Classification of Raisin Varieties Using Machine Learning: A Comparative Model Analysis

Word count: 2,198

Python Script: <https://github.com/Aghedo67/Machine_learning_coursework>

# INTRODUCTION

Raisins are dried grapes enjoyed around the world, whether eaten raw or used in cooking and baking. In places like the UK, Australia, and South Africa, "raisin" refers to dark dried grapes, "sultana" to golden ones, and "currant" to small black seedless grapes. This project applies supervised machine learning to classify two raisin types Kecimen and Besni using features like area, perimeter, and eccentricity. We explore models including Logistic Regression, Random Forest, Decision Tree, Gradient Boosting, and Support Vector Machine. Each is evaluated on accuracy, precision, recall, and interpretability to find the most effective approach for this classification task.

# Data Summary

This study uses a dataset from Kaggle, originally developed by Çınar, Köklü, and Taşdemir (2020) for research on classifying raisin varieties with machine vision. It includes detailed morphological features of 900 raisin grains equally split between Turkey’s Kecimen and Besni types extracted through image processing. Though I found the dataset on Kaggle, it was created for academic use and offers a strong foundation for applying machine learning to distinguish between these two popular raisin varieties.

**Figure 1** Comparison of raisins (left) and currants (right)

Source: Wikipedia, 2025.



Area: The area measures the size showing how big

MajorAxisLength**:** This length captures the longest axis.

MinorAxisLength: This captures the shortest.

Eccentricity**:** This shows the round or stretch of the raisin from 0 (round) to 1 (long).

ConvexArea: This is the area of the smallest shape the fully covers the raisin which is useful for spotting shape irregularities.

Extent**:** This shows the amount of the bounding box the raisin fills, and it is shown as ratio.

Perimeter: The distance around the boundary of the raisin.

Class: The categorical target variable, denoting the raisin variety ('Kecimen' or 'Besni').

Exploratory data analysis (EDA) of the dataset was conducted.

Our initial examination of the dataset revealed several key characteristics. Notably, the dataset exhibits a perfect balance in the 'Class' distribution, with 450 instances of each variety. This balanced target variable simplifies model training and evaluation, allowing for a more direct interpretation of performance metrics. Furthermore, we found no missing values across any of the eight features, streamlining the initial data preprocessing.

**Figure 2**

A chart of a class distribution

AI-generated content may be incorrect.

Table Summary Statistics of the data set

|  | **Count** | **Mean** | **Std Dev** | **Min** | **25%** | **Median** | **75%** | **Max** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Area | 900 | 87,804.13 | 39,002.11 | 25,387.00 | 59,348.00 | 78,902.00 | 105,028.25 | 235,047.00 |
| Major Axis Length | 900 | 430.93 | 116.04 | 225.63 | 345.44 | 407.80 | 494.19 | 997.29 |
| Minor Axis Length | 900 | 254.49 | 49.99 | 143.71 | 219.11 | 247.85 | 279.89 | 492.28 |
| Eccentricity | 900 | 0.7815 | 0.0903 | 0.3487 | 0.7418 | 0.7988 | 0.8426 | 0.9621 |
| Convex Area | 900 | 91,186.09 | 40,769.29 | 26,139.00 | 61,513.25 | 81,651.00 | 108,375.75 | 278,217.00 |
| Extent | 900 | 0.6995 | 0.0535 | 0.3799 | 0.6709 | 0.7074 | 0.7350 | 0.8355 |
| Perimeter | 900 | 1,165.91 | 273.76 | 619.07 | 966.41 | 1,119.51 | 1,308.39 | 2,697.75 |

The descriptive statistics of the numerical features indicated a considerable range and variability in the morphological measurements. This suggests inherent physical differences between the raisin samples, which our machine learning models will aim to capture to perform accurate classification. The continuous nature of most numerical features, evidenced by the high number of unique values, informs the selection of appropriate algorithms

