

CompareML

A comparator for machine learning algorithms
libraries and services

User Manual

for CompareML 1.0 - KBS OSP version

Antonio Jesús Fernández-García¹,
Juan Carlos Preciado²,
Álvaro Prieto²,
Fernando Sánchez²,
Juan D. Gutiérrez².

¹*Applied Computing Group, University of Almería*

²*Quercus Research Group, University of Extremadura*

This user guide is for *CompareML* version 1.0

Copyright (c) 2019 – 2020

Antonio Jesús Fernández-García¹, Juan Carlos Preciado², Álvaro Prieto², Fernando Sánchez², Juan D. Gutiérrez².

¹*Applied Computing Group, University of Almería.*

²*Quercus Research Group, University of Extremadura.*

Email addresses: ajfernandez@ual.es (Antonio Jesús Fernández-García), jcpreciado@unex.es (Juan Carlos Preciado), aeprieto@unex.es (Álvaro Prieto), fernando@unex.es (Fernando Sánchez), andy@unex.es (Juan D. Gutiérrez)

The MIT License (MIT)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CompareML

A comparator for machine learning algorithms libraries and services

Table of Contents

1. Introduction	4
2. Installation.....	4
3. User Interface.....	5
3.1. 'Experiments set up' interface	5
3.2. 'Experiments Results' interface.....	6
4. Functionalities.....	7
4.1. Dataset Selection	7
4.2. Label Selection	7
4.3. Providers Selection	8
4.4. Algorithms Selection.....	8
4.5. Experiments Running	9
5. Results of the Experiments	10
6. Illustrative Example	12
Appendix I: List of Machine Learning libraries and services supported	14
Appendix II: List of Machine Learning algorithms supported.....	15
Appendix III: "Iris" and "Mushroom" datasets.....	18

1. Introduction

CompareML is a comparator for machine learning algorithms libraries and services. It makes it easy for users to create a test model of their dataset in three of the most widespread options such as Scikit-Learn, Turi Create and R libraries and, at the same time, allows selecting different well-known classification and regression algorithms available in all providers.

The characteristics of *CompareML* facilitates data scientists the task of choosing the most suitable machine learning provider for their data, improving notably the experiment results while reducing time and costs. Furthermore, *CompareML* helps in selecting the algorithms which are liable to produce the best results for their datasets.

This user manual is structured as follows. Section 2 describes how to access to *CompareML*. Section 3 describes the user interface. Section 4 details the functionalities of the software. Section 5 describes the results shown by *CompareML* after running experiments and Section 6 provides an Illustrative Example that can be easily reproduced. Finally, the Appendix I lists the machine learning libraries and services supported, Appendix II lists the algorithms supported, and Appendix III describes the datasets preloaded in the software to facilitate its usage.

CompareML

A comparator for machine learning algorithms libraries and services

2. Installation

CompareML is a web application, hence it is not necessary to install additional software or driver in order to be used, just a web browser is enough. We suggest using the newest version of the Chrome browser for a better experience.

CompareML does not require users to be registered so, anonymous users can directly access the web application functionalities without a previous sign in.

The fact that *CompareML* users do not need to sign up for the website and it is accessible through conventional web browsers make it easy to researchers, data scientist, researchers, or any users who take an interest in this useful tool for their own purposes without losing time in unnecessary tasks.

3. User Interface

The user interface of CompareML has been designed to maximize usability being simple, consistent and offering cross-browser compatibility. It has an interface to set up the experiments and an interface to show the results of the experiments once they have been completed. The user interfaces are described in the next subsections.

3.1. 'Experiments set up' interface




In this interface, it is possible to set up experiments that want to be carried out. Through this interface, it is possible to upload data, choose the machine learning services and libraries that need to be covered in the experiment, and select the algorithms on which the test models will be built. Figure 1 illustrates this interface:

CompareML A comparator for machine learning algorithms libraries and services

1) Upload your dataset (csv): Ningún archiv...seleccionado or select a default dataset:

2) Select the label (target feature):

3) Choose one or multiple providers:




  

4) Choose one or multiple algorithms:

Regression ☐ Linear Regression ☐ Boosted Decision Trees ☐ Decision Tree

Classification ☐ Random Forest ☐ Logistic Regression ☐ Support Vector Machines

CompareML A comparator for machine learning algorithms libraries and services

Quercus Research Group  GitHub Code Repository  Journal Link 

Cite as:
Paper not yet published

• Antonio Jesús Fernández-García
• Juan Carlos Preciado
• Álvaro Prieto
• Fernando Sánchez
• Juan D. Gutiérrez

Figure 1: 'Experiments set up' user interface

The functionalities and tasks that can be performed in this interface are described in Section 4: Functionalities and a usage example in Section 5: Illustrative Example.

