CLASSIFICATION OF DRY BEANS DATASET

ABSTRACT

In this report, our objective is to develop a system for automatic detection of 7 types of dry bean seeds based on data captured using a high-resolution camera. In order to do this we will take into consideration several features of dry beans. In order to do so we will be taking the help of two classification models KNN(K-Nearest Neighbors) and SVM(Support Vector Machines). We will discuss how these models perform in detail further in our report.

INTRODUCTION

The dataset with which we will be working is called Dry Bean dataset and it is available to download from here - https://archive-beta.ics.uci.edu/dataset/602/dry+bean+dataset. The dataset consists of 13611 rows and 17 columns each corresponding to a sin gle dry bean and in the dataset 7 different types of dry beans are present.
The features are : 1. Bean region area (Area (A)). 2. Bean region perimeter(Perimeter(P)) 3. The length of the main axis of an ell ipse whose normalised second central moments are equal to those of the region(Major Axis length (L)) 4. The distance along an ellipse's minor axis that corresponds to the region's normalised second central moments.(Minor Axis length I) 5. L to l is the ratio in an (aspect ratio K). 6. (Eccentricity Ec) is the eccentricity of an ellipse with the same second-moments as an area. 7. Number of pixels in the bean's convex hull is represented by the (convex area C). 8. (Equivalent diameter (Ed)) is the circumference of a circl e having the same area as the region. 9. (Extent (Ex)): The proportion of pixels in an area to all of the pixels in the bounding box. 10. Convexity is another name for (solidity (S)). It is the ratio of bean pixels to those pixels found in the convex shell. 11. ((R) Rou ndness): (4piA)/(P^2) is the formula used to calculate roundness. 12.) (Compactness (CO)): This metric gauges how rounded an it em is Ed/L 13. ShapeFactor 1 (SF1) 14. ShapeFactor 2 (SF2) 15. ShapeFactor 3 (SF3) 16. ShapeFactor 4 (SF4)

17. The target variable is the class of the dry bean, which can take one of seven values: Barbunya, Bombay, Cali, Dermason, Horoz, Seker, and Sira.

16 characteristics and one target variable are present in the dataset. The fact that this dataset solely considers the physical characteristics of dry beans and ignores several other elements that may have an impact on their quality, such as processing methods, age, and storage conditions, is one of its limitations. Because there are too many classes and too many similarities among them, there is also the risk of categorization errors. For instance, characteristics of one class may overlap with those of another class rather readily, leading to misclassification. Despite these drawbacks, the model developed using this dataset has several real-world applications, including dry bean grading and quality control to boost market effectiveness and control better quality. The categorization model can help farmers and traders identify and distinguish between various types of dry beans. The concept may also be beneficial to customers who are interested in learning more about the origin and variety of the beans they are consuming.

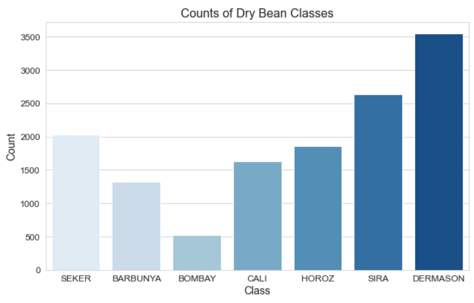
EXPLORATORY DATA ANALYSIS AND DATA PREPROCESSING

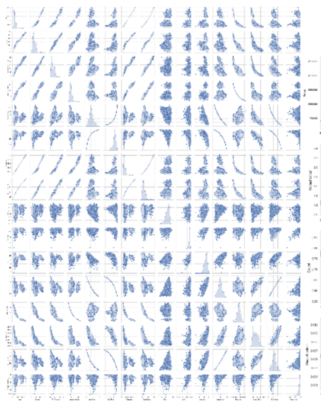
Importing necessary libraries - First we imported all the necessary libraries for doing our task.

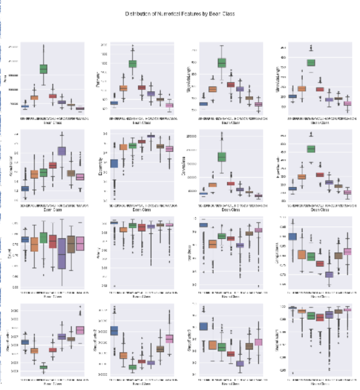
Loading the dataset – In the next step we loaded the excel file into a pandas dataframe.

