Zeszyty Naukowe WWSI, No 30, Vol. 18, 2024, s. 7-26
DOI: 10.26348/znwwsi.30.7

Python Machine Learning. Dry Beans Classification Case

Grzegorz Słowiński*

Warsaw School of Computer Science, Warsaw, Poland

Abstract

A dataset containing over 13k samples of dry beans geometric features was analyzed using machine learning (ML) and deep learning (DL) techniques with the goal to automatically classify the bean species. Performance in terms of accuracy, train and test time was analyzed. First the original dataset was reduced to eliminate redundant features (too strongly correlated and echoing others). Then the dataset was visualized and analyzed with a few shallow learning techniques and simple artificial neural network. Cross validation was used to check the learning process repeatability. Influence of data preparation (dimension reduction) on shallow learning techniques were observed. In case of Multilayer Perceptron 3 activation functions were tried: ReLu, ELU and sigmoid. Random Forest appeared to be the best model for dry beans classification task reaching average accuracy reaching 92.61% with reasonable train and test times.

Keywords – machine learning, deep learning, data dimension reduction, activation function

* E-mail: gslowinski@ms.wwsi.edu.pl

Manuscript received December 14, 2023.

1. Introduction

Since 2010, there has been an intensive development of artificial intelligence, including machine learning and deep learning (artificial neural networks). There is a shortage of specialists in a broad field of computer science and engineering. A similar deficit of specialists exists in the area of the artificial intelligence. The prevalence of demand over supply causes that the salaries of data scientists and deep learning engineers are significantly higher than the average in the economy [1].

Two programming languages have gained particular popularity in this area: R and Python [2]. This work tries to provide a (necessarily limited) introduction to the subject of the artificial intelligence and machine learning, using the case of dry beans classification.

1.1. Terms: Artificial Intelligence, Machine Learning and Deep Machine Learning

The term Artificial Intelligence was probably created and used for the first time in 1955 by John McCarthy [3] in submission of a summer research project. The purpose of the project was to check the hypotheses as to whether human learning or any other manifestation of intelligence can be described by the rules so precisely that it will be possible to create a machine simulating this process. Artificial intelligence (AI) is a broad and blurred concept. There is no precise definition of this term. On the one hand, the term artificial intelligence is sometimes used to describe relatively simple algorithms that are assumed to imitate human thinking and acting, on the other hand, it addresses not yet existing a self-aware IT solution [4].

The Oxford Reference defines artificial intelligence as follows: “The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages” [5]. It seems that in practice the term artificial intelligence also refers to an IT solution that simulates (imitates) human thinking. The literature distinguishes certain degrees of AI development: narrow AI, general AI, and artificial superintelligence [6]. It should not be forgotten that the term AI or the adjective intelligent itself is sometimes used in exaggeration, for the purpose of marketing, to use positive connotation of the word “intelligent” to raise the

importance of the offered product or service in the eyes of a potential customer. Hence, one can meet intelligent (by name): houses, cars, household appliances and other solutions in which the element of "intelligence" may be debatable.

1.2. Machine Learning

Machine Learning is part of AI. Dependence between AI, machine learning (ML) and deep learning (DL), is shown in Figure 1. Machine learning is a subset of AI techniques. Similarly deep learning is a particular subset of techniques in the ML field.

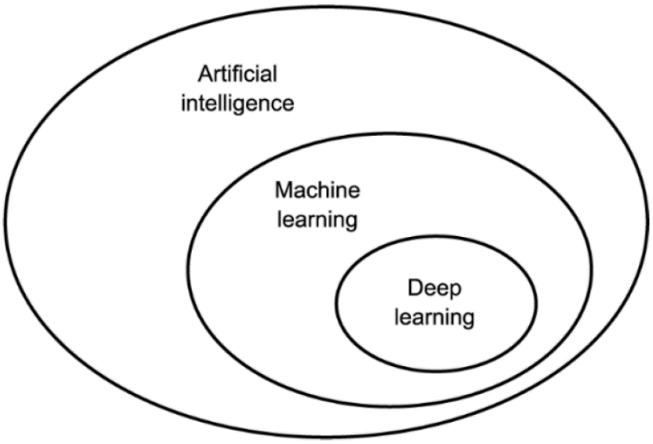

Figure 1. Relation between Artificial Intelligence, Machine Learning and Deep Learning [7]

There is also the term "shallow learning" which covers all of the machine learning techniques that are not deep learning.

Machine learning includes a number of techniques that can vary quite a bit. The question is how to judge whether a technique qualifies as machine learning. The main difference between machine learning and the classic way of solving a problem (classical program) is shown in Figure 2.