**Figure 3**

|  |
| --- |
| A graph of two people  AI-generated content may be incorrect.  A graph and diagram of a graph  AI-generated content may be incorrect. |
| A graph and a chart  AI-generated content may be incorrect.  A graph of a function  AI-generated content may be incorrect.  A graph and a diagram  AI-generated content may be incorrect.  A close-up of a graph  AI-generated content may be incorrect.A graph and diagram of a graph  AI-generated content may be incorrect. |

Given the skewed distributions in several features, we applied a logarithmic transformation to make them more symmetrical. This can improve linear model performance by making relationships more linear and reducing the impact of extreme values. We used np.log1p(), which handles both skewness and zeros well. It also helped compress large values, making outliers less influential. Applying this transformation across all features ensured a more balanced dataset, preventing bias toward heavily skewed variables as illustrated in Figure 2.

Logarithmic transformation is a common way to reduce positive skew in data (Field, 2009; Tukey, 1977). Right-skewed distributions where most values are low but a few are very high can affect the assumptions behind many statistical models and hurt the performance of algorithms, especially linear ones. Using a log function helps by compressing the larger values more than the smaller ones, making the data more symmetrical and closer to a normal distribution. This can lead to better model fit and more reliable results.

Logarithmic transformation can sometimes help straighten out non-linear relationships between features and the target variable. While we didn’t specifically check scatter plots against the ‘Class’ label, making these relationships more linear can still boost the performance of linear models. Compared to square root transformation, the log approach tends to reduce skewness more effectively, especially when the data is heavily skewed (Osborne, 2010). Ultimately, the best choice depends on how skewed each feature is, but logs often offer a stronger correction in such cases.

**Figure 4**

A red and blue squares with white text

AI-generated content may be incorrect.

The correlation analysis between the numerical features provided valuable insights into their relationships. Strong positive correlations, such as that between 'Area' and 'ConvexArea', are logically consistent with our understanding of physical objects.

Figure Pairwise Feature relationship

A graph of different colored dots

AI-generated content may be incorrect.

The classes were visually separable, especially using features like Eccentricity and Extent. Since most features were right-skewed, we first applied a log transformation to reduce skewness and make the data more linear. We then tackled feature redundancy by removing highly correlated features. Using a function with a 0.99 threshold, we found that ConvexArea was strongly correlated with others like Area and Perimeter, so it was removed. This step helps avoid confusion in the model about which feature matters most, making it more stable, easier to interpret, and less prone to overfitting.

# Model Pipeline

|  |  |
| --- | --- |
|  |  |
| Load & Understand Data | Grab the raisin data, quickly see what's what (features, classes, no missing stuff). |
|  |  |
| Visualize Data | Plot the features to see their spread and how the two raisin types look different (saw some skew!). PCA gave a rough idea of separation. |
|  |  |
| Data Preprocessing | Tried a log transform on skewed features. Definitely scale everything so the models play fair |
|  |  |
| Train & Cross-Validate | Run Logistic Regression, SVM, Random Forest, Gradient Boosting, and Decision Tree using 5-fold cross-validation to see which performs best on average. |
|  |  |
| Tune Top Models | Tweak the settings (hyperparameters) of the best couple of models (looks like SVM and Gradient Boosting are promising) using more cross-validation to squeeze out better performance |
|  |  |
| Pick Final Model | Choose the champion based on the tuned cross-validation scores. |
|  |  |
| Final Test | Check how the final model does on the untouched test data for a real-world feel. |
|  |  |
| Report | Show what features mattered (if using trees) and how well the model did overall |
|  |  |

# problem definition

The problem is to accurately classify raisin grains into two distinct varieties, Kecimen and Besni, based on their morphological features extracted through image processing. Using a dataset of 900 raisin samples with eight quantitative attributes (Area, MajorAxisLength, etc.), the goal is to develop and evaluate machine learning models capable of reliably distinguishing between these two commercially important raisin types. This classification task has potential applications in agricultural quality control and food processing.