3.2. 'Experiments Results' interface

In this interface, it can be analyzed the results of the experiments previously set up and executed. Through this interface is displayed the information and metrics obtained from the experiments which are the key to deduce which provider and algorithm suits better the provided data. Figure 2 illustrates this interface:

Random Forest	Turi Create	Scikit-learn	R
Accuracy	0.7105831533477321	0.6630669546436285	0.7171
Precision	0.7101214574898785	0.33153347732181426	0.6198
Recall (Sensitivity)	0.9522258414766558	0.5	0.4145
Confusion Matrix	<pre> +-----+ target_label predicted_label count +-----+ 1 0 44 0 1 358 0 0 110 1 1 877 +-----+ [4 rows x 3 columns] </pre>	<pre> 0 1 0 0 468 1 0 921 </pre>	<pre> Reference Prediction 0 1 0 194 119 1 274 802 </pre>
Logistic Regression	Turi Create	Scikit-learn	R
Accuracy	0.714902807775378	0.7163426925845933	0.7163
Precision	0.7432808155699722	0.6816254196128848	0.6186
Recall (Sensitivity)	0.8707926167209555	0.641592889557059	0.4124
Confusion Matrix	<pre> +-----+ target_label predicted_label count +-----+ 1 0 119 0 1 277 0 0 191 1 1 802 +-----+ [4 rows x 3 columns] </pre>	<pre> 0 1 0 193 275 1 119 802 </pre>	<pre> # weights: 51 (50 variable) initial value 3849.046294 iter 10 value 3060.419397 iter 20 value 2988.788748 iter 30 value 2980.109534 iter 40 value 2978.171126 iter 50 value 2977.114804 iter 60 value 2977.051337 final value 2977.051266 converged Reference Prediction 0 1 0 193 119 1 275 802 </pre>
Support Vector Machines	Turi Create	Scikit-learn	R
Accuracy	0.6975241900647948	0.7098632109431245	0.7091
Precision	0.7507507507507507	0.671143594869232	0.5894
Recall (Sensitivity)	0.8143322475570033	0.6430127509117738	0.4509
Confusion Matrix	<pre> +-----+ target_label predicted_label count +-----+ 1 0 171 0 1 249 0 0 219 1 1 750 +-----+ [4 rows x 3 columns] </pre>	<pre> 0 1 0 205 263 1 140 781 </pre>	<pre> Reference Prediction 0 1 0 211 147 1 257 774 </pre>

Figure 2: 'Experiments Results' user interface

The functionalities and tasks that can be performed in this interface are described in Section 4: Functionalities and a usage example and in Section 5: Illustrative Example.

4. Functionalities

The main functionalities of *CompareML* are:

4.1. Dataset Selection

This functionality allows to select the dataset which will be used to perform the experiments. Figure 3 graphically shows the appearance of this functionality.

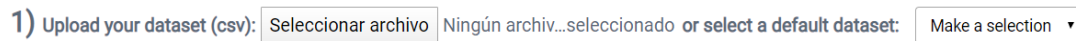


Figure 3: 'Dataset Selection' functionality screenshot.

To select the dataset is possible to either upload a file or choose between some default's dataset that CompareML put at disposal. In the first case, the "choose file" button allows to directly load the dataset from devices' local storage. In the second case, the "make a selection" dropdown menu allows to select between the CompareML available datasets, which are "mushroom" and "iris". Further information about these datasets can be found in Appendix III.

The dataset uploaded by users must be given in CSV format using a comma to separate values. They must contain a header row. Notice that due that the comma is used as a separator, CompareML may not handle field data containing commas or embedded line breaks. Additionally, it may not handle other unconventional characters.