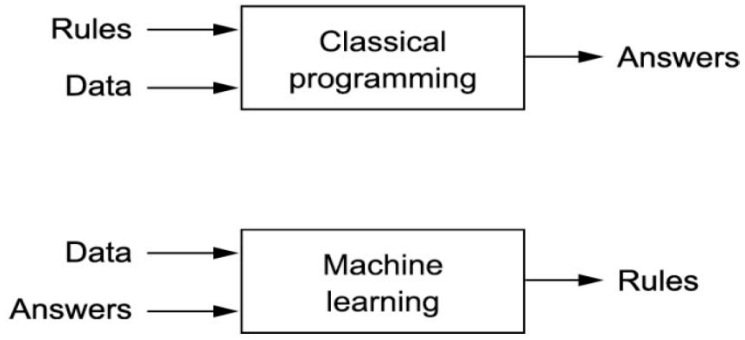

Figure 2. Classical Programming vs Machine Learning [7]

In the classic approach to solving decision problems we try to get to know and describe the rules of making decisions and receiving answers. Based on the rules, an algorithm is created that usually becomes implemented in the form of a computer program.

Many problems can be successfully solved by applying the approach described above. However, there are many issues where it is not clear what rules must be used to get the correct answer. Not infrequently, though not always trivial to humans. For example, the classification of whether a given photo contains a likeness of a dog or a cat is trivial to a human (this issue is a popular exercise during learning how to build UM models, a collection of photos of cats and dogs can be found in [8]). However, establishing the rules to be followed in distinguishing a dog from a cat and expressing them with a strict algorithm is extremely difficult, if at all possible.

1.3. Shallow learning versus deep learning

The term Deep Learning is used frequently, but less frequently explained. It is usually associated with artificial neural networks (ANN) that have at least one hidden layer. According to F. Cholett (the creator of the Keras framework) the adjective “deep”, including the context, means multi-layer data representation. Each layer of an ANN model creates a new representation (transformation) of the data input. “Deep” in this context means a large number of the following transformations. According to Chollet, one should not interpret this as a suggestion that the ANN technique achieves some deeper understanding of the analyzed subject issues [7]. The basic differences between shallow learning and deep learning are listed in Table 1.

Table 1. Shallow and deep learning comparison

	Shallow Learning	**Deep Learning**
Dataset size	Small	Large
Training time	Short	Long
Hardware	CPU	GPU, TPU

Basically, shallow learning should be used when we have less data and the data is simpler (fewer numbers describe a given sample). It requires less computing power. Shallow learning models have fewer degrees of freedom and fewer coefficients that are fine-tuned during the learning process.

Deep Learning neural network models have more factors that need to be fine-tuned during the process of training. Due to the nature of the process, that can be done in some part in parallel, it is advantageous to use graphics cards (GPU) in calculations, which speeds up the computation. Due to the large number of degrees of freedom ANN is quite easy to trigger the overfitting phenomenon which is manifested by a god operation of the model on the training data, but not on new ones, previously unseen data.

1.4. Basic problems solved by machine learning

Basic problems solved by learning come down to the issue of classification and regression. The issue of classification is assigning a given object to one from several categories or classes. Examples of such issues include recognizing the variety of irises by the geometric structure of the flower, predicting whether a given passenger survived the Titanic disaster, handwritten number recognition.

The second type of problem is regression. This task is matching an appropriate response for a given sample over a continuous range of values. Examples of issues are: determining the strength of concrete hardness on composition and seasoning time, house appraisal, wine quality assessment.

Some issues depending on the ways of formulation can be presented as both regression and classification. For example, in the case of the prediction of concrete strength, we can formulate the problem on regression method and develop a model that will predict the endurance a specific number, e.g. in MPa, or be satisfied with

determining whether a given concrete will be: strong, average, weak and formulate the problem in the form of classification.

1.5. Computing environment

The computing environment used for this work consisted of: Jupyter Notebook as the programming IDE. Python and its frameworks for: data manipulation – Numpy, Pandas, visualization – Matplotlib and Seaborn and machine learning – Scikit Learn and Keras (on Tensorflow backend). Jupyter notebooks containing the codes of the machine learning models developed are available in [9].

2. Dataset preliminary characterization and visualization

The dataset used in this work has been published in the UCI machine learning repository [10]. The set was originally described in work [11]. The set consists of over 13 k samples of dry beans of 7 various species. Classification of dry beans is of some economic importance. Manual classification is labor intensive, etc. The dry beans were photographed and their geometry was measured via computer vision techniques in [11]. Then the set was analyzed via several machine learning (or data science) and deep learning (or artificial neural network) techniques. The overall accuracy obtained was 87.92%–93.13%, depending on technique used.

The dataset under study consists of 13611 samples. A sample amounts to 16 geometrical features and a label identifying the species of the bean. The species are as follows: Barbunya, Bombay, Cali, Dermason, Horoz, Seker, and Sira. The features are: Area, Perimeter, Major Axis Length, Minor Axis Length, Aspect Ratio, Eccentricity, Convex Area, Equiv Diameter, Extent, Solidity, Roundness, Compactness, Shape Factor 1, Shape Factor 2, Shape Factor 3, and Shape Factor 4. A detailed explanation how the features were calculated is presented in [11]. The geometrical data carry no information about the bean color. From the practical point of view it is unfortunate, as different dry bean species tend to vary in color.