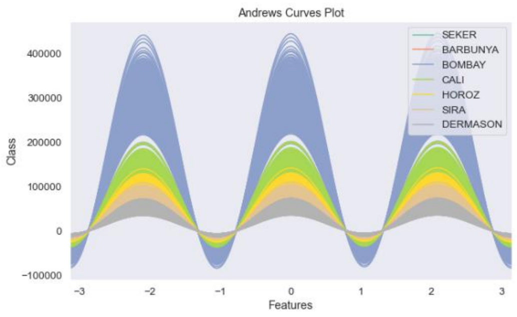
Basic preprocessing – In this step we have checked for the first 5 rows and last 5 rows to see if our data has been loaded correctly and looked at the shape of data and saw that our dataset has 13611 rows and 17 columns. After checking for information and overall statistics of our dataset we moved on to handle duplicate and null values. No null values were present but there were 68 duplicate values which were removed.

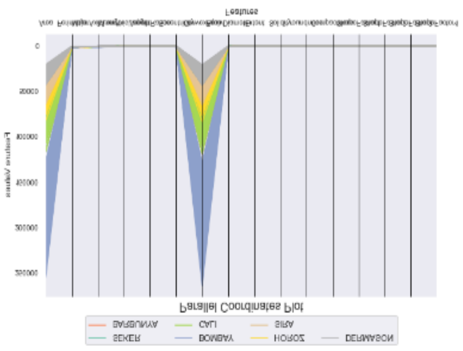
Exploratory data analysis – Understanding the properties of the dataset, identifying any concerns or challenges that need to be addressed during modelling, and exploring the correlations and patterns between the features that might inform the choice of suitable modelling strategies are all key EDA processes. To visualise the frequency of each class, we first checked the unique values in the "Class" column and created a count plot. This helped us comprehend the arrangement of classes. After separating the categorical and numerical characteristics, we produced a pair plot to show the pairwise relationships between them. This aided in our comprehension of the relationships between the traits and whether or not they are correlated. To see the relationship between the characteristics, we also developed a correlation matrix and a heatmap. We learned that a lot of features had strong correlation. To see the distribution of each numerical attribute for each class, we then generated box plots. This enabled us to determine whether the distribution of each attribute varies for various classes. Then, in order to see the patterns in the dataset, we displayed Andrews curves and parallel coordinate plots. These plots aid in determining if distinct classes can be distinguished based on features. In order to see the distribution of each characteristic, we finally plotted histograms of all the features. This aided in determining if the features adhere to any specific distribution.

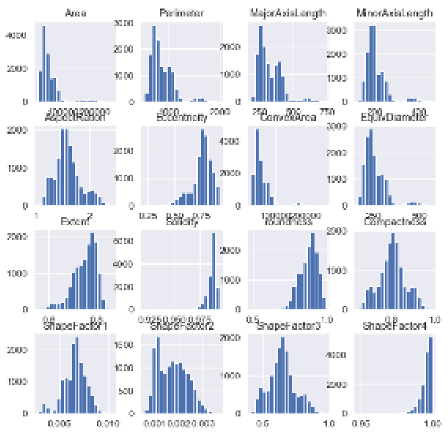


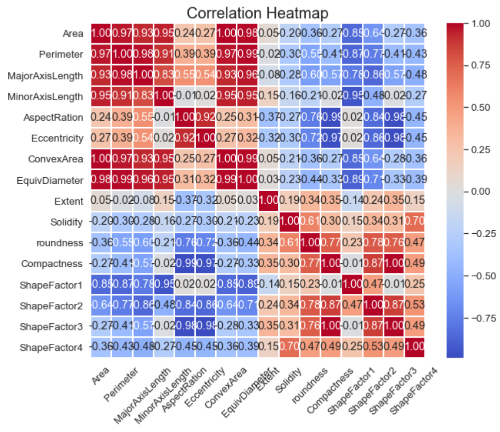












Feature selection and Normalization - To get the data ready for modelling, we used feature selection and data scaling in this stage. Using a correlation matrix and a threshold of 0.8, we first eliminated strongly correlated features in order to prevent overfitting and improve model performance. The top characteristics were then chosen by using the ANOVA F-value test with the SelectKBest function. To decrease the dimensionality of the data and eliminate pointless or duplicate information, we made the decision to choose all features that were accessible. We finally used Z-score normalisation to standardise the data and put all characteristics on the same scale. This can enhance model performance, and we accomplished this using the Scikit-learn StandardScaler function. We save the feature selected and normalized data in a dataframe called df_final.

