Purpose

The purpose of this document was to record personal supplementary notes and the results of following a starter tutorial for machine learning found at <https://machinelearningmastery.com/machine-learning-in-python-step-by-step/>. For the tutorial the Iris flower dataset, <http://archive.ics.uci.edu/ml/datasets/Iris>, was used to build and train the model.

Also, for completeness, an ongoing compilation of a glossary can be found at <https://github.com/JonathanC13/Machine-Learning-Reference-Docs/blob/master/Glossary.docx> and of algorithms used up to date can be found at <https://github.com/JonathanC13/Machine-Learning-Reference-Docs/blob/master/Algorithms.docx>.

This machine learning problem’s model was offline, used supervised learning, and had classification outputs. The dataset requires no categorical conversions or handling of null data.

Procedure

The workflow of building an accurate machine learning model is an iterative process where the data used to train the model dictates how the learning algorithms parameters should be tweaked to impact to model. The goal is that each iteration of the process results in a higher accuracy for its predictions on the unseen training data. In this tutorial it will be an end to end project to practice one iteration of the machine learning process.

Following several well-known steps:

1. Define/understand the problem: Identify relevant data sources since increasing the amount of relevant data will improve the system. Depending on the data that is being used to build a model it may require domain experience. There are a few model properties that can be determined like:

* The model needs to be online or offline.
* The type of learning, supervised, semi-supervised, unsupervised, or reinforced, that should be used to fit the data
* The model is either classification or regression, where it is based on the expected outputs.

In this problem the model is offline since the model was trained on a dataset and deployed. It used supervised learning since in the dataset each feature vector had a target attribute. The model can be defined as a classification model since the outputs are discrete values.

X Attributes: ['sepal-length', 'sepal-width', 'petal-length', 'petal-width'];

Y, target, Attribute: [‘class’];

1. Prepare data:

* Somehow need to validate the data or else if the data is inaccurate the model will fail.
* Real world data may not contain the desired format to be used with certain libraries, so encoding needs to be done, for example categorical data to numerical because many algorithms require numerical inputs and using categorical data can have undesired impact of the model’s fitting.
* Also, real world data is raw in the sense that it can have incomplete values for attributes in some instances, it could be represented by a multitude of symbols, it helps when the dataset provider provides the symbol(s) that indicate incomplete data. When there is incomplete data those values need to be filled using various methods because simply removing instances that have missing values will result in reduction of the amount of data to use and those instances could still provide insight on potential patterns.

In this case, the dataset doesn’t have any categorical values except for the target attribute, which is fine, and the dataset is already cleaned of missing values.

1. Selecting the Algorithms:

* From the available dataset it is split into a portion for the training dataset to use for the training process and the remaining unique portion for the validation dataset for the testing process, it is important that this data is ‘unseen’ by the model.
* Depending on the observed or predicted relationships for the dataset, multiple algorithms can be effective for fitting the dataset. It is beneficial to train multiple models so they can be compared by way of metrics like accuracy. Common outcomes of the fitting can be good fit, underfit, or overfitting, this can be indicated with the model’s accuracy on the training data and then its accuracy on the testing data.
* Be warned when choosing algorithms to build models because in real world big data correlation doesn’t always mean causation. The nature of the relationship may require domain experience to determine which correlations should be weighed more.
* It is important that the dataset is valid somehow and/or have access to a very large dataset, so a normal distribution is more likely to present and data that seem like anomalies in a smaller dataset may actually present a pattern with a larger dataset.

In this project the training dataset was 80% of the original dataset and the testing dataset was the remaining 20%.

In this case it was observed that some attributes have a strong linear correlation and allows for a predictable relationship. Regression algorithms were used because it is the analysis of the relationships between datapoints, which also means that is assumes that correlation is directly related to causation therefore if the context of the data is not understood it can lead to inaccuracies. It is also noted that even from a layman perspective the relationships can be made sense of, for example the linear correlation of petal-length vs petal-width. Although, don’t fall into the trap of forcing a belief that a relationship makes sense.

