21.05.2023

Murad Rajabov

201ADB075

Practical Work #2

GitHub link: https://github.com/muradphpsnik/PracticalWork2

Overview of Orange tool workflow:

A screenshot of a computer

Description automatically generated with low confidence

On this screenshot I am demonstrating my workflow for practical work. All three parts are included: data exploration is on above, supervised learning in the middle, and unsupervised learning on the bottom of the workflow. All three parts I will describe in detail explaining my choices, solution, and etc.

Part I – Pre-preprocessing/Exploring the data.

Description of the Dataset:

Title of the dataset: Rice (Cammeo and Osmancik) Data Set

Source: <https://archive.ics.uci.edu/ml/datasets/Rice+%28Cammeo+and+Osmancik%29>

Author and owner of the Dataset:

Ilkay CINAR (Graduate School of Natural and Applied Sciences, Selcuk University, Turkey),

Murat KOKLU (Faculty of Technology, Selcuk University, Turkey).

Description of the problem domain:

Among all certified sorts of rice growing in Turkey Cammeo and Osmancik sorts have been chosen for this research. Two students made overall 3810 photos of Cammeo and Osmancik sorts and processed them. After that, they came up with a dataset describing all differences and equalities between the two sorts.

Licensing:

There is no licensing of dataset that was chosen for this practical work.

The way dataset was collected:

Dataset was found on one of the links in “Description and Requirements”: UC Irvine Machine Learning Repository. The dataset was downloaded in XLSX(Excel) format and converted to CSV format using Pandas library(pd.read\_excel(), and pd.to\_csv() functions). Nothing in the dataset was changed, all data used throughout the practical work is original.

Description of the content of dataset:

The dataset consists of seven distinctive features describing rice characteristics in detail. There is one classifying attribute (Class) with two discrete values: Cammeo and Osmancik. The remaining attributes describe characteristics of grains of rice:

‘Area’ – represents the number of pixels within the boundaries of the grain. Value type: numeric. Range used: [0,1].

‘Perimeter’ – determines the space surrounding the edges of the grain. Value type: numeric. Range used: [0,1].

‘Major Axis Length’ - The main axis distance, or the longest line that can be drawn on a grain of rice. Value type: numeric. Range used: [0,1].

‘Minor Axis Length’ - The minor axis distance, or the shortest line that can be drawn on a grain of rice. Value type: numeric. Range used: [0,1].

‘Eccentricity’ – measures how round the grain of rice is. Value type: numeric. Range used: [0,1].

‘Convex Area’ - returns the number of pixels in the rice grain region's smallest convex shell. Value type: numeric. Range used: [0,1].

‘Extent’ - returns the ratio of the region that the rice grain formed to the pixels in the bounding box. Value type: numeric. Range used: [0,1].

‘Class’ – target classifying attribute specifying sort or class of grain of rice, having two discrete values: Cammeo and Osmancik. Value Type: categorical (0 or 1).

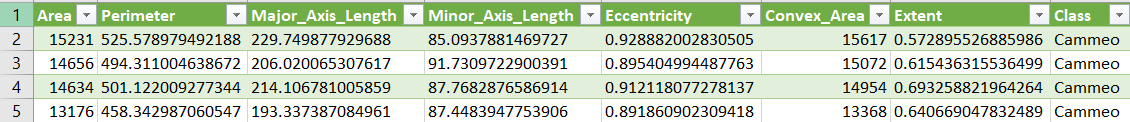
Therefore, each class has 7 attributes.

Number of data objects: 3810 instances, 8 columns, and 7 attributes.