## Algorithm selection

### Random Forest: is an ensemble method that builds many decision trees using random parts of the data and features, then averages their results. This helps prevent overfitting and works well with both simple and complex patterns. It also shows which features matter most. Given the possible complex interactions between raisin features, Random Forest is a strong choice for accurate predictions..

### Gradient Boosting: it builds trees one by one, with each new tree learning from the errors of the last. Libraries like XGBoost and LightGBM are industry favorites for classification. They excel at spotting subtle patterns in raisin morphology, often boosting accuracy.

### Logistic Regression: is a simple and easy-to-understand model, great for binary classification and setting a baseline. It predicts the chance of a raisin being Kecimen or Besni using a formula that combines the features. Since our features are continuous, it works well for finding linear relationships and shows which features matter most through its coefficients.

### Support vector machine(SVM): SVM is a strong algorithm that can handle both simple and complex boundaries using different kernels. It finds the best dividing line between classes to separate them clearly. Because raisin features may have complex patterns, using a non-linear kernel like RBF helps SVM capture these well, often leading to high accuracy. It works great in high-dimensional data and usually generalizes well, making it a solid choice here.

### Decisiom Tree: is a simple, easy-to-understand model that uses if-then rules to classify raisins. While it can overfit with complex data, it’s great for a baseline and shows which features and thresholds matter most. Its clear structure makes it easy to see how decisions are made, giving helpful insights even if it’s less accurate than more advanced models.

## Model Evaluation

From Tabel 2 we can see that Logistic Regression (LR) achieved an accuracy of 85.6%. It shows a slightly better recall for 'Besni' but better precision for 'Kecimen', RF performed slightly better in terms of overall accuracy 86.1% compared to LR in this split. It shows a balanced precision and recall for both classes, DT Has the lowest accuracy 79.4% among the models, indicating it might be overfitting to the training data or is not capturing the underlying patterns as well as the others. GB Achieved an accuracy of 85%, like Logistic Regression and SVM in this split. It shows a slightly better recall for 'Kecimen'. Support Vector Machine (SVM) also achieved an accuracy of 85%. It has the highest recall for 'Kecimen' but the lowest recall for 'Besni' among the top performers. I explored all 5 models to understand these different approaches and allow for a comparative analysis

which is often a strong point in building a machine learning model.

**Table 2** Classification performance measurement

| Table Head |  |  |  | Performance metrics | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Models | Accuracy | precision | | recall | f1-score | support |
|  | LR | 0.855 | 0.83 | | 0.87 | 0.85 | 86 |
|  | RF | 0.861 | 0.86 | | 0.85 | 0.85 | 86 |
| Besni | DT | 0.794 | 0.78 | | 0.79 | 0.79 | 86 |
|  | GB | 0.85 | 0.85 | | 0.84 | 0.84 | 86 |
|  | SVM | 0.85 | 0.86 | | 0.81 | 0.84 | 86 |
|  | LR |  | 0.88 | | 0.84 | 0.86 | 94 |
|  | RF |  | 0.86 | | 0.87 | 0.87 | 94 |
| Kecimen | DT |  | 0.81 | | 0.80 | 0.80 | 94 |
|  | GB |  | 0.85 | | 0.86 | 0.86 | 94 |
|  | SVM |  | 0.84 | | 0.88 | 0.86 | 94 |

## Algorithm not selected

Naive Bayes: NB classifiers assume feature independence, meaning the presence of one feature doesn’t impact the others. However, in our raisin morphology data (e.g., Area, MajorAxisLength, MinorAxisLength), this assumption is likely violated, as many features are correlated (e.g., Area and ConvexArea, dimensions and Perimeter). When features depend on each other, Naive Bayes can give unreliable probability estimates, leading to suboptimal performance compared to models that account for feature dependencies.

K-Nearest Neighbors (KNN): KNN is a non-parametric, instance-based algorithm that classifies data based on the majority class of its k-nearest neighbors. While simple and effective, its performance can suffer from the curse of dimensionality, especially with many features (though we have a moderate amount). KNN also depends on choosing the right distance metric and value for ‘k’. Without careful tuning and feature scaling, it may struggle to capture complex, non-linear boundaries and doesn’t provide insights into feature importance.

