Introduction to Data Science with Python for engineers and researchers

Summative Assessment 1 – Written Essay

Binomial Classification for Raisin Grains

Course Convener: Dr. Joshua F. Einsle

Student: Allan Hernandez Cotto, 2589702H

7 July 2022

Introduction

The purpose of this study is to implement an application of supervised machine learning for a binary classification problem using Python. Machine learning fundamentally involves building mathematical models to help understand data. [1] Supervised machine learning refers to working with a set of labeled training data. For every example in the training data there is an input object and an output object. [2] This project will implement data science and machine learning techniques and tools to understand how to classify between two raisin grain categories. The methodology section provides a clear overview of all the design choices and implementations on this project. The results section presents the findings from this study in the form of performance metrics for the model. The conclusion brings together main areas of the report and gives a final judgement of this study. Finally, the action plan presents some alternative techniques to explore and research to potentially improve the model performance.

Problem statement

The problem statement is to predict the correct raisin class given a set number of features. The aim of the project is to train a machine learning model to predict a class based on the input features of the record. This prediction will be compared to the real result to determine the performance metrics and assess the model performance.

Description of data

The project utilizes the Classification of Raisin Grains Using Machine Learning and Artificial Intelligence Methods dataset [3] to implement a supervised machine learning solution to solve a classification problem. This dataset is in CSV (comma-separated values) format and consists of 900 samples of grains data, equally distributed between the two classes of raisin grains to analyze: Kecimen and Besni. Each entry has 7 numeric features and a category label. Originally, the images were subject to feature extraction using image processing techniques.

	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

Figure 1 Raisin Dataset

Methodology

The methodology for this project consists of following the steps for a traditional machine learning workflow. This process is explained and presented in a logical order. Likewise, elements from the Machine learning pipeline are considered and explored in detail.

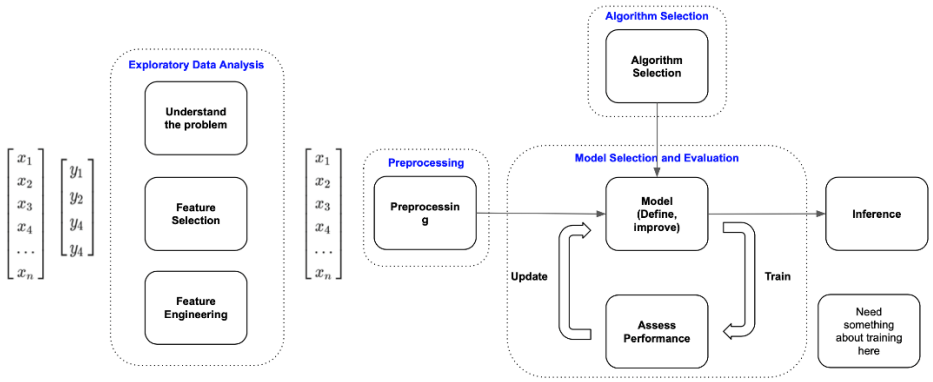

Figure 2 Machine Learning Workflow

	Initial Data Analysis: *Understand the data quality and properties of data*	**Exploratory Data Analysis:** *Understand what the data is telling us*	**Model Preparation:** *Prepare Data for modelling*
Properties Understand the properties of a dataset	**What are the 'internal' properties of the data?** • Data types (computational - e.g. strings, integers) • Data size • Data shape • Missing values • Feature descriptions • Data provenance	**What are the 'external' properties of the data?** • Data types (statistical - e.g.continuous, discrete, categorical) • Unique and Duplicate entries	**What are the properties of the data most relevant to building a model?** • Class imbalance
Preprocessing Get data in the right shape for the next stage	**How should I treat these 'internal' characteristics?** • Change data types • Reshape data **What other changes to the data are required?** • Drop target variable (if known)	**How should I treat these 'external' characteristics?** • Normalize or Standardize the data if required for analysis • Deal with missing values • Deal with categorical data **What other changes to the data are required?** • Drop target variable (if known)	***How* to prepare the data for modelling?** • Normalize and Standardize the data if required for the model • Deal with class imbalances if necessary (can be done as part of training strategy)
Understanding of data and how it informs the problem we are investigating	**What will help with the next steps?** • Identify if there is missing data and acquire it • Develop Data Dictionary and get feature descriptions if required • Understand provenance and go to the source • Is this a sample or all the data? • Data size suggests strategy for processing (e.g. hadoop, etc.)	**Non-computational activities to aid understanding:** • Generate assumptions about how we are treating the data • Generate hypotheses for investigation • Identify new data requirements **Undertake analysis:** • Univariate • Bivariate • Multivariate • Visualisations	***What* data should be modelled?** • Feature Selection • Feature Engineering • Identify initial ways of modelling data