Snippet of the structure of a datafile:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Area** | **Perimeter** | **Major\_Axis\_Length** | **Minor\_Axis\_Length** | **Eccentricity** | **Convex\_Area** | **Extent** | **Class** |
| 15231 | 525.578979492188 | 229.749877929688 | 85.0937881469727 | 0.928882002830505 | 15617 | 0.572895526885986 | Cammeo |
| 14656 | 494.311004638672 | 206.020065307617 | 91.7309722900391 | 0.895404994487763 | 15072 | 0.615436315536499 | Cammeo |
| 14634 | 501.122009277344 | 214.106781005859 | 87.7682876586914 | 0.912118077278137 | 14954 | 0.693258821964264 | Cammeo |
| 13176 | 458.342987060547 | 193.337387084961 | 87.4483947753906 | 0.891860902309418 | 13368 | 0.640669047832489 | Cammeo |
| 14688 | 507.1669921875 | 211.743377685547 | 89.3124542236328 | 0.906690895557404 | 15262 | 0.64602392911911 | Cammeo |

Screenshot for better view because it does not fit:



After targeting Class column and normalizing numeric values to interval [0,1] except Eccentricity and Extend because these two columns already were between 0 and 1 showing ratio:  
A screenshot of a computer

Description automatically generated with low confidence

Count of Cammeo class grains of rice: 1630

Count of Osmancik class grains of rice: 2180

Visual representation of the dataset:

Scatter plots:

After finding informative projections, I found out scatterplots with attributes making two classes separable the most:

Convex Area and Minor axis length:

A red and blue dots

Description automatically generated with low confidence

As we can see there is significant difference in Convex Area as Cammeo grains have bigger area, even last outlier has the biggest convex area which is 1. On the other hand, Osmancik grains of rice have less convex area and the last outlier is 0, which means it does not have convex shell. When it comes to Minor Axis length, it is obvious that Cammeo sort has longest minor axis, and Osmancik has the shortest minor axis. But overall, convex area as a feature shows more separability than minor axis length.

Eccentricity and Perimeter:

A red and blue dots

Description automatically generated with low confidence

This scatterplot shows significant separability between two classes based on Eccentricity and Perimeter. From graph, Osmancik grains of rice have lower eccentricity and smaller perimeter than Cammeo. The eccentricity of the most Osmancik grains of rice is between 0.84 and 0.91, and for Cammeo it is between 0.88 and 0.93. Osmancik has lower outlier and Cammeo has higher outlier. As for perimeter, Cammeo has bigger perimeter than Osmancik, which is obvious from the graph.

Histograms:

A screenshot of a graph

Description automatically generated with low confidence

Here I show histograms for all features of the dataset. Overall, dataset is balanced as can be seen, because in almost half of cased Osmancik grains have lower values, and in other half Cammeo have bigger values. I will describe Area and Extent features.

For Area, as we can see Osmancik grains have lowest values. Starting from the average values, Cammeo is “winning” as it has more grains. Osmancik has lower outlier, and Cammeo has a higher outlier. Overall, Cammeo grains are bigger than Osmancik ones.

For Extent, we can see that there is not such a significant difference between two classes. Both classes have the same lower and higher outliers. Even though, Osmancik has many grains with high extent value, Cammeo has more.

Distribution for the features of interest:

Major Axis Length distribution:

A picture containing plot, diagram, screenshot, line

Description automatically generated

As we can see average value (mean) for major axis length is 0.46145 and standard deviation is 0.186103. From standard deviation, we can understand that 68% of grains’ major axis lengths are between 0,278042 and 0.650248. Overall, it looks like normal distribution. Main data is between 0.2 and 0.8 values with lower outlier on 0 and higher outlier on 1.

Minor Axis Length distribution:

A diagram of a normal distribution with Ryugyong Hotel in the background

Description automatically generated with low confidence

As we can see average value (mean) for major axis length is 0.557828 and standard deviation is 0.119331. From standard deviation, we can understand that 68% of grains’ major axis lengths are between 0,358497 and 0.677159. Overall, it looks like normal distribution. Main data is between 0.4 and 0.7 values with lower outlier on 0 and higher outlier on 1.

Conclusion for the graphs:

Even though there is more grains of Osmancik class(2180 out of 3810) dataset is balanced. Cammeo sort has larger values overall than Osmancik. Data groupings are not that far from each other, but the difference is noticeable.