* To cover the bases a good mix of machine learning algorithms are used to build different models:
* Logistic Regression (LR); linear
* Linear Discriminant Analysis (LDA); linear
* K-Nearest Neighbors (KNN); nonlinear
* Classification and Regression Trees (CART); nonlinear
* Gaussian Naive Bayes (NB); nonlinear
* Support Vector Machines (SVM); nonlinear

1. Train the Models:

* To train a model it depends on the type of data and the algorithm used. These factors determine whether the training process’s learning is supervised, semi-supervised, unsupervised, or reinforced.

In this case, supervised learning is used due to the data having a labeled attribute.

1. Evaluate:

* Evaluate the models’ accuracy on testing dataset during the testing process to determine the performance of the model.

1. Predict:

* Use the model to predict on new unseen data and access the performance to determine improvements that can be made on the model to improve accuracy.

1. Improve Results:

To be able to improve the results the model must be retrained with additional tuning of interface parameters that the programmer can control, since modifying the algorithm itself is not feasible. This tuning is called hyper-parameter tuning since the parameters are the outermost parameters that affect the model. For example, for a decision tree model a hyper-parameter that can be tuned is the maximum height it can grow, it is important that the algorithm itself is not changed.

1. Present Results:

The usual metrics to present is the accuracy of the model on the testing data, the confusion matrix to observe, numerically, the model’s correct and incorrect predictions, and the classification report to present the model’s precision of the labels.

Tutorial Supplementary Notes

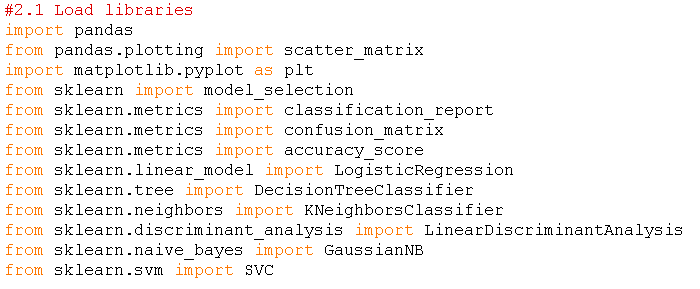
Note: After adding each section of code to a python file in order, it can be executed to review the results.

1. Downloading, Installing and Starting Python SciPy.

Reference a separate document for setup, <https://github.com/JonathanC13/Machine-Learning-Reference-Docs/blob/master/setupMachineLearningTools.docx>.

2. Load the data

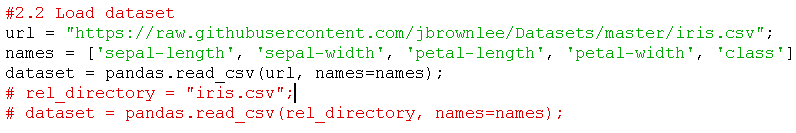
2.1. Import Libraries to be used



These libraries contain the functions and objects that were used to complete a machine learning project end-to-end. Execute a python file with only the code to load the libraries and if there are errors, then SciPy may have not installed completely.

<1. define/understand data>

2.2 Load the Dataset



The dataset can be loaded directly from an online source or the file can be downloaded at <http://archive.ics.uci.edu/ml/datasets/Iris> and referenced locally. Many datasets, when saved, have the file extension .data and if it is preferred to be viewed with excel it can easily be converted by renaming the file. E.g. Iris.data -> Iris.csv

3. Summarize the Dataset

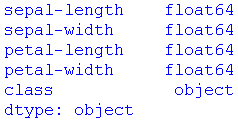
This step is to look at the data and get a better understanding of it, the following properties can be explored.

1. Types in the dataset
2. Dimensions of the dataset.
3. Peek at the data itself.
4. Statistical summary of all attributes.
5. Breakdown of the data by the class variable.