Correlation analysis (see Table 1) has shown that several of the features are strongly (positively or negatively) correlated. This is due to the fact that basically all of them are kind of geometric measures. Generally, strongly correlated (over 0.9) features bring little extra information, so their elimination should reduce

computational complexity (speed up training) with little if any loss in classification accuracy.

Principal Component Analysis is an often-used method of dimensionality reduction. The method is an affine transformation that uses data translation, rotation and uniform scaling. The transformation of data axes into PCA axes is carried in a way that maximizes variation of the 1st axis, then the variation of the 2nd axis and so on. The higher axes contain less (often very little) variance than lower axes. Omitting them produces lower-dimensional projection of data preserving as much variance as possible. PCA can be also very useful for obtaining 2D visualization of data, as show in Figure 3. The drawback of the PCA is that the resulting dimensions have no clear interpretation. The plot shows that the Bombay species is trivial to classify as its beans are significantly bigger than others. the classification of other species seems to be much more difficult, and we can expect more errors. The correlations between pairs of the selected features are listed in Table 2.

Table 2. Correlation between selected beans features

	Major Axis Length	**Minor Axis Length**	**Aspect Ratio**	**Extent**	**Solidity**	**Roundness**	**Shape Factor 2**	**Shape Factor 4**
Major Axis Length	1	0.826	0.550	-0.078	-0.284	-0.596	-0.859	-0.483
Minor Axis Length	0.826	1	-0.009	0.146	-0.156	-0.210	-0.471	-0.264
Aspect Ratio	0.550	-0.009	1	-0.370	-0.268	-0.767	-0.838	-0.449
Extent	-0.078	0.146	-0.370	1	0.191	0.344	0.238	0.149
Solidity	-0.284	-0.156	-0.268	0.191	1	0.607	0.344	0.702
Roundness	-0.596	-0.210	-0.767	0.344	0.607	1	0.783	0.472
Shape Factor 2	-0.859	-0.471	-0.838	0.238	0.344	0.783	1	0.530
Shape Factor 4	-0.483	-0.264	-0.449	0.149	0.702	0.472	0.530	1

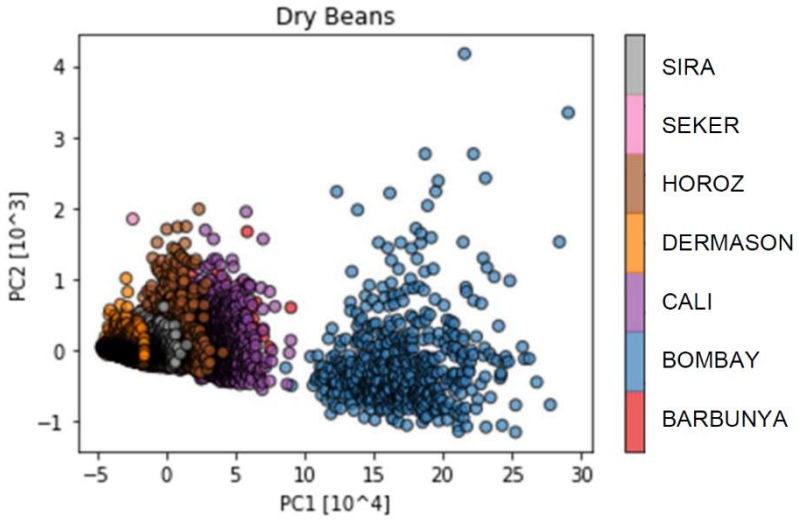

Figure 3. Dry beans dataset representation after PCA transformation

Thus, in this work it was decided to arbitrary limit the set of features to 8, to avoid strongly corelated features. These are: Major Axis Length, Minor Axis Length, Aspect Ratio, Extent, Solidity, Roundness, Shape Factor 2, Shape Factor4, and to exclude: Area, Perimeter, Eccentricity, Convex Area, Equiv Diameter, Compactness, Shape Factor 1, Shape Factor 3. The issue of high correlations among some features was not addressed in [11].

In addition to it, data visualization obtained with help of PCA means, so called pairplot is often applied. Pairplot is a composition of several smaller plots. It presents the plots of different feature pairs correlation. On the main diagonal it presents distribution of features in data classes. Figure 4 presents visualization of the data (3 selected low corelated features) done by pair-plot.