4.2. Label Selection

This functionality allows us to define the label feature, *i.e.*, the column of the dataset which will be predicted by the models built by the machine learning algorithms. Figure 4 graphically shows the appearance of this functionality.

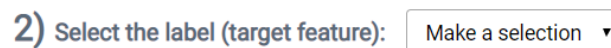


Figure 4: 'Label Selection' functionality screenshot.

After uploading a dataset, the 'label' dropdown menu will load every existing feature in the dataset. Users must select the label from the dropdown menu. Only one label can be selected.

4.3. Providers Selection

Through the “provider selection” functionality, it is possible to choose between the variety of providers available in *CompareML* to build models. Figure 5 graphically shows the appearance of this functionality.

3) Choose one or multiple providers:



Figure 5: 'Providers Selection' functionality screenshot

In *CompareML* version 1.0, the services, libraries and tools available are Turi Create, Scikit-Learn and R. Each of these options will be described in detail later in this manual.

4.4. Algorithms Selection

Through the “algorithms selection” functionality, it is possible to choose between the variety of algorithms available in *CompareML*. Figure 6 graphically shows the appearance of this functionality.

4) Choose one or multiple algorithms:

Regression

☐ Linear Regression ☐ Boosted Decision Trees ☐ Decision Tree

Classification

☒ Random Forest ☒ Logistic Regression ☒ Support Vector Machines

Figure 6: 'Algorithms Selection' functionality screenshot

In *CompareML* version 1.0, there are available Regression and Classification algorithms. In the case of the label feature is a 'numerical' data type, the Linear Regression, Decision Tree, and Boosted Decision Trees regression algorithms can be used. In the case of the label feature is a 'categorical' data type the Random Forest, Logistic Regression, and Support Vector Machine classification algorithms can be used.

It is mandatory to select only one type of algorithm, *i.e.*, Regression or Classification. After that, it is mandatory to select at least one algorithm of the selected type, having the choice to select all of them.

Notice that, in version 1.0, *CompareML* does not automatically recognize the data type of the selected label so the experiments will be run regardless of the appropriateness of the label data type and the type of algorithms selected. In such cases, the results of the experiments will be given according to how each one of the libraries, services, and tools selected responds to these kinds of situations.

4.5. Experiments Running

This functionality compiles all the data entered by the user, ensures that it is correct and processes it to perform the experiments that users need to carry out. If the data is wrong or incomplete, the user is notified.



Figure 7: 'Running Experiments' functionality button screenshot.

5. Results of the Experiments

This functionality provides reports with the results of the experiments carried out. These reports contain metrics that allow to compare the models built in the experiments and helps to decide who is, a priori, the optimal machine learning service or library. Likewise, these metrics allow to easily evaluate the performance of the models, comparing their results, which helps at finding the optimal algorithms for concrete situations.

The metrics thrown by the experiments are different, depending of the type of algorithm selected: *Regression* or *Classification*.

In the case of the *Regression* algorithms are selected, these metrics are:

Metric	Description
RMSE	<p>RMSE is a measure of the differences between the values predicted by a model and the values observed. It can be defined as:</p> $RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2}$ <p>Where:</p> <ul style="list-style-type: none">• N is the number of instances,• $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$, are the values predicted by the model,• y_1, y_2, \dots, y_n are the values observed.
Max-Error	<p>The Max-Error metric is the worst case error between a predicted value and a true value. It can be defined as:</p> $Max\ Error(y, \hat{y}) = \max(y_i - \hat{y}_i)$ <p>Where:</p> <ul style="list-style-type: none">• $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$, are the values predicted by the model,• y_1, y_2, \dots, y_n are the values observed.
Raw Data	Raw data with information directly thrown by the provider.

Table 1: Regression algorithms results.

In the case of the *Classification* algorithms are selected, these metrics are:

Metric	Description
Accuracy	It indicates the correctly predicted instances (number of correctly predicted items / total of items to predict). An accuracy value is given for each algorithm selected.
Precision	It indicates the proportion of predicted positives. With the confusion Matrix it can be calculated as: number of true positives / (number of true positives + number of false positives). A precision value is given for each algorithm selected.
Recall (Sensitivity)	It indicates the proportion of proportion of positives predicted as positives. With the confusion Matrix it can be calculated as: number of true positives / (number of true positives + number of false negatives). A recall value is given for each algorithm selected.
Matrix Confusion	Table layout where each row represents the number of instances of each class and each column represents the class that has been predicted by the model. A confusion matrix is created for each algorithm selected.
Raw Data	Raw data with information directly thrown by the provider.