Figure 3 Machine Learning Pipeline

Initial data analysis

An initial data analysis was performed using pandas methods to understand the dataset properties. The info() method shows all columns and their respective data type. This is useful as it shows that this dataset consists of seven numeric features (int64 and float64) and one categoric object (Class) which will be the target. The describe() method was used to have a better understanding of the mean, standard deviation, and limit values of the data. From this initial analysis it can also be determined that there are no missing values on any feature. This means that there is no need to drop null values or replace them with the mean of the entire series, which makes the preprocessing stage simpler.

```
<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
```

Figure 4 Output of .info() method on data frame

A pair plot was created using the seaborn library to visualize the relationship between features and the marginal data distribution of the data in each column, based on the Class object. [4] The plot illustrates the best set of features to explain the relationship between the classes. Most of these combinations show a clear division between Kecimen and Besni raisins.

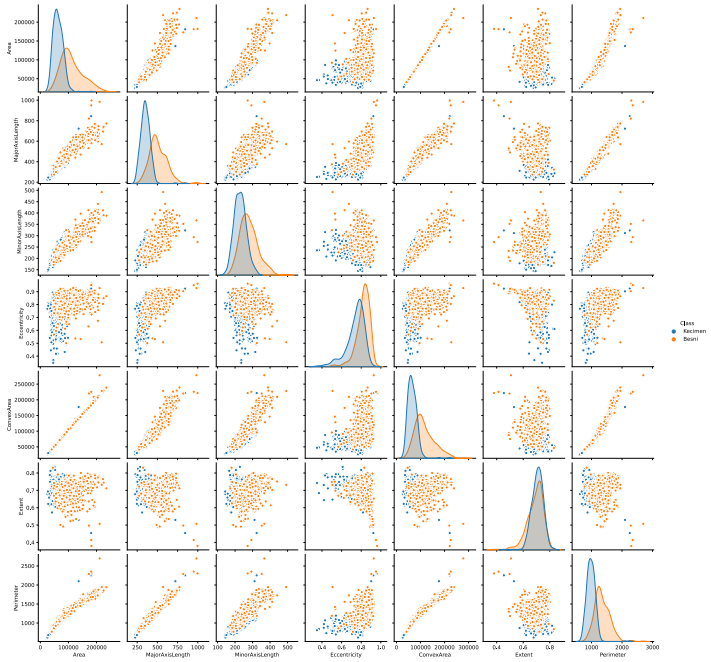

Figure 5 Pair plot

Next, the data must be separated into features and target, in preparation to fit and train the model. A new data frame representing the features, X, was created from the original data loaded from the CSV file, but without the last column (target). For the target data frame, the categories Kecimen and Besni must be converted into numeric values, 0 and 1 respectively, to facilitate the development of the machine learning model. The target y was created using list comprehension to update the categories from text to 0s and 1s, and then applying a numpy array function to it.

```
# Create dataframe with all features
X = df.iloc[:, 0: -1]

# Create numpy array for the target with binary classification
y = np.array([0 if c == 'Kecimen' else 1 for c in df['Class']])
```

Figure 6 Defining features (X) and target (y)