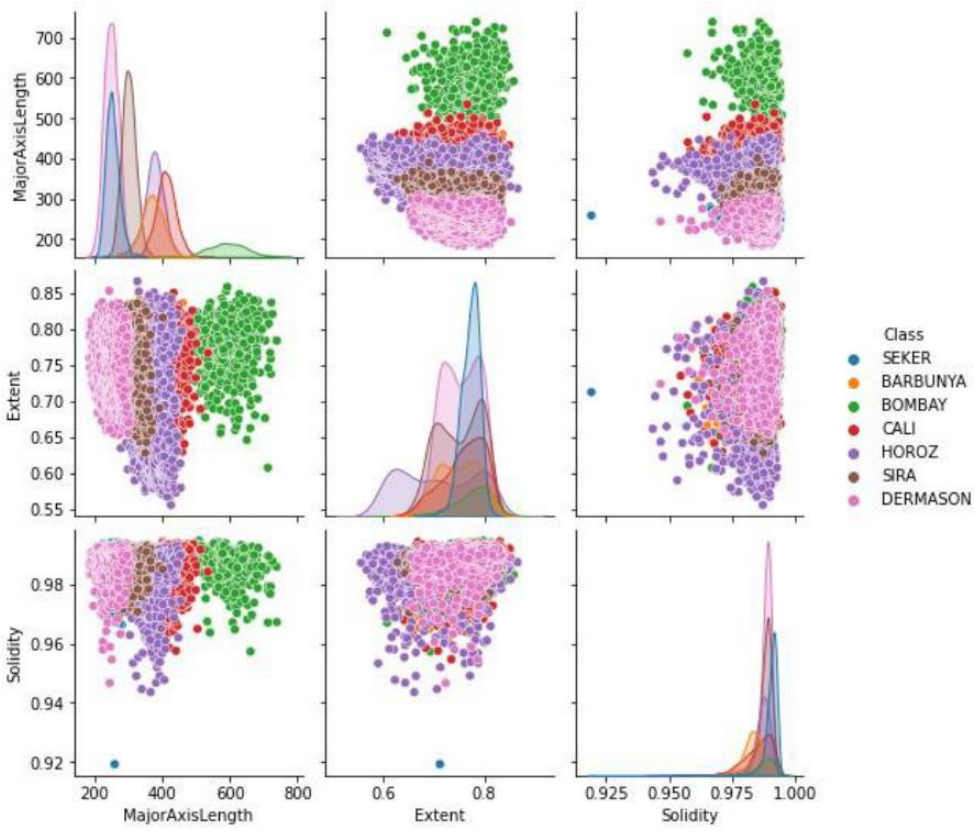

Figure 4. Pair-plot of 3 selected (low corelated) bean features

3. Shallow Machine Learning techniques used and results

In this work the following shallow machine techniques were used: Gaussian Naive Bayes Classifier, Support Vector Classifier, Decision Tree, Random Forest and Voting Classifier. In addition an Artificial Neural Network (Multilayer Perceptron or MLP) was applied, as discussed in a later section. The full dataset was divided into the training and test subsets. 80% of samples were used for training and 20% for testing. Division of all available samples into the training and test subsets is crucial for a correct methodology. The aim of all ML or DL methods is to achieve a generalization ability. Thus it is important to check the accuracy of classifying new samples, ones that have not been used during training. Otherwise, there is a very

serious risk that the model will suffer from overfitting. Overfitted models perform very well on the training data but much worse on new data. Overfitting (as one of the most important issues in ML) is widely discussed in ML handbooks [7] [12] [13].

3.1. Cross Validation

Generally, it can be observed that ML results depend to some extend on dataset division into train and test subsets. Cross validation is a technique moderating this effect. The ML process is being repeated several times and different parts of the data are being used as test subset. For example, if the 5-fold CV test is done, each time different 20% of data is used as a test subset. This gives better idea regarding the expected performance range of the ML model. A very important (but easy to forget) issue when performing CV tests is to shuffle the data to make sure that each class is uniformly distributed in the dataset. In this experiment 5-fold cross validation was done.

3.2. Naive Bayes classifier

Naive Bayes models are based on Bayes's theorem. They are extremely fast and simple, but on the other hand, their performance is usually limited. They can be used as a baseline for classification problems (see [4], p. 382). Here a Gaussian Naive Bayes (GNB) Classifier were tried. Gaussian means that the model “assumes” that each feature in the class has gaussian distribution (which is usually false).

3.3. Support Vector Classifier

Support vector machines (SVM), which can be used as regressors or classifiers, are considered a very powerful and flexible algorithms. On the other hand, they may need a lot of computing power (see [12] p.405). The SVM principle is to partition the classes by “drawing a line” (or plane) in a way that maximizes the margin between classes. As straight lines (or planes) do not usually produce the best solution, support vector classifier (SVC) can apply different kernels (polynomial, radial and others). SVC is wider explained in [12] [13]. Here SVC with 3_{rd}degree polynomial kernel was applied.