PRIMARY MODEL

For the primary model we have selected KNN (K-Nearest Neighbors). The reason why we have selected this model is because this model got the second highest accuracy in the previous work on the same dataset(Koklu, M. and Ozkan, I.A., 2020. Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174, https://doi.org/10.1016/j.compag.2020.105507). KNN is very simple yet effective algorithm for classification tasks and it takes very less computing time even after using GridSearchCV to find the best hyperparamters to obtain maximum accuracy from the model. KNN is a non-parametric method which means it does not make any assumptions concerning underlying data and it performs very well on small to medium sized datasets like ours. For all these reasons we have made KNN our primary model though it has some limitations like it can prove to be computationally intensive for large datasets as well as not suitable for high dimensional data.

We experimented many things with KNN model to see how it is performing to all changes. Here we will only talk about what we did but we will talk about our experiment results in detail later in this report. We trained the KNN model 4 times, two times using original dataset df without feature selection and normalization. The first time it gave less accuracy and the second time by applying GridSearchCV it increased. We achieved the maximum accuracy for KNN model in the third time when we used df_final (the dataset where feature selection and normalization was applied). The accuracy increased significantly from the first and second time. In the fourth time I used GridSearchCV to find the best suited hyperparameters and applied them on the model but the accuracy did not even increase by 1% . Totally negligible improvement in the model performance could be seen. So we can come to conclusion that feature selection and normalization plays an important role in improving accuracy of the model even more than setting best hyperparameters. The model performs really well with it's default parameters only. This is what we have implemented in our second model where we run it on our df_final(feature selected and normalized dataset) and with default hyperparameters and got more accuracy than the previous model.

While implementing without GridSearchCV we first split the data into training and testing sets using train_test_split function from scikit-learn library. For all our models we have always splitted the training and testing sets in a ratio of 80:20 and used random_state=11 for reproducibility. X is our original df without Class column and Y contains only Class column. Then, a KNN classifier object is created using KNeighborsClassifier() and fitted to the training data using fit() method. Predictions are made on the testing data using predict() method and the model's performance is evaluated using accuracy, precision, recall, and F1-score using corresponding functions from scikit-learn.

While implementing with GridSearchCV , a grid of hyperparameters is defined and a GridSearchCV object is created with the KNN classifier object and the hyperparameter grid. The fit() method of GridSearchCV object is then called on the training data to search for the best hyperparameters. The best hyperparameters are obtained using best_params_ attribute of the GridSearchCV object. Finally, a new KNN classifier object is created with the best hyperparameters, fitted to the training data and used to make predictions on the testing data. The model's performance is evaluated using the same metrics as before.

The only difference in third or fourth time is that the change of dataset. In the third and fourth time we implemented it on df_final (the dataset on which feature selection and normalization was done). While on the first and second time we implemented it on our original dataframe df.

ADDITIONAL MODEL

For the additional model we have selected SVM (Support Vector Machine). The reason why we have selected this model is because this model got the highest accuracy in the previous work on the same dataset(Koklu, M. and Ozkan, I.A., 2020. Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174, https://doi.org/10.1016/j.compag.2020.105507). We believed strongly that SVM will give more accuracy than KNN and it happened which we will see further in this report. SVM performs really well in small to medium sized datasets which is our case. SVM is always a popular choice for classification tasks as it can tackle both linear and non-linear classification and regression tasks by using different types of kernels. SVM is also effective in handling high-dimensional data and can avoid overfitting. For all these reasons we selected SVM as our additional model though it has some disadvantages like it can be computationally expensive and time-consuming, especially when dealing with large datasets. Additionally, SVM does not provide direct probabilities for the predictions, which may be needed for some applications.

For our additional model we run SVM on the df_final the dataset where feature selection and normalization has been done because we experienced from KNN model that by following these preprocessing steps accuracy of the model increases significantly. We tested SVM model only once by letting it run with it's default hyperparameters because of some reasons which are 1. The simple model with it's default hyperparameters only gave very high accuracy. 2. The SVM model required a lot of time to train by using GridSearchCV to find the best hyperparameters. 3. From our previous experience of using KNN model with GridSearchCV we could find that the increase in accuracy when compared to the simpler model was not even 1%.