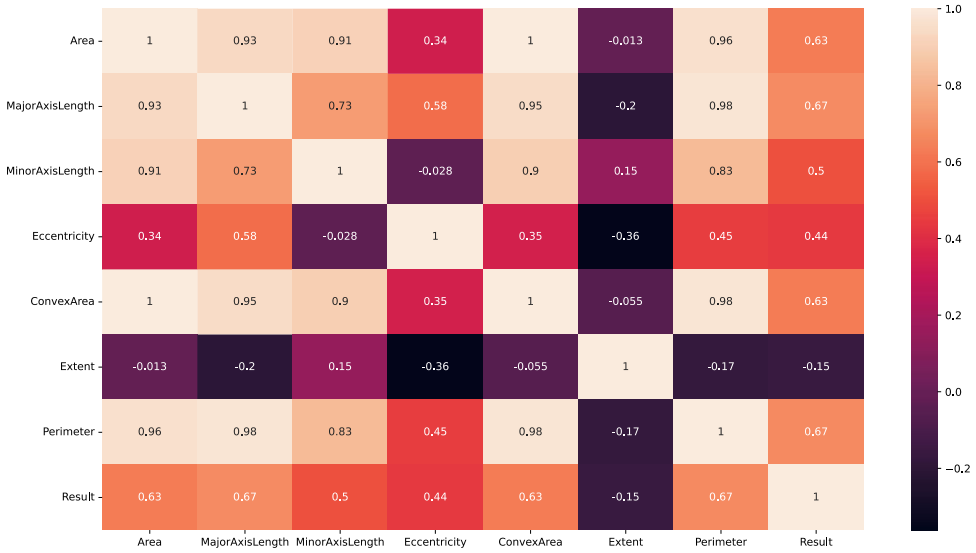

Figure 7 Heat map

A heat map was created with seaborn to visualize the correlation between the feature pairs and the target, shown as 'Result' in the right side of the map. A dark purple shade represents negative correlation between the variables and the target, whereas brighter shades of red or lighter colors illustrate a high correlation between them. The heat map shows that there are four out of seven features with a correlation value over 0.6 with respect to the target, including Area, MajorAxisLength, and Perimeter. These features also have a correlation with each other. This suggests that these physical values are significantly linked to each other. On the contrary, the Extent feature has a low or negative correlation with other features and with the target. This analysis provides insight into the role of the features in the data frame and how they can impact the model.

Data split

The data must be split into training and testing sets before training the model and evaluating performance. The train_test_split() function from sklearn assists in creating random training and testing subsets. This function takes a features dataframe (X) and a target numpy array (y) as arguments and returns the train and test splits for the features and the target. The optional parameter of test_size was set to 0.2 to assign a 20% split to the test data, and a random state of 0 was selected for consistency in results. The shape and number of datapoints of the training and test sets was verified and is consistent with the initial parameter settings.

```
# Train/Test data split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
✓ 0.9s
```

Figure 8 Train and test data split

```
# Get shape of test and training sets
print('Training Set:')
print('Number of datapoints: ', X_train.shape[0])
print('Number of features: ', X_train.shape[1])
print('\n')
print('Test Set:')
print('Number of datapoints: ', X_test.shape[0])
print('Number of features: ', X_test.shape[1])
```

```
✓ 0.1s
Training Set:
Number of datapoints:  720
Number of features:  7


Test Set:
Number of datapoints:  180
Number of features:  7
```

Figure 9 Training and Test set characteristics

Data Scaling

The next step consists of performing data scaling. The dataset input variables have different magnitudes, ranges, and units. Machine learning models require feature scaling to process all input variables in the same scale. Machine learning models require feature scaling to process all input variables in the same scale, as they might have different orders of magnitude and units. This project implements a Standard Scaler, which standardizes the features by removing the mean and scaling to unit variance. [4]

Logistic Regression

Logistic regression is used to determine the probability of an event in classification problems. Normally, the event is represented as a categorical dependent variable. [5] The probability of the event is expressed using the sigmoid function. This function represents the natural logarithm log(x) of a variable x for values between 0 and 1 and is therefore suitable for applications in classification. [6] The logistic regression technique was chosen because it is a fast and offers clear interpretations for binary classification. [10]

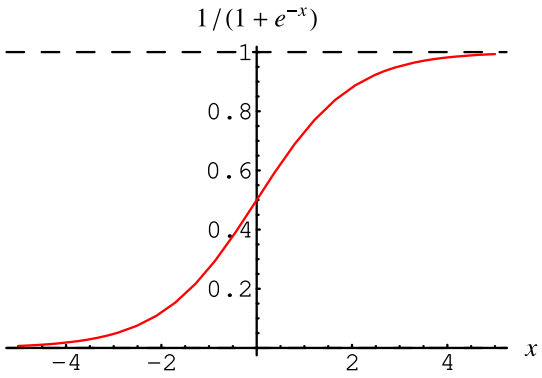

Figure 10 Sigmoid function [7]

The logistic regression model was implemented in Python. The solver parameter determines the algorithm to be used in the optimization problem. Documentation for the function suggests using 'liblinear' for small datasets, such as the raisin dataset with 900 entries. A random state of 1 ensures consistency in the results during testing.

```
model = LogisticRegression(solver='liblinear', random_state=1)
model.fit(X_train, y_train)
```
✓ 0.1s

Figure 11 Logistic regression model

Results

After the model has been trained, its predictions on the test subset should be evaluated to assess its performance. Classification problems use several evaluation metrics, such as accuracy, F1 score, and ROC AUC. An evaluation function was created to calculate three different metrics for this model. The most straightforward and relevant metric for this study is the accuracy. From this evaluation it can be observed that the classification accuracy is around 87.78%.

```
def evaluate(y_test, y_pred):
    accuracy = metrics.accuracy_score(y_test, y_pred)
    f1 = metrics.f1_score(y_test, y_pred)
    auc = metrics.roc_auc_score(y_test, y_pred)

    print(f"Accuracy:   {accuracy}")
    print(f"F1 Score:   {f1}")
    print(F"ROC_AUC:    {auc}")

evaluate(y_test, y_pred)
```