3.4. Decision Tree

A decision tree (DT) belongs to the class of so-called non-parametric algorithms. The term nonparametric can be misleading. In fact, a decision tree has parameters, but their number is not constant. During the learning phase, a decision tree tries to find the best questions partitioning the dataset to reduce information impurity (the measure is the Gini index or information entropy). The great advantage of decision trees is that they are extremely intuitive. On the other hand, a decision tree has no limited degrees of freedom, so it is easy to overfit (if the user is not aware of that). The splits made by a decision tree are always orthogonal (made on one feature at a time), so the decision tree is very sensitive to data rotation (see [5], p. 188). It was decided to limit the depth of the decision tree (number of decisions) to 5 and the number of leaves to 16, to get a decision tree that has size similar to DT obtained in [1].

Obtained tree structure is presented in figure 5. Unfortunately is quite difficult to visualize a DT in a limited space keeping the readability. As a compromise, it was allowed for some nodes and leaves to overlap.

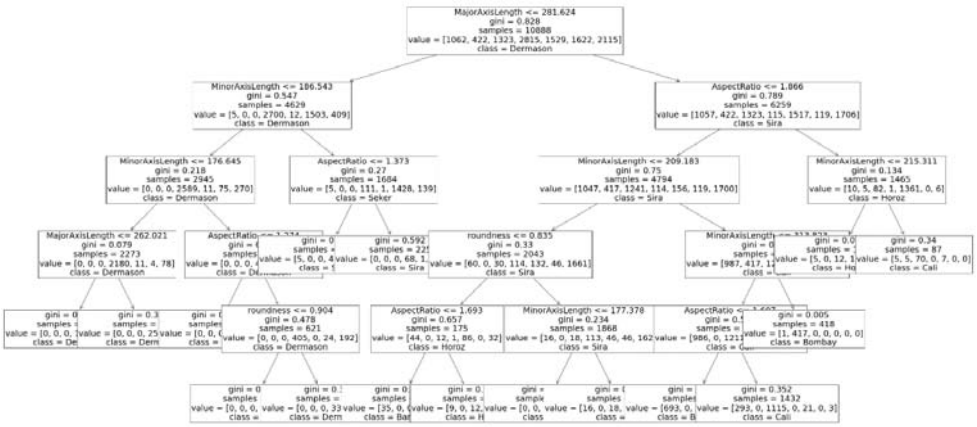

Figure 5. Visualization of the decision tree obtained during the experiment

3.5. Random Forest

The random forest (RF) idea is as follows: take many decision trees (employing some randomness, so the trees differ) and let them vote. So, the classification decision taken by RF is a decision taken by the majority of decision trees in a random set of

trees. Usually, a random forest performs better than a single decision tree. However, a random forest is considered "black-box" model being very hard to interpret. A RF of 100 decision trees (this is model's default value) was created. No restrictions on tree size imposed. It seems that there is some point size of a single tree, as very big decision trees can be (1) difficult in interpretation and (2) sensitive to noise in data. In case of RF, which is a "black box" model, the noise effect should be somehow "averaged" by the fact that there are tenths or hundreds of trees in the forest.

3.6. Voting Classifier

The idea of "voting", which, by default, is used in random forests, can be applied to any classifiers. There are 2 main ways of voting: "hard" (straightforward, direct voting) and "soft" (the votes are weighted depending on how confident the classifier is with its choice). Like in the case of a random forest, there is a good chance that the voting result will be more accurate than for any particular classifier.

Table 3. Summary of classifier models applied

No	Classifier type	Parameters
1	Gaussian Naive Bayes	
2	Support Vector	3rd degree polynomial kernel, $C = 10^5$
3	Decision Tree	Depth up to 5 levels, 16 leaves maximum.
4	Random Forest	Made of 100 DT, no limit on single tree
5	Voting Classifier	Made of models 1, 2 and 4

3.7. Cross Validation Results

Results of the cross validation experiment performed are summarized in Table 4. One can see that the RF classifier achieved the best accuracy. Performance of GNB is surprisingly high. Usually, this method is applied for preliminary research (just to check if there is any coincidence in the data) because it is fast, while not being very accurate. Here it performance was only slightly weaker than SVC. SVC appeared to be the second most accurate model. However, its accuracy is by over 1.5% point weaker than RF and only about 0.13% point better than GNB. Considering how time-

consuming SVC is (it takes about 10 times more time to run than RFC and 1500 times more than GNB) this is rather disappointing. The other important time feature is the score time. This is the time of the test set classification. For business, real-time models it could be even more important than training time as we train the model once and then it does its classification work for long. Here SV is the slowest one (167 ms), RF in the middle (37 ms), DT and GNB are much faster (2.4 and 3 ms).