To get a good glance of what type of data that is being used the dtypes property can be used:



Outputs the type of each attribute:



From the output for the types the ‘class’ attribute type is object which indicates categorical data and it is known that it is also the target attribute for the dataset, so it may be useful to know what the unique values are for the ground truths and predictions.



Outputs the unique values of:



3.1. Dimensions of the Dataset

The dimensions of the data are described by how many instances (rows) and attributes (columns) it has. The shape property of the dataset is used:



Which outputs:

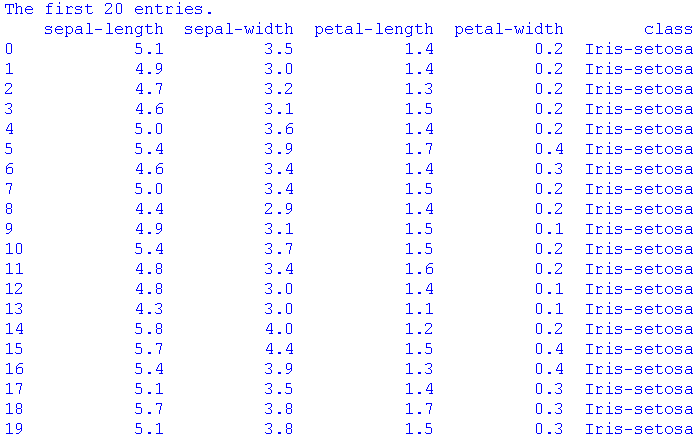


3.2. Peek at the Data

To get a sample of the dataset, the head property is used:



Which outputs:



A note, if there are too many attributes to display the output will be shortened but the option of max columns can be adjusted so they are displayed, place after loading the libraries:

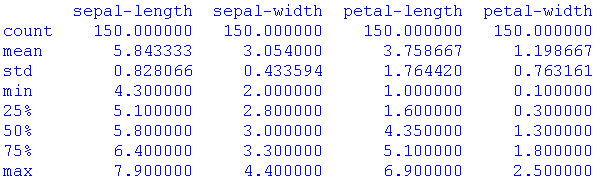


3.3. Statistical Summary

A summary of each attribute can be calculated to provide the user with some helpful statistics.



Which outputs:



- count: number of entries for the column.

- mean: the sum of all the values of the column and divided by the total count.

- std: standard deviation: To quantify the spread or dispersion of the data values related to the mean. A low standard deviation indicates that the data points tend to be close to the mean (also called the expected value) of the set, while a high standard deviation indicates that the data points are spread out over a wider range of values, <https://www.mathsisfun.com/data/standard-deviation-formulas.html>

- min: minimum value for that column.

- max: maximum value for that column.

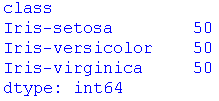
- percentiles: e.g. 25 percentile value is where 25 % of the observations may be found. Therefore, for the 100 percentile all the values are found below it (max value).

3.4 Class Distributions

The number of instances that are associated with each ‘class’ can be retrieved by grouping the instances by class then getting the size:



Which outputs:



4. Visualize the Data

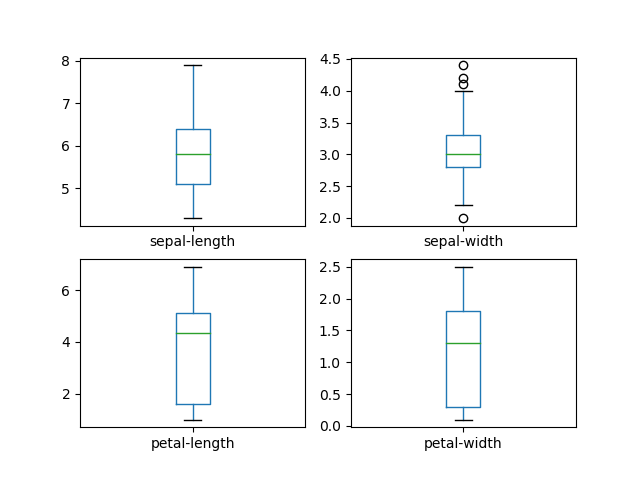
4.1 Univariable Plots

It is must easier to see the distribution of individual attributes using univariable plots.