Statistical Calculations:

A screenshot of a graph

Description automatically generated with low confidence

Mean shows the average value for each column, so Eccentricity feature has biggest mean, as most values are high. Mode is most frequent value, so Eccentricity again has the highest mode of 0.871757, and in Convex Area there is the lowest mode of 0.299666. Median shows the middle value which is mostly around 0.50. Dispersion, overall, for all features is low which shows that values are close to the central value. For Eccentricity, its value is 0.0234701 which means that data is not spread out. In conclusion, dataset is balanced and good for predictions.

Part II – Unsupervised learning.

K-means algorithm

There are a few hyperparameters for K-means algorithm in Orange tool:

Number of clusters which lets you to choose how many clusters algorithm uses on data. By changing this hyperparameter it is possible to find best value for number of clusters, as if you choose small number. algorithm will not use some clusters and the performance will be bad. Same for choosing high number, which will make many small, insignificant clusters.

Preprocessing lets you normalize numeric values in dataset between 0 and 1.

Initialization hyperparameter has two choices of method: KMeans++ and Random. KMeans++ method selects the first clusters that are far away from each other. Random method selects the first clusters randomly.

Re-runs parameter lets you to choose how many times the algorithm will be executed to get more precise clustering results.

Maximum iterations lets you choose the number of iterations for algorithm before finishing.

Experiments:

I tried to perform k-Means algorithm with and without some features, overall I received best Silhouette scores with excluding Eccentricity and Extent features (values were between 0 and 1 even before normalizing). The Silhouette score I received with 2 clusters and 300 maximum iterations was 0.519. Changing initialization method to random, adding maximum iterations did not give better result. Considering that Silhouette score of 0.519 is closer to 1, we can say that separation is good.

A screenshot of a computer

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated with low confidence

Algorithm showed best separability based on Convex Area and Perimeter.

Hierarchical Clustering

There is one important hyperparameter: Linkage

It lets you choose which Linkage method should algorithm perform. Single linkage shows minimum distance between data of clusters, complete shows the maximum distance, and average linkage shows average distance.

Experiments:

A screenshot of a computer

Description automatically generated with low confidence

By moving cut-off line I have not experienced big differences. But on the bottom Osmancik class was better predicted than Cammeo on the top(there were too many Osmancik classes mixed with them). In my opinion, by moving line right and selecting lower number of clusters, the performance will be better.

Top of the dendrogram:

A screenshot of a computer

Description automatically generated with medium confidence

Bottom of dendrogram:

A picture containing text, screenshot, rectangle, diagram

Description automatically generated

Conclusions:

In my opinion, K-means algorithm performed much better in this situation, prediction categorical values more precisely than hierarchical clustering. In conclusion, hierarchical clustering did not perform that well in my dataset, incorrectly predicting classes.

Part III – Supervised learning.

Logistic Regression- I chose this algorithm because it is used specifically for classifying. This algorithm is used classification specifically. It can predict the probability of one of classes based on all the data given. Taking into account that my dataset is balanced, and two classes of rice have obvious differences, this algorithm can predict classes accurately.

kNN Algorithm (k-Nearest Neighbors) – I chose this algorithm because it shows reliable performance in classification by predicting precisely. It classifies data points based on how the neighbor of this data point is classified. Also, it works with complex data much better than other algorithms. Even if there is no much relationship between classifying attribute and other numeric columns it can predict with high accuracy.

Hyperparameters

**Logistic Regression** has two important hyperparameters: Regularization Type and Strength. Also it has Balance Class Distribution.

Regularization type: Ridge and Lasso. Ridge regression decreases coefficients and helps to minimize model complexity. Lasso regression decreases over-fitting and helps to select feature.

Strength – this hyperparameter controls overfitting by C value. A smaller C value makes model simple but strengthens regularization, while higher C value reduces regularization and makes model more complex.

Balance Class Distribution – this hyperparameters lets you make program to control class weights.