The hard voting classifier (VC) was implemented using 3 classifiers described above: GNB, SVC and RF. VC is really time consuming. Its training (17.34 s) and score (213 ms) are the longest as they sum up all the classifiers it is consists of. Its accuracy results are about 0.66% point worse than for the RF. This gives us a clue that voting should be used carefully and preferably with models exhibiting similar performance; otherwise "naive" models can outvote "smart" models. It seems that this flaw of democracy does not only apply to human societies but is more universal in nature.

Decision tree is the least accurate model. Generally, its main advantage is "white box" character and the fact that the model is fast. In fact it appeared that it is even faster than GNB. If information how decision is taken is important – this can be an advantage. However, it is paid by a significantly (over 3% points) lower accuracy.
The decision which model to choose for business application is deliberate. If accuracy is the most important criterion – RF should be considered. However, if the classification speed is crucial (and one can pay for it with a lower accuracy), then GNB may be better.

Table 4. Summary of 5-fold cross validation experiment

Classifier type	**Accuracies (min; max; avr) [%]**	**Average training time [s]**	**Average score time [ms]**
Gaussian Naive Bayes	90.45; 91.55; 90.94	0.01	3.0
Support Vector	90.41; 91.77; 91.07	16.01	166.7
Decision Tree	87.17; 88.46; 87.78	0.04	2.4
Random Forest	92.10; 93.24; 92.61	1.26	37.2
Voting Classifier	91.37; 92.43; 91.95	17.34	213.1

3.7.1. Closer Look at Random Forest Results

Random Forest appeared to be the best model. It offers the best accuracy with reasonable training and test time. Accuracy is an overall measure of model quality. More detailed insight into results can be achieved by confusion matrix. Figure 5 shows the confusion matrix for classification of test set done by RF classifier.

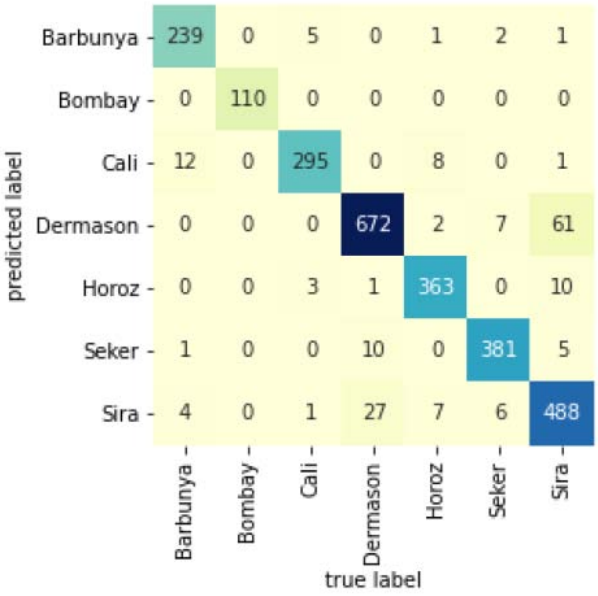

Figure 6. Test subset confusion matrix for the random forest classifier

The columns in the confusion matrix show real labels of beans and rows – model decisions. The matrix diagonal (when true label equals predicted label) shows the correct prediction. Other fields give insight into nature of false predictions. As one can see, Bombay beans are correctly classified all the time. The main source of mistakes are classes Dermason and Sira. As one can see 61 Sira beans were classified as Dermason and 27 Dermason beans as Sira. Totally there were 175 mistakes, so Dermason and Sira were responsible for about 50% of mistakes. Perhaps in business model one should consider classifying these beans to one class (Dermason & Sira) and then separate them using some other features not taken into account in the dataset. As can be visible from the photographs reproduced in [11] these species looks very similar.

3.7.2. Influence of dimension reduction on ML Performance

Two models: Support Vector Classifier and Random Forest were selected to check what is the influence of dimension reduction on model performance. The performance measures are accuracy and training time. In addition to the experiment performed above on the data manually reduced to 8 dimensions, the original 16-dimension dataset was tried, and the dataset modified by PCA means to 8 dimensions. The results are presented in table 5 below.

Table 5. Influence of different dataset prior-preparation on ML models performance

Classifier	Dataset version	Accuracies (min; max; avr) [%]	Average training ime [s]	Average score time [ms]
Support Vector	8 "manually selected" features	90.41; 91.77; 91.07	16.01	166.7
	Full features	90.50; 91.84; 91.00	26.84	247.9
	PCA reduced 8 features	80.34; 82.18; 81.15	14.49	376.0
Random Forest	8 "manually selected" features	92.10; 93.24; 92.61	1.26	37.2
	Full features	91.92; 92.76; 92.35	2.52	382
	PCA reduced 8 features	92.14; 93.57; 92.58	1.36	36.3

As can be seen data reduction (both "manual" and by PCA) decreases significantly training time for both SVC and RF. In case of RF it also decreases test time by about 90%. In case of SVC data reduction by PCA appears to be a "disaster". Accuracy dropped by about 10% points and score time is even longer than for full 16-features dataset. It is hard to explain why PCA transformation leads to such poor performance of SVC. However, this shows how sensitive models can be and how careful data scientist should be during data preparation.