A simple plot used was the whisker plot:



Outputs the whisker plot:



*Figure 1: Whisker plots of the distributions of the dataset’s attribute’s values*

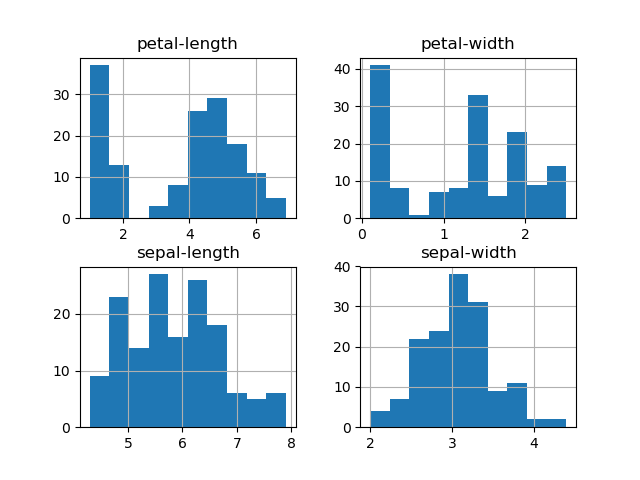
* The box shows the main box that represents the quartile 1 (25 percentile) and quartile 3 (75 percentile).
* Whiskers: The vertical lines extending to the most extreme, non-outlier data points.
* Fliers: (circles outside whiskers) points representing outlier data. <https://www.wikihow.com/Calculate-Outliers>
* Caps: The horizontal lines at the ends of the whiskers to represent the minimum and maximum values.
* The green line in the box is the mean line.

Observations: From the whisker plot, the sepal-width and sepal-length may resemble a Gaussian distribution, but a histogram will more clearly show the distribution.

Another plot that can be used to visualize the distribution of the attributes is to use a histogram:



Outputs the histogram plot:



*Figure 2: Histogram plots of the distribution of the dataset’s attribute’s values*

Observations: Sepal-length and sepal-width histogram resembles a Gaussian distribution and some algorithms are based on the assumption that the data is normally distributed.

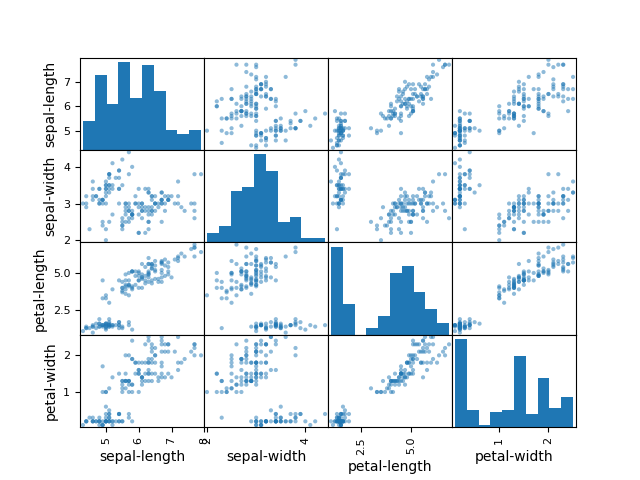
4.2 Multivariable Plots

The use of multivariable plots is to visualize the interactions between the attributes, and it may reveal some correlation.

A quick method to plot all the combinations is with a scatter matrix:



Outputs the scatter matrix:



*Figure 3: Scatter matrix plots of the combinations of the dataset’s attributes*

Observations: The pair of petal-width and petal-length present a strong linear correlation which increases predictability in this relationship. Also, some of the other pairs start to show a linear correlation like the pair petal-length and sepal-length. For a more defined patterns or for a pattern to emerge to emerge more data would be needed, the more valid data is used the better the model.