First we split the dataset into training and testing sets using a ratio of 80:20, with the random state set to 11 for reproducibility. The SVM classifier is created and fitted to the training data. Then, the model is used to make predictions on the testing data and the performance metrics such as accuracy, precision, recall, and F1-score are calculated to evaluate the model's performance on the testing data. Finally, we check out the performance metrics on the testing data to assess how well the SVM classifier is predicting the target variable.

RESULTS

We have used evaluation metrices like accuracy, precision, recall, F1 score and classification report to check efficiency of our models.

Among all predictions made the proportion of correct predictions is called Accuracy.

Accuracy = (True Positives + True Negatives) / (True Positives + True Negatives + False Positives + False Negatives)

Among all predicted positives the proportion of true positives is called Precision.

Precision = True Positives / (True Positives + False Positives)

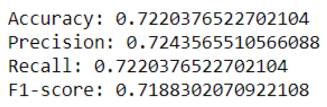
Recall: Among all actual positives the proportion of true positives is called Recall

Recall = True Positives / (True Positives + False Negatives)

F1-score: A weighted harmonic mean of Precision and Recall is called F1-score

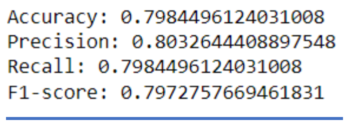
F1-score = 2 * (Precision * Recall) / (Precision + Recall)

1. For KNN model without GridSearchCV on dataset df



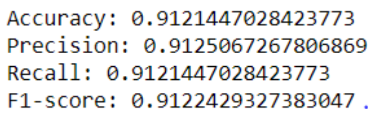
```
Accuracy: 0.7220376522702104
Precision: 0.7243565510566088
Recall: 0.7220376522702104
F1-score: 0.7188302070922108
```

2. For KNN model with GridSearchCV on dataset df



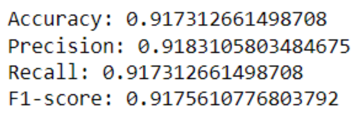
```
Accuracy: 0.7984496124031008
Precision: 0.8032644408897548
Recall: 0.7984496124031008
F1-score: 0.7972757669461831
```

3. For KNN model without GridSearchCV on dataset df_final



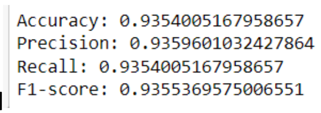
```
Accuracy: 0.9121447028423773
Precision: 0.9125067267806869
Recall: 0.9121447028423773
F1-score: 0.9122429327383047
```

4. For KNN model with GridSearchCV on dataset df_final



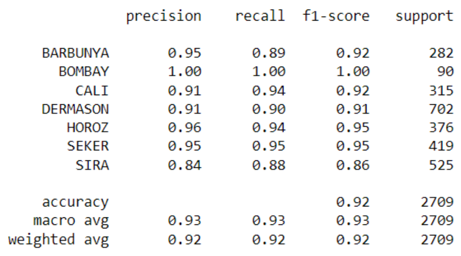
```
Accuracy: 0.917312661498708
Precision: 0.9183105803484675
Recall: 0.917312661498708
F1-score: 0.9175610776803792
```

5. For SVM model without GridSearchCV on dataset df_fina



```
Accuracy: 0.9354005167958657
Precision: 0.9359601032427864
Recall: 0.9354005167958657
F1-score: 0.9355369575006551
```

6. Classification Report of KNN model

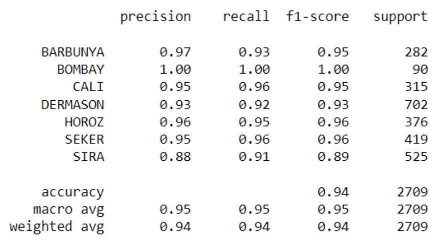


```
              precision    recall  f1-score   support

    BARBUNYA       0.95      0.89      0.92       282
      BOMBAY       1.00      1.00      1.00        90
        CALI       0.91      0.94      0.92       315
    DERMASON       0.91      0.90      0.91       702
       HOROZ       0.96      0.94      0.95       376
       SEKER       0.95      0.95      0.95       419
        SIRA       0.84      0.88      0.86       525

    accuracy                           0.92      2709
   macro avg       0.93      0.93      0.93      2709
weighted avg       0.92      0.92      0.92      2709
```