4. Multilayer Perceptron Classifier

Besides Classification models presented above also multilayer perceptron or in other words small artificial neural network (ANN) was tried. For an artificial neural network,

the data needs additional treatment. First, the names of bean species were labelled with numbers and then these numbers 0-6 were coded as the so called “one-hot”.

The reason of using the “one-hot” encoding is well explained in [7] pp. 190-194 or [12] p. 376. The other operation is scaling, a standardization or normalization of the training data. The data (each feature) is centered around zero (by subtracting the average) and normalized (by dividing by the standard deviation). Standardization is said to ease the training process and tends to bring in improvement in performance [13] p. 72. The structure of the ANN is presented in Figure 6.

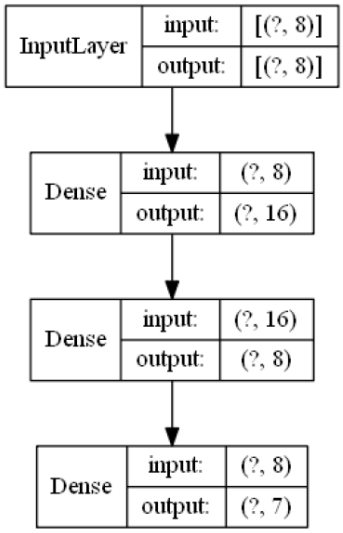

Figure 7. Structure of ANN used

Moreover, the data is being shared into 3 subsets: training, validation and test. The validation subset is used to monitor the network learning process. It is used to test the network performance after each epoch of training. This allows to see how smooth learning process is and to observe possible overtraining.

The ANN is built of input layer. The input layer is feed with beans dataset with 8 manually selected features (as described in point 3). Then 2 hidden layers are added with 16 and 8 neurons. The output layer has 7 neurons, one for each bean species. Softmax is the activation function for the output layer. The optimizer used was RMSprop, and the loss function was the categorical cross entropy. The validation set was 20% of the training set. The network was trained for 10 epochs. The plot of

training is presented in figure 7. Right column in the box describing each layer shows the dimension of the input/output data. The "?" sign denotes that the first dimension of data (number of samples) can vary. Three variants of the ANN were tried. The versions had different activation function in the hidden layers. Rectified linear unit (ReLU), exponential linear unit (ELU) and sigmoid were tried.

To check the repeatability of the process, the training was repeated 5 times with different data split into train, validation and test subsets for each version of ANN. Results are summarized in Table 6. Training process curves are presented in Figure 7.

Table 6. Performance of ANN classifier

Activation	**ReLU**	**ELU**	**Sigmoid**
Training epochs	10	10	30
Average training time	1.83 s	2.19 s	4.61 s
Average test time	55.5 ms	55.3 ms	55.2 ms
Accuracy (min; max; avr) [%]	91.41; 93.68; 92.20	91.74; 93.13; 92.16	91.22; 92.54; 91.70

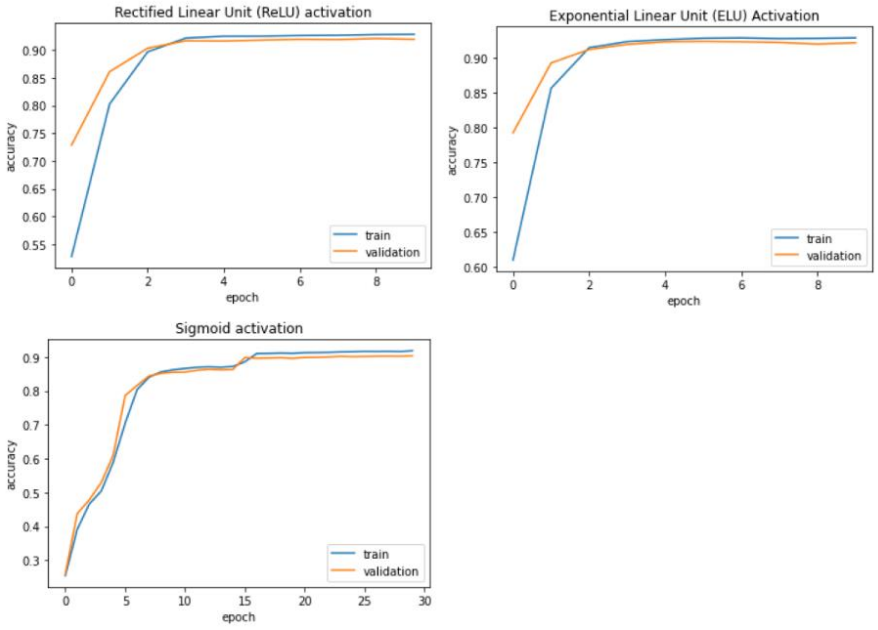

Figure 8. Training process of ANN with different activation functions in hidden layers

Both ReLU and ELU versions were trained for 10 epochs, but for sigmoid activation 10 epochs were too few and 30 epochs were used, which also increased the training time. The training process in case of ReLU and ELU can be seen as smooth. ELU version trains a bit longer than ReLU. This is probably connected with the fact that it takes longer to compute exponential function than a combination of linear and constant (ReLU). In the case of sigmoid activation, the training process is significantly slower as it needs more epochs. The training process reaches plateau at about 7_{th} epoch and next improvement in accuracy appears around 15 epoch.