A correlation between attributes greatly improves the algorithms accuracy for the training dataset due to this predictability and, if the training dataset is valid, the model’s performance will carry over to the unseen dataset.

A note: Attempting to scatter matrix datasets with many attributes will cause long calculation time and the plot window will become cluttered and unreadable. The alternative is to choose a few pairs that may have a correlation that would be interesting to visualize.

</1. define/understand data>

<2. Prepare data>

5. Evaluate Some Algorithms

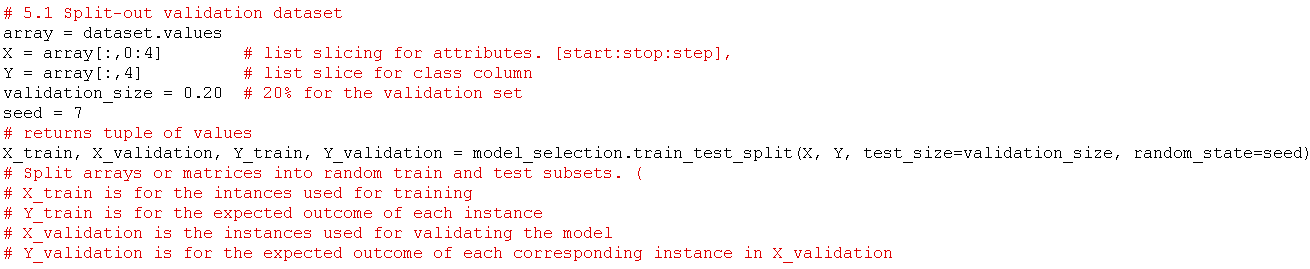
The steps for evaluating algorithms includes:

1. Separate out a validation dataset.
2. Set-up the test harness to use 10-fold cross validation.
3. Build 5 different models to predict species from flower measurements
4. Select the best model.

5.1 Create a validation dataset

From the total dataset a portion of it must be reserved as validation dataset and not be included in the training process’s dataset. This unseen data is used to evaluate how well the model predicts on it and based on the accuracy of each model we can compare algorithms.

In this case 20% of the dataset was reserved as the validation dataset:

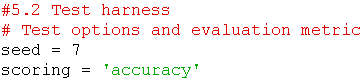


model\_selection.train\_test\_split: Split arrays or matrices into random train and test subsets because it is important that the training data and testing data are not the same, if the model trained on testing data it is expected that is would score perfect on that data when it was time for the testing process, therefore causing overfitting and may result in poor results on actual unseen data.

5.2. Test Harness

For the test harness 10-fold cross validation was used to estimate accuracy. Fold cross validation definition: <https://www.openml.org/a/estimation-procedures/1>

This will split our dataset into 10 parts, train on 9 and 1 part to test predictions and it repeats for all combinations of train-test splits.



The parameter ‘accuracy’ refers to the percentage of correctly predicted outcomes out of the total number of instances in the dataset. This was used to evaluate how well the model performed and to compare models based on this metric.

</ 2. Prepare data>

<3. Select algorithms>

<4. Train the models>

5.3. Build Models

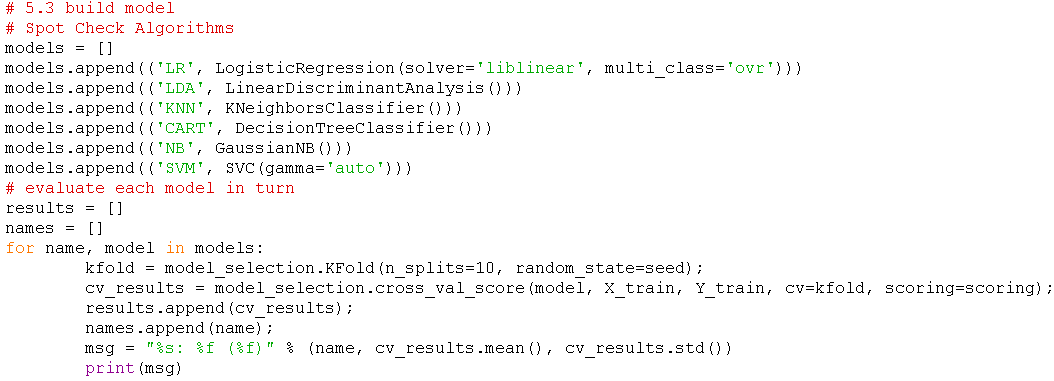
Generally, the best algorithms or even the type of algorithm to use to build an accurate model is unknown, so multiple models can be built and compared. From the multivariable plots, in section 4.2, some of the attributes present some linear correlation, since there is some semblance of a pattern it indicates that the accuracy of the some of the models should be fairly good.