Table 2: Classification algorithms results.

6. Illustrative Example

In this section, as an illustrative example, we make use of *CompareML* to find the provider that, a priori, is best suited for dealing with the **Qualifications** dataset, which contains qualifications data obtained across several years of students from the University of Extremadura (Spain) in the Computer Science Engineering degree.

We are also interested in getting a better idea about which machine learning algorithm is more likely to get better results at predicting the most appropriate subjects for concrete users with the aim of maximizing their academic performance. This real-world dataset consists of 6942 instances (observations) and 5 features (variables) which are detailed in Table 3.

Feature Name	Description	Type
Subject	Name of the subject.	Categorical
Attempt	Attempt number. A student has up to 6 attempts to pass a subject.	Numerical
Year	Year of the degree programme where the subject takes place.	Categorical
examinationPeriod	Students have the right to attend to pass a subject twice per enrollment: Ordinary and Extraordinary.	Categorical
Mark	The marks have been simplified in 0 (Fail) and 1 (Pass).	Categorical (Label)

Table 3: Set of features of the Qualifications dataset.

The steps followed in this illustrative example are:

1. Provide the dataset to *CompareML*. To do this, the option of selecting a default dataset is ignored, and the **Qualifications** dataset is directly upload from our computer.
2. Select the label feature that we are interested to predict. After uploading the dataset, the dropdown menu of this section is filled with all variables names. In our case, we select the “Mark” variable, which indicates whether a student has passed a concrete subject, *i.e.*, the field we aim to predict.
3. Choose the machine learning libraries and services in which we want to run the experiment (Scikit-Learn, Turi Create and R). At least one of them must be selected. In our example, we are interested in comparing the 3 providers.
4. Select if we are going to carry out a *Regression* or *Classification* experiment. Depending on the type of algorithm selected we can choose from a variety of algorithms. We are interested in predicting a categorical value and, for that reason, we mark the *Classification* algorithms checkbox. After selecting this option, we must select at least one algorithm. In our example, we want to perform an experiment including all the algorithms available (Random Forest, Logistic Regression and Support Vector Machine).

If the experiment is set up properly, we can run the experiments by pressing the “Start” button. If not, an error message will be shown describing how to solve the problem.

When the experiments are performed, *CompareML* shows the results obtained below the “Start” button. Our experiment reports the results shown in Table 4, which include the Accuracy, Precision and Recall metrics. *CompareML* also provides the *Confusion Matrix* and raw data with information directly thrown by the provider, which are omitted for greater readability.

		Turi Create	Scikit Learn	R
Random Forest	Accuracy	0.7106	0.6630	0.7171
	Precision	0.7101	0.3315	0.6198
	Recall	0.9522	0.5000	0.4145
Logistic Regression	Accuracy	0.7149	0.7163	0.7163
	Precision	0.7433	0.6816	0.6186
	Recall	0.8708	0.6416	0.4124
Support Vector Machine	Accuracy	0.6976	0.7099	0.7091
	Precision	0.7508	0.6711	0.5894
	Recall	0.8143	0.6430	0.4509

Table 4: Experiments Results.

As expected, building a model using the same algorithm and the same data produce similar but slightly different results, which implies that even when the algorithms are the same, their implementations by different providers impact in the performance of the models built using them.

Although in most cases those differences are small, in some cases they are significant. Let us focus in the *Accuracy* metric. For example, in our case study, the accuracy of the *Random Forest* model with *R* is 0.0541 points greater than with *Turi Create*, which is a considerable difference. Although it is true that in the *Logistic Regression* and *Support Vector Machine* models, the difference between the best and worst model is 0.0014 and 0.123 respectively, which is much smaller.

Looking at the outcome of the experiments, we can see that in all cases, the accuracy of the *Support Vector Machine* models, regardless of the vendor, produces worst results than the *Logistic Regression* models. It may be an indication that these models are more suitable for this dataset, even though when the best accuracy results are met by the *Random Forest* model built with *R*.