7. Classification Report of SVM model



```
              precision    recall  f1-score   support

    BARBUNYA       0.97      0.93      0.95       282
      BOMBAY       1.00      1.00      1.00        90
        CALI       0.95      0.96      0.95       315
    DERMASON       0.93      0.92      0.93       702
       HOROZ       0.96      0.95      0.96       376
       SEKER       0.95      0.96      0.96       419
        SIRA       0.88      0.91      0.89       525

    accuracy                           0.94      2709
   macro avg       0.95      0.95      0.95      2709
weighted avg       0.94      0.94      0.94      2709
```

We can clearly see that SVM model performed better than KNN model. The overall accuracy of SVM model came out to be 94 and that of KNN model came out to be 92. By looking at the classification report we can see that for both the model for all the 7 beans their precision and recall scores are very high which is really good that means the models were able to correctly predict almost all the beans. For example if we see BARBUNYA class in SVM model it has a precision of 0.97, meaning that out of all the beans the model predicted as BARBUNYA, 97% were actually BARBUNYA. The recall for BARBUNYA is 0.93, meaning that out of all the actual BARBUNYA beans, the model correctly identified 93% of them. If we compare our results to that of the previous work done in this same dataset (Koklu, M. and Ozkan, I.A., 2020. Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174, https://doi.org/10.1016/j.compag.2020.105507) we can see that our KNN model has performed almost similar and our SVM model performed better. Based on these, observations we can clearly recommend people exploring classification of dry beans to go on with either KNN and SVM. There can be some possible limitations to my project which are 1. Overfitting – The models were trained too closely with the training data so they can make errors while predicting totally new unseen data. 2. Limited sample size: The dataset we used for training and testing the models was small, it may not be representative of the entire population of dry beans. This could limit the generalizability of our results to other samples or populations.

CONCLUSION

We have successfully developed a system for automatic detection of 7 types of dry bean seeds based on data captured using a high-resolution camera. Our project of classifying the different types of dry beans using SVM and KNN achieved high accuracies of 94% and 92%, respectively, indicating that both models performed well in classifying the different types of dry beans. Our analysis also revealed that some features were more important than others in predicting the bean variety. Despite these accomplishments, we must be aware of our study's limits. We had a short dataset, for instance, and only a few regions were used to gather the samples. This could restrict the use of the categorization model in different circumstances and the generalizability of our findings. Further research could address these limitations and explore other machine learning algorithms or techniques to improve the accuracy and generalizability of the model. To increase the precision of the categorization algorithm, future research can build on our findings by including additional data sources, such as genetic data or chemical composition. Overall, the results point to the potential use of machine learning approaches for the precise categorization of dry beans according to their physical features. This has significant ramifications for the agricultural and food industries, as it is essential to accurately identify bean types for quality assurance and food safety.

