes the origin of this project, the best model performance (accuracy value) is obtained with the SVM model, while the accuracy values of the LR and MLP (deep learning model) models are close.

In my study, if we look at the accuracy scores of the models I created, it is seen that the best performance is obtained in the DT model, while the score of the RF model, which I expect more success, is low. In addition, it is seen that my LR and SVM models are close to the scores described in the publication.

I believe that the reason for the low score of the RF model is the conflicting variable names and the kernell error that is not restarted when each model is running. I realised the importance of running models on separate notebooks.

In addition, the author does not explain why only these three methods are used in the article. In my opinion, a pioneering study can be carried out by trying DT and RF models in such a study and by adjusting the parameters and hyperparameters. And by increasing the data set a little more, it can be inferred that the accuracy scores of 0.85 can be increased.

6. References

1. CINAR I., KOKLU M. and TASDEMIR S., (2020). "Classification of Raisin Grains Using Machine Vision and Artificial Intelligence Methods", Gazi Journal of Engineering Sciences, vol. 6, no. 3, pp. 200-209, December, 2020, DOI: https://doi.org/10.30855/gmbd.2020.03.03
2. Karimi,N.,Arabhosseini,A.,Kianmehr,M.H.,andKhazaei,J.,"Modellingofraisin berriesbysomephysicalandstatisticalcharacteristics,"*Int.Agrophys*.,vol.25(2),pp.141-147,April2011
3. Semerci,A.,Kızıltuğ,T.,Çelik,A.,andKiracı,M.,"Türkiyebağcılığının geneldurumu" *MustafaKemalÜniversitesiZiraatFakültesiDergisi*,vol.20(2),pp.42-51,October2015.