**kNN Algorithm** has one important hyperparameter: Number of Neighbors.

Number of neighbors – this hyperparameter lets you choose the number of neighbors or, in other words, close data points to predict class or new point.

Metric – lets you choose distance metrics.

Weight – lets you choose the type of weight which algorithm will consider. Uniform will consider all neighbors equally. But “By distance” will consider closest points more than far ones.

**Neural Network** algorithm has a few important hyperparameters.

Neurons in hidden layers – lets you choose the number of neurons in each layer and number of layers in network specifying it by comma.

Activation – lets you choose the type of activation function in a layer of network.

Regularization – lets you choose how much regularization will be used by algorithm on weights.

Maximum number of iterations – how many iterations will algorithm use before finishing.

Information on test/training datasets

A screenshot of a computer

Description automatically generated with medium confidence

For this case I selected 70% (2667 objects) of data objects to be training and remaining 30%(1143) of data will be test data.

For training data there is 1149 Cammeo grains of rice, and 1518 Osmancik grains of rice.

For testing data there is 481 Cammeo grains of rice, and 662 Osmancik grains of rice.

Experiments:

Logistic Regression

|  |  |  |  |
| --- | --- | --- | --- |
| Regularization | Strength | Correctly predicted(Cammeo | Correctly predicted(Osmancik) |
| L2 | 1 | 92.8% | 92.8% |
| L2 | 0.400 | 92.9% | 92.1% |
| L2 | 4 | 92.4% | 93.6% |
| L2 | 2 | 92.9% | 93% |

Table represents experiments on training data(cross-validation) for logistic regression.

kNN Algorithm

|  |  |  |
| --- | --- | --- |
| Number of neighbors | Correctly predicted(Cammeo) | Correctly predicted(Osmancik) |
| 10 | 90.6% | 93.2% |
| 5 | 91.7% | 93.2% |
| 7 | 91.6% | 93.1% |
| 3 | 90.8% | 92.9% |

Table represents experiments on training data(cross-validation) for kNN Algorithm.

Neural Network

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Neurals in hidden layers | Activation | Solver | Regularization | Max num of iterations | Correctly predicted(Cammeo | Correctly predicted(Osmancik) |
| 100,100 | Logistic | SGD | 0.1 | 200 | 98.3% | 74.8% |
| 100, | Logistic | SGD | 0.5 | 200 | 98.9% | 70.7% |
| 100,50 | Logistic | SGD | 0.3 | 200 | 97.0% | 81.0% |
| 100,50 | Logistic | SGD | 0.4 | 200 | 97.1% | 81.2% |
| 100,100 | Logistic | SGD | 0.2 | 200 | 98.6% | 73.9% |

Table represents experiments on training data(cross-validation) for Neural Network Algorithm.

Conclusions:

Logistic regression shows almost the same performance as the kNN Algorithm. But interestingly, neural network showed satisfactory results only Cammeo class of rice, prediction Osmancik class is much worse with difference in 16 percents.

All experiments have been done using 3 different algorithms and best models have been chosen.

Testing/training comparison

Hyperparameters chosen for Logistic regression

A screenshot of a computer

Description automatically generated with medium confidence

Training:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with low confidence

Test:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with medium confidence

Hyperparameters chosen for kNN Algorithm.

A screenshot of a computer

Description automatically generated

Training:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with low confidence

Testing:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with low confidence

Hyperparameters chosen for Neural Network.

A screenshot of a computer

Description automatically generated

Training:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with medium confidence

Test:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with low confidence

In conclusion, as we can see algorithms have better performance with training data, but there is not that much difference. All models working good enough.

References:

<https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b>

<https://archive.ics.uci.edu/ml/datasets/Rice+%28Cammeo+and+Osmancik%29>

<https://orangedatamining.com/widget-catalog/>

<https://www.simplilearn.com/tutorials/statistics-tutorial/what-is-probability-distribution>

<https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>

<https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point>.