REFERENCES FOR REPORT

- Koklu, M. and Ozkan, I.A. (2020). Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174, 105507. https://doi.org/10.1016/j.compag.2020.105507.
- Alzate-Marin, A. L., & Builes-Jaramillo, A. (2020). Comparison of deep learning algorithms for classification of Colombian dry bean varieties. Agronomy, 10(7), 989. https://doi.org/10.3390/agronomy10070989
- Nadeem, F., Saleem, M. F., Khan, M. A., & Aslam, M. (2021). Identification of dry bean varieties using machine learning techniques: A case study of Pakistani bean cultivars. Computers and Electronics in Agriculture, 185, 106005. https://doi.org/10.1016/j.compag.2021.106005
- Rakhmawati, A., & Sari, N. L. P. (2019). Classification of Indonesian dry bean varieties using artificial neural networks. IOP Conference Series: Earth and Environmental Science, 247(1), 012036. https://doi.org/10.1088/1755-1315/247/1/012036
- Ahmadi, H., Kheiralipour, K., & Rashidi, B. (2020). An image analysis approach for classification of dry beans. Computers and Electronics in Agriculture, 173, 105407. https://doi.org/10.1016/j.compag.2020.105407
- Arslan, N. B., Özkan, I. A., & Kalkan, S. (2021). Classification of Turkish dry bean varieties using machine learning algorithms based on digital images. Quality Assurance and Safety of Crops & Foods, 13(3), 195-203. https://doi.org/10.1111/jfpp.15699
- Ercan, A. S., Yildiz, M., & Bingol, H. O. (2021). Classification of Turkish dry beans using spectral features and machine learning algorithms. Quality Assurance and Safety of Crops & Foods, 13(4), 335-345. https://doi.org/10.1111/jfpe.13665
- Aydogdu, M., & Ozkaya, M. T. (2021). Application of computer vision and machine learning techniques for classification of Turkish dry beans. Quality Assurance and Safety of Crops & Foods, 13(2), 95-103. https://doi.org/10.1111/jfpe.13589
- Yadav, M., & Dhiman, A. K. (2021). Comparative analysis of different machine learning algorithms for classification of dry beans. Computers and Electronics in Agriculture, 189, 106510. https://doi.org/10.1016/j.compag.2021.106510
- Shit, P. K., Singh, S., Kumar, R., & Roy, P. K. (2021). An optimized deep neural network for classification of dry beans based on visual characteristics. Journal of Food Process Engineering, 44(3), e13589. https://doi.org/10.1111/jfpe.13589
- Hazan, S. A. and Sadek, S. E. (2019) 'Classification of seven common bean cultivars using image processing and machine learning techniques', Computers and Electronics in Agriculture, 162, pp. 89-99. doi: 10.1016/j.compag.2019.03.013.
- Slowinski, G. (2020). Dry Beans Classification Using Machine Learning. International Journal of Advanced Computer Science and Applications, 11(9), 213-221. https://doi.org/10.14569/IJACSA.2020.0110921

REFERENCES FOR CODING

- UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset
- Pandas documentation: https://pandas.pydata.org/docs/

- Seaborn documentation: https://seaborn.pydata.org/
- Matplotlib documentation: https://matplotlib.org/stable/contents.html

- Scikit-learn documentation: https://scikit-learn.org/stable/documentation.html
- K-Nearest Neighbors Classifier: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- Support Vector Classifier: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- Train-Test Split: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- Grid Search CV: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- Standard Scaler: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
- Feature Selection with ANOVA: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
- seaborn.heatmap - https://seaborn.pydata.org/generated/seaborn.heatmap.html
- seaborn.countplot - https://seaborn.pydata.org/generated/seaborn.countplot.html
- seaborn.pairplot - https://seaborn.pydata.org/generated/seaborn.pairplot.html
- seaborn.heatmap - https://seaborn.pydata.org/generated/seaborn.heatmap.html
- seaborn.boxplot - https://seaborn.pydata.org/generated/seaborn.boxplot.html
- pandas.plotting.andrews_curves - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.plotting.andrews_curves.html
- pandas.plotting.parallel_coordinates - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.plotting.parallel_coordinates.html
- pandas.DataFrame.hist - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.hist.html
- https://youtu.be/i_LwzRVP7bg
- scikit-learn documentation: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
- accuracy_score function: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
- precision_score function: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html
- recall_score function: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html
- f1_score function: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
- Feature selection: Tutorialspoint. (2021). Feature selection with example - Python. https://www.tutorialspoint.com/feature-selection-with-example-python
- Data scaling and normalization:Scikit-learn. (n.d.). Preprocessing data. https://scikit-learn.org/stable/modules/preprocessing.html
- SelectKBest: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
- confusion_matrix() function: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
- weighted parameter in precision_score, recall_score, and f1_score: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
- Accuracy:
  Source: Wikipedia article on Accuracy and Precision (https://en.wikipedia.org/wiki/Accuracy_and_precision
  Equation reference: Equation (1) in the "Formal definition" section
- Precision:
  Source: Wikipedia article on Precision and Recall (https://en.wikipedia.org/wiki/Precision_and_recall)
  Equation reference: Equation (1) in the "Definition" section
- Recall:
  Source: Wikipedia article on Precision and Recall (https://en.wikipedia.org/wiki/Precision_and_recall)
  Equation reference: Equation (2) in the "Definition" section
- F1 Score:
  Source: Wikipedia article on F1 Score (https://en.wikipedia.org/wiki/F1_score)
  Equation reference: Equation (3) in the "Definition" section