The algorithms used were:

1. Logistic Regression (LR)
2. Linear Discriminant Analysis (LDA)
3. K-Nearest Neighbors (KNN).
4. Classification and Regression Trees (CART).
5. Gaussian Naive Bayes (NB).
6. Support Vector Machines (SVM).

It is good mixture of simple linear (LR and LDA), nonlinear (KNN, CART, NB and SVM) algorithms. The random number seed is reset before building the next model to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable.

To build the models:



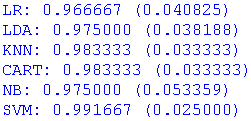
</ 3. Select algorithms>

</ 4. Train the models>

<5. Evaluate>

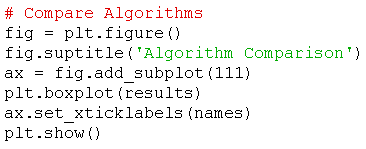
5.4. Select Best Model

The accuracy output:

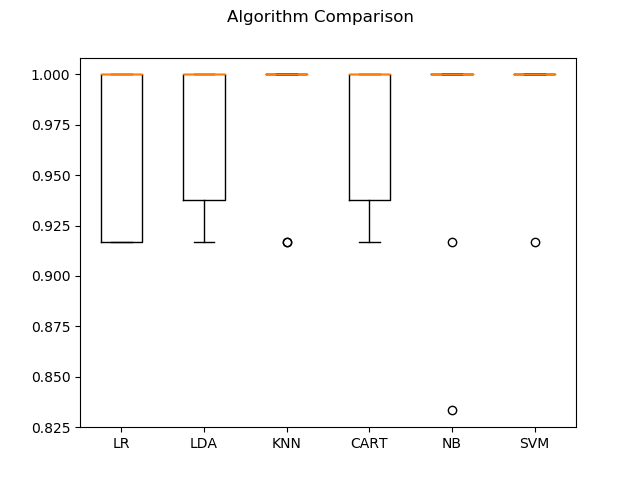


From the evaluation of each algorithm it can be observed that SVM is the most accurate with ~ 99 % and a standard deviation of 0.025 meaning the accuracy scores distribution is very low indicating high consistency.

Each algorithm was evaluated 10 times since 10 k fold cross evaluation was used and to visualize the distribution of the accuracies of each algorithm a whisker plot was used:



The algorithm comparison whisker output:



*Figure 4: The whisker plot for the algorithm comparisons of their accuracies*

</ 5. Evaluate>

<6. Predict>

6. Make predictions

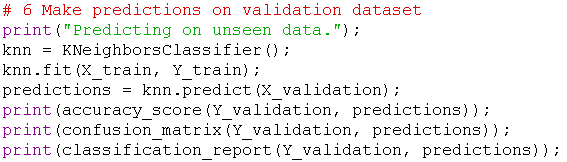
In order to know what improvements that can be made, the problem needs to be identified. In this case the accuracy can be analyzed to determine how fit the algorithm is to the training dataset and then its accuracy on the validation dataset.

Testing on a validation dataset is important because it helps determine whether the model is well fitted, an overfitted model with have high accuracy on the training dataset but when tested on the validation dataset may result in lessened accuracy to indicate this situation.