In term of accuracies ReLU and ELU also outperform sigmoid by about 0.45–0.5 % point. However both models are a less accurate than Random Forrest by about 0.40–0.45 % point.

5. Conclusions

The dry beans dataset was analyzed with shallow learning and ANN models. Different models can vary significantly in terms of computing time for training and testing. The accuracy of all models was in the range 87-92%. Random Forest appeared to be the most accurate model, reaching 92.61% average accuracy with moderate training and test times. Support Vector Classifier is the slowest model in this task. Its accuracy was also rather moderate (91.07%).

The influence of data pre-processing was checked for Random Forest and Support Vector Classifiers. Data dimensionality reduction had positive influence on training time. In case of RF dimension reduction from 16 to 8 (either manually or by PCA) shortened the training time and improved the accuracy.

In the case of support vector classifier manual reduction improved the model. However the reduction done by PCA damaged the model by drastically decreasing the accuracy to 81% and significantly increasing the score time. This shows that such methods should be used with care.

The Multilayer Perceptron experiment showed about 0.45% lower accuracy than Random Forest. The models with ReLU and ELU activation functions showed similar performance and accuracy around 92.2 %. Sigmoid activation caused longer less stable training process with lower accuracy around 91.7%.

When applying such model it should be remembered that model needs periodical update. Beans geometry can change in different seasons or if they come from

different location. Bean shape and size depends also on weather condition (that can vary from season to season), soil condition and etc., not only on its species. If this not taken into account and no periodical retraining, even the most sophisticated model could fail.

References

[1] B. Schroeder. (2021) *The Data Analytics Profession And Employment Is Exploding – Three Trends That Matter*. [Online]. https://www.forbes.com/sites/bernhardschroeder/2021/06/11/the-data-analytics-profession-and-employment-is-exploding-three-trends-that-matter/?sh=589c02c33f81.

[2] *9 Top Programming Languages for Data Science*. [Online]. https://www.edx.org/resources/9-top-programming-languages-for-data-science.

[3] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. (1955) *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. [Online]. http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf.

[4] T. Zalewski, "Definicja sztucznej inteligencji," [In:] *Prawo sztucznej inteligencji*, L. Lai and M. Świerczyński (Eds.). Warszawa: C.H. Beck, 2020.

[5] Artificial Intelligence [in:] *Oxford Reference*. [Online]. https://www.oxfordreference.com/display/10.1093/oi/authority.20110803095426960.

[6] A. Kaplan and M. Haenlein, "Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence," *Business Horizons*, Vol. 62, No. 1, pp. 15-25, 2019. [Online]. https://doi.org/10.1016/j.bushor.2018.08.004.

[7] F. Chollet, *Deep Learning with Python*, 2nd ed. Shelter Island, NY: Manning Publications Co., 2021. [Online]. https://www.academia.edu/download/62075089/Francois_Chollet-Deep_Learning_with_Python-Manning_2018_120200212-91819-unzepp.pdf.

[8] W. Cukierski. (2013) *Dogs vs. Cats*. [Online]. Kaggle. https://kaggle.com/competitions/dogs-vs-cats.

[9] [Online]. https://github.com/stophouse/DryBeansPostProceedings.

[10] *Dry Bean Dataset*. (2020) UCI Machine Learning Repository. [Online]. https://doi.org/10.24432/C50S4B.

[11] M. Koklu and I. A. Ozkan, "Multiclass classification of dry beans using computer vision and machine learning techniques," *Computers and Electronics in Agriculture*, Vol. 174, p. 105507, 2020. [Online]. https://doi.org/10.1016/j.compag.2020.105507.

[12] J. VanderPlas, *Python Data Science Handbook*. Sebastopol, CA: O'Reilly Media, Inc., 2017. [Online]. https://www.academia.edu/download/62974298/pythondatasciencehandbook20200415-66956-1noqwv2.pdf.

[13] A. Géron, *Hands on Machine Learning with Scikit Learn*, Keras, 2nd ed. Sebastopol, CA: O'Reilly Media, 2019. [Online]. http://14.139.161.31/OddSem-0822-1122/Hands-On_Machine_Learning_with_Scikit-Learn-Keras-and-TensorFlow-2nd-Edition-Aurelien-Geron.pdf.