✓ 0.1s

```
Accuracy:   0.8777777777777778
F1 Score:   0.8829787234042553
ROC_AUC:    0.8780185758513932
```

Figure 12 Performance evaluation function

A visual representation of how the accuracy was determined can be accessed by creating a confusion matrix, which illustrates how well the model was able to predict raisin categories compared to the real result. This plot shows the true positives, false positives, true negatives, and false negatives for the predictions. It can be observed that 75 raisins were correctly categorized as 0 (Kecimen) and 83 were predicted as 1 (Besni). There were 22 incorrect predictions in total, which generate a loss that lowers the accuracy score. Even though the accuracy seems reasonably high, for larger test sets there could be a significant number of erroneous predictions.

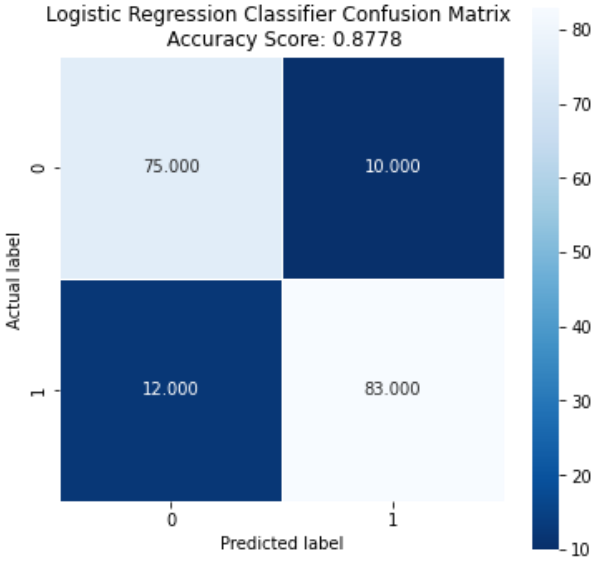

Figure 13 Confusion Matrix

Additional optimizations must be researched to improve the model performance and reduce the number of incorrect predictions. Alternatively, different classification methods could also be explored to determine if there are improvements. These ideas are discussed in greater detail in the action plan section.

Conclusion

This project was undertaken to design and evaluate the performance of a machine learning model for binomial classification between the Kecimen and Besni classes of raisins. A comprehensive process was carried out to understand and solve the classification problem, including performing exploratory data analysis, data preprocessing, creating visualizations, splitting the train and test data, model selection, fitting and training the model, and finally evaluating the model performance. The study implemented a logistic regression model and identified an accuracy of 87.78% in correctly identifying the based on the test input variables. A confusion matrix summarized the results from the model training and illustrated how the accuracy and loss was calculated. This project presented a valuable opportunity to develop technical and analytical skills for data science and machine learning using Python.

Discussion and action plan

Different methods have been proposed for this binomial classification problem, including multilayer perceptron (MLP) and support vector machine (SVM) implementations. Further research is required to explore and compare these techniques with the current approach of logistic regression. Furthermore, there are additional potential alternatives for a classification model, including random forest, gradient boosting tree, decision trees, and Naïve Bayes. Cross-validation techniques should be considered to assist in determining the optimal classification technique for the model and optimize its respective hyperparameters to improve its performance. [10] Finally, the implementation of TensorFlow, an open-source software library for AI and machine learning, could provide facilitate model training using deep neural networks.

References

[1] Jake VanderPlas, "Python Data Science Handbook" https://jakevdp.github.io/. https://jakevdp.github.io/PythonDataScienceHandbook/

[2] Bell, Jason. *Machine Learning: Hands-On for Developers and Technical Professionals*, John Wiley & Sons, Incorporated, 2014. *ProQuest Ebook Central*, https://ebookcentral.proquest.com/lib/gla/detail.action?docID=1818248.

[3] 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

[4] "scikit-learn Machine learning in Python" https://scikit-learn.org/stable/

[5] Antonio Gulli, Amita Kapoor, Sujit Pal, "Deep Learning with TensorFlow 2 and Keras"

[6] Mirko Stojiljković, "Logistic Regression in Python", realpython.com, https://realpython.com/logistic-regression-python/

[7] "Sigmoid Function", mathworld.wolfram.com, https://mathworld.wolfram.com/SigmoidFunction.html

[8] Tanay Agrawal, "Hyperparameter Optimization in Machine Learning"