Appendix I: List of Machine Learning libraries and services supported

CompareML version 1.0 support the following machine learning libraries and services:

Turi Create

Turi Create makes use of the *turicreate* library to build and evaluate the models and the *pandas* and *Sframe* libraries to manipulate data using its *DataFrame* and *Sframe* data structure and functions respectively.

Turi Create is a Python package that allows programmers to perform end-to-end large-scale data analysis and data product development. It is a distributed computation framework written in C++ developed at the Carnegie Mellon University acquired by Apple Inc. in 2016.

Scikit Learn

Scikit Learn makes use of the *sklearn* library to build and evaluate the models and the *pandas* library to manipulate data using its *DataFrame* data structure and functions.

Scikit-Learn is one of the most popular machine learning libraries. It is largely written in Python with some core algorithms written in Cython to improve performance. It is supported by several institutional and private grants.

R

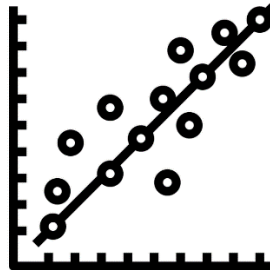
R is a language and environment for statistical computing and graphics. It needs to be emphasized that *CompareML* runs R code embedded in Python through the access provided by the *rpy2* library.

CompareML make use of the following R resources: *lm* function (Linear Regression); *rpart* package (Decision Tree) *xgboost*, *caret* and *tidyverse* packages (Boosted Decision Trees); *randomForest* and *caret* packages (Random Forest); *caret* package (Logistic Regression); *e1071* package (Support Vector Machines).

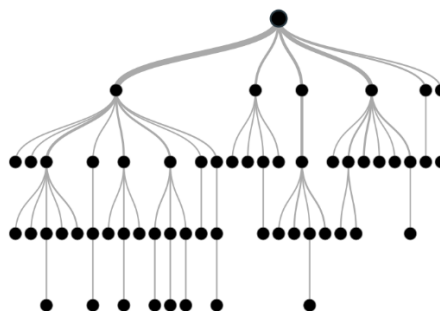
Appendix II: List of Machine Learning algorithms supported

Regression Algorithms

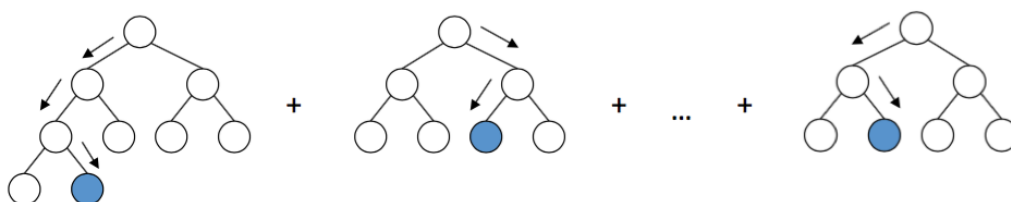
- **Linear Regression.** It is a linear model that assumes a linear relationship between the input variables and the single output variable. The model learns estimating the values of the coefficients used in the representation with the data that we have available. Linear regression can be defined as $y = ax + b$ where a and b are the mentioned coefficients.



- **Decision Tree.** Decision trees algorithms build a tree-like structure where each node represents a question over an attribute. The answers to that question create new branches to expand the structure until the end of the tree is reached, being the leaf node the one that indicates the predicted class.