A solution to overfitting would be to use an algorithm that produces a simpler model from the dataset, so it does not take every outlier or data noise into the model. Another solution would be to add a regularization term to the algorithm that is overfitting, where it penalizes the model when it attempts to fit a data point that variates to much from the mean (?).

On the other hand of underfitting, it produces a model that does not fit the training data well and results in poor accuracy (subjective depending on application, assume < 98%). It indicates that the algorithm is to simple to determine a relationship or pattern within the training dataset. A solution would be to choose a more complicated algorithm.

Since the accuracy of the models built are high, the next step is testing a model on the validation dataset. In the tutorial KNN was used to train a model and then had it make predictions on the unseen data in the validation dataset to test its accuracy.



</ 6. Predict>

<8. Present results>

Result outputs:

- Accuracy score: Compares the validation known answer with the predicted to determine accuracy.



- Confusion matrix: Matrix of accuracy classification.



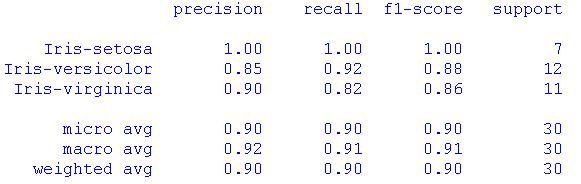
It can be interpreted as:

|  |  |  |  |
| --- | --- | --- | --- |
| Actual \ Prediction | Iris-setosa | Iris-versicolor | Iris-virginica |
| Iris-setosa | 7 | 0 | 0 |
| Iris-versicolor | 0 | 11 | 1 |
| Iris-virginica | 0 | 2 | 9 |

- Classification report:

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report>

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html#sklearn.metrics.precision_recall_fscore_support>



* Columns with the target classes [Iris-setosa, Iris-versicolor, Iris-virginica]:
  + Precision: The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.
    - For Iris-versicolor: 11 / (11 + 2) = 0.85
  + Recall: The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.
    - For Iris-virginica: 9 / (9 + 2) = 0.82
  + F1-score: The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.
    - For Iris-virginica: (0.9 + 0.82) / 2 = 0.86
  + Support: The number of predictions used to determine the other columns values
* Rows: <https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin>
  + Micro avg: averaging the total true positives, false negatives and false positives. It is preferable if there is a suspicion that there might be class imbalance – many more samples in some classes compared to others.
    - Global:
      * = ∑ Tp / ∑ support = (7 + 11 + 9) / 30 = 0.90.
  + Macro avg: averaging the unweighted mean per label, treats all the classes equally in terms of contribution to predictions.
    - Global:
      * I-s: Precision = 1.
      * I-ve: Precision = 0.85.
      * I-vi: Precision = 0.90.
      * = (1 + 0.85 + 0.90) / 3 = 0.92
  + Weighted avg: averaging the support-weighted mean per label, if weight imbalance is known this is helpful to present.
    - Global:
      * = [(( + ) / support)\* ] + [(( + ) / support) \* ] + [(( + ) / support) \* ]
      * = [0.23 \* 1] + [0.43 \* 0.85] + [0.33 \* 0.90]
      * = 0.89.

Observations:

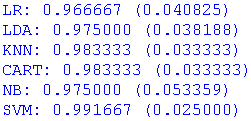
The model has good overall precision in its predictions.

</8. present results>

7. Conclusion

This machine learning problem’s model was offline, used supervised learning, and had classification outputs. The dataset required no categorical conversions or handling of null data.

Due to some strong linear correlation the regression algorithms created accurate models,



, and when the chosen model, KNN, was used to predict on new unseen data after being deployed it performed fairly accurately at an average score of 90%.

In this context this accuracy may be acceptable but in context’s like business analytics or medical imaging the accuracy is expected to be nearing perfect.

Completing this project provided basic insight on the procedure to complete one iteration of an end to end machine learning project.

The next project should exercise at least one additional step in the machine learning procedure, for example data cleaning, a regression output, an online model, or other.