## Cross Validation

A cross validation was also carried out to provide a much more robust picture of how our chosen models are likely to perform on unseen raisin data compared to a single train-test split. Furthermore, Cross-validation was used to assess model performance in a reliable and unbiased manner. A 5-fold cross-validation technique was employed, which partitions the data into five equal subsets. Each model was trained on four folds and tested on the remaining fold, repeated five times. This approach minimizes the risk of overfitting, provides a more accurate estimate of generalization performance, and enables fair comparison across models (Kohavi, 1995; Refaeilzadeh, Tang and Liu, 2009).

**Table 3** Cross-Validation Mean Accuracy and Standard Deviation for Classification Models

|  |  |  |
| --- | --- | --- |
| Model | Mean Accuracy | Standard Deviation |
| LR | 0.8611 | 0.0304 |
| SVM | 0.8750 | 0.0311 |
| RF | 0.8569 | 0.0379 |
| GB | 0.8708 | 0.0235 |
| DT | 0.8347 | 0.0203 |

To evaluate the classification performance, five models were assessed using 5-fold cross-validation. Support Vector Machine (SVM) yielded the highest mean accuracy (87.5%) with reasonable consistency (Std = 0.031). Gradient Boosting followed closely (87.1%) and showed the least variation across folds (Std = 0.023), indicating strong stability. Logistic Regression and Random Forest also performed competitively with accuracies of 86.1% and 85.7%, respectively. The Decision Tree model lagged with 83.5% accuracy, likely due to its sensitivity to training data and tendency to overfit. These findings align with existing literature that supports the effectiveness of SVM and ensemble methods in classification tasks with well-structured features (Cortes & Vapnik, 1995; Breiman, 2001; Friedman, 2001).

## PCA (Principal Component Analysis)

**Figure 6** Principal Component Analysis

A graph with red and blue dots

AI-generated content may be incorrect.

Principal Component Analysis (PCA) was applied to project the high-dimensional feature space into a 2D plane. The resulting scatter plot (Figure 5) shows a clear, though not perfect, separation between the two raisin classes (Besni and Kecimen). This visualization supports the idea that the selected features possess discriminative power and validates the potential for effective classification. PCA also confirms that much of the variance in the dataset can be captured using a few components, reducing dimensionality without significant loss of information.

## Receiver Operating Characteristic (ROC) curve

It provides an excellent way to evaluate the performance of a classification model especially when dealing with a binary classification problem

A graph with a line

AI-generated content may be incorrect.

The performance of the classification model was further evaluated using the Receiver Operating Characteristic (ROC) curve. The ROC curve plots the true positive rate against the false positive rate at various threshold levels. The Area Under the Curve (AUC) was found to be 0.93, indicating excellent model performance and a high ability to distinguish between the Besni and Kecimen raisin classes. We need this so that we can evaluate the model discrimination power and to also visualize tradeoffs between sensitivity and specificity, even in balanced settings, ROC-AUC adds extra assurance by evaluating performance across thresholds and is still considered a more comprehensive metric.

##### Discussion and Conclusion

This study successfully applied machine learning to classify raisin varieties, with Support Vector Machines and Gradient Boosting emerging as the top-performing models during cross-validation. Logistic Regression and Random Forest also showed strong results, making them solid alternatives. However, the relatively small dataset and limited number of features may limit how well these models perform in broader, real-world scenarios. Future research could benefit from using larger and more diverse datasets, incorporating additional morphological features, and placing more focus on model interpretability. Further fine-tuning of model parameters may also help improve accuracy and robustness, making these models even more useful for applications like agricultural quality control.