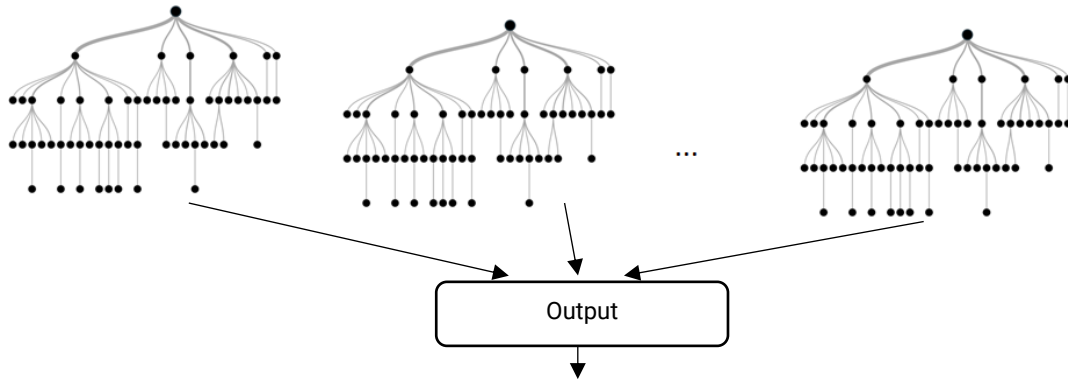


- **Boosted Decision Tree.** It is a general method, not limited to decision trees, that consists of combining many classifiers into a new more stable one with smaller error applying a boosting method. In a boosting the predictors are not made independently, but sequentially, applying the logic that the subsequent predictors learn from the mistakes of the previous predictors.

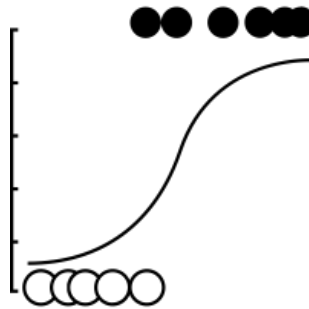


Classification Algorithms

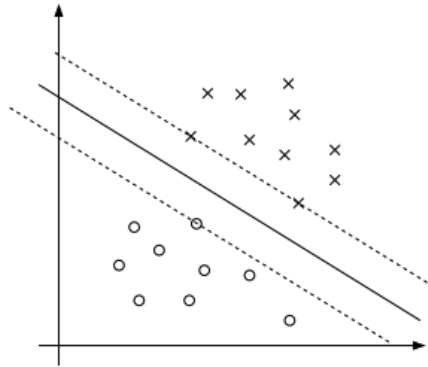
- Random Forest. The *Random Forest* algorithm is an improvement that creates several decision trees, using bagging or other technique, and votes the most popular output of them. Usually, most of the implementation does not directly count the output of them but sum the normalized frequency of each output in each tree to get the label with more probability.



- Logistic Regression. It uses a more complex cost function than the Linear Regression model, which is the 'Sigmoid function' or 'logistic function'. Input values are combined linearly using weights or coefficient values to predict an output value. A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value. It can be defined as: $f(x) = \frac{1}{1+e^{-(x)}}$.



- A Support Vector Machine algorithm classifies cases by finding a separator. It works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. Then, a separator is estimated for the data. The data should be transformed in such a way that a separator could be drawn as a hyperplane. As there are many possible hyperplanes, the Support Vector Machine algorithm finds a hyperplane that represents the largest separation, or margin, between classes.



Appendix III: “Iris” and “Mushroom” datasets

Iris dataset

It contains 4 features and a label with 3 classes (Iris Setosa, Iris Versicolour, Iris Virginica) of 50 instances each, where each class refers to a type of iris plant. The predicted attribute class is “iris plant”.

The 5 first instances are:

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa

More information about this dataset can be found in:

<https://archive.ics.uci.edu/ml/datasets/Iris>

Mushroom dataset

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family.

It contains 23 features and a label with 2 classes (edible=e, poisonous=p). The number of Instances is 8124.

The 5 first instances are:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	L
e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	n	g	0
e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	n	m	1
p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	k	s	u	1
e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e	n	a	g	0
e	x	y	y	t	a	f	c	b	n	e	c	s	s	w	w	p	w	o	p	k	n	g	0

The name of the features is:

1. cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s.
2. cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s.
3. cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y.
4. bruises?: bruises=t, no=f.
5. odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s.
6. gill-attachment: attached=a, descending=d, free=f, notched=n.
7. gill-spacing: close=c, crowded=w, distant=d.
8. gill-size: broad=b, narrow=n.
9. gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y.
10. stalk-shape: enlarging=e, tapering=t.
11. stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?.
12. stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s.
13. stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s.
14. stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y.
15. stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y.
16. veil-type: partial=p, universal=u.
17. veil-color: brown=n, orange=o, white=w, yellow=y.

- 18. ring-number: none=n, one=o, two=t.
- 19. ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z.
- 20. spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y.
- 21. population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y.
- 22. habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d.

More information about this dataset can be found in:

<https://archive.ics.uci.edu/ml/datasets/mushroom>