##### References

1. Osborne, J., 2010. Improving your data transformations: applying the Box-Cox transformation. *Practical Assessment, Research, and Evaluation*, 15(1), p.12. Available at: <https://doi.org/10.7275/qbpc-gk17> [Accessed 17 May 2025].
2. Field, A., 2009. *Discovering statistics using SPSS*. 3rd ed. London: Sage [Accessed 17 May 2025].
3. Tukey, J. W. (1977). *Exploratory data analysis*. Addison-Wesley [Accessed 17 May 2025]..
4. Cinar, I., Koklu, M. and Tasdemir, S., 2020. *Classification of raisin grains using machine vision and artificial intelligence methods*. Gazi Journal of Engineering Sciences, 6(3), pp.200–209. Available at: <https://doi.org/10.30855/gmbd.2020.03.03> [Accessed 17 May 2025].
5. Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd ed.). John Wiley & Sons [Accessed 17 May 2025].
6. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989 [Accessed 17 May 2025].
7. Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.
8. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, *29*(5), 1189-1232 [Accessed 17 May 2025].
9. Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine learning*, *29*(2-3), 103-130 [Accessed 17 May 2025].
10. Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is “nearest neighbor” meaningful?. In *International conference on database theory* (pp. 217-235). Springer, Berlin, Heidelberg [Accessed 17 May 2025].
11. Breiman, L. (2001) ‘Random forests’, *Machine Learning*, 45(1), pp. 5–32. Available at: <https://doi.org/10.1023/A:1010933404324> [Accessed 17 May 2025].
12. Cortes, C. and Vapnik, V. (1995) ‘Support-vector networks’, *Machine Learning*, 20(3), pp. 273–297. Available at: <https://doi.org/10.1007/BF00994018> [Accessed 17 May 2025].
13. Friedman, J.H. (2001) ‘Greedy function approximation: A gradient boosting machine’, *Annals of Statistics*, 29(5), pp. 1189–1232.
14. Field, A. (2009) *Discovering statistics using SPSS*. 3rd edn. London: Sage [Accessed 17 May 2025].
15. Kohavi, R. (1995) ‘A study of cross-validation and bootstrap for accuracy estimation and model selection’, *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 1137–1143[Accessed 17 May 2025].
16. Refaeilzadeh, P., Tang, L. and Liu, H. (2009) ‘Cross-validation’, in LIU, L. and ÖZSU, M.T. (eds.) *Encyclopedia of database systems*. Boston: Springer, pp. 532–538. Available at: <https://doi.org/10.1007/978-0-387-39940-9_565> [Accessed 19 May 2025].
17. Fawcett, T. (2006) ‘An introduction to ROC analysis’, *Pattern Recognition Letters*, 27(8), pp. 861–874. <https://doi.org/10.1016/j.patrec.2005.10.010> [Accessed 19 May 2025].
18. Fawcett, T. (2006). *An introduction to ROC analysis*. *Pattern Recognition Letters*, 27(8), pp. 861–874. <https://doi.org/10.1016/j.patrec.2005.10.010>
19. Bradley, A.P. (1997). *The use of the area under the ROC curve in the evaluation of machine learning algorithms*. *Pattern Recognition*, 30(7), pp. 1145–1159 [Accessed 19 May 2025].
20. Jolliffe, I.T. & Cadima, J. (2016). *Principal component analysis: a review and recent developments*. Philosophical Transactions of the Royal Society A, 374(2065), 20150202. <https://doi.org/10.1098/rsta.2015.0202> [Accessed 19 May 2025].
21. Abdi, H. & Williams, L.J. (2010). *Principal component analysis*. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4), 433–459. <https://doi.org/10.1002/wics.101> [Accessed 19 May 2025].
22. Wikipedia, 2025. *Raisin*. [online] Wikipedia. [image online] Available at: <https://en.wikipedia.org/wiki/Raisin> [Accessed 19 May 2025].
23. Wikipedia, 2025. *Raisin*. [online] Wikipedia. Available at: <https://en.wikipedia.org/wiki/Raisin> [Accessed 19 May 2025].
24. Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81-106 [Accessed 19 May 2025]..