CompareML

A comparator for machine learning algorithms libraries and services

---

User Manual

for CompareML 1.0

Antonio Jesús Fernández-García,
Juan Carlos Preciado,
José María Conejero,
Roberto Rodríguez-Echeverria,
Fernando Sánchez.

This user guide is for *CompareML* version 1.0

Antonio Jesús Fernández-García, Juan Carlos Preciado, José María Conejero, Roberto Rodríguez-Echeverria, Fernando Sánchez.

Quercus Research Group, University of Extremadura
Applied Computing Group, University of Almeria

Welcome to the *CompareML* User Manual

CompareML

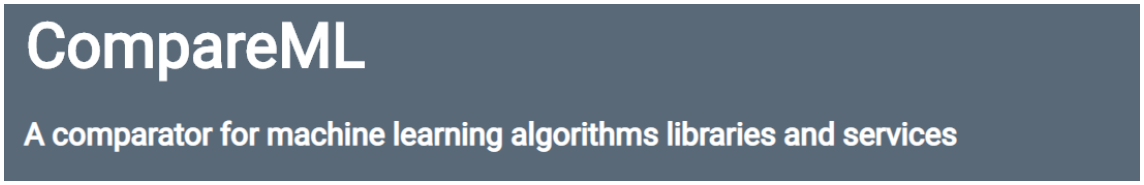
A comparator for machine learning algorithms libraries and services

Table of Contents

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 Graphlab 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.



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 and users, in general, to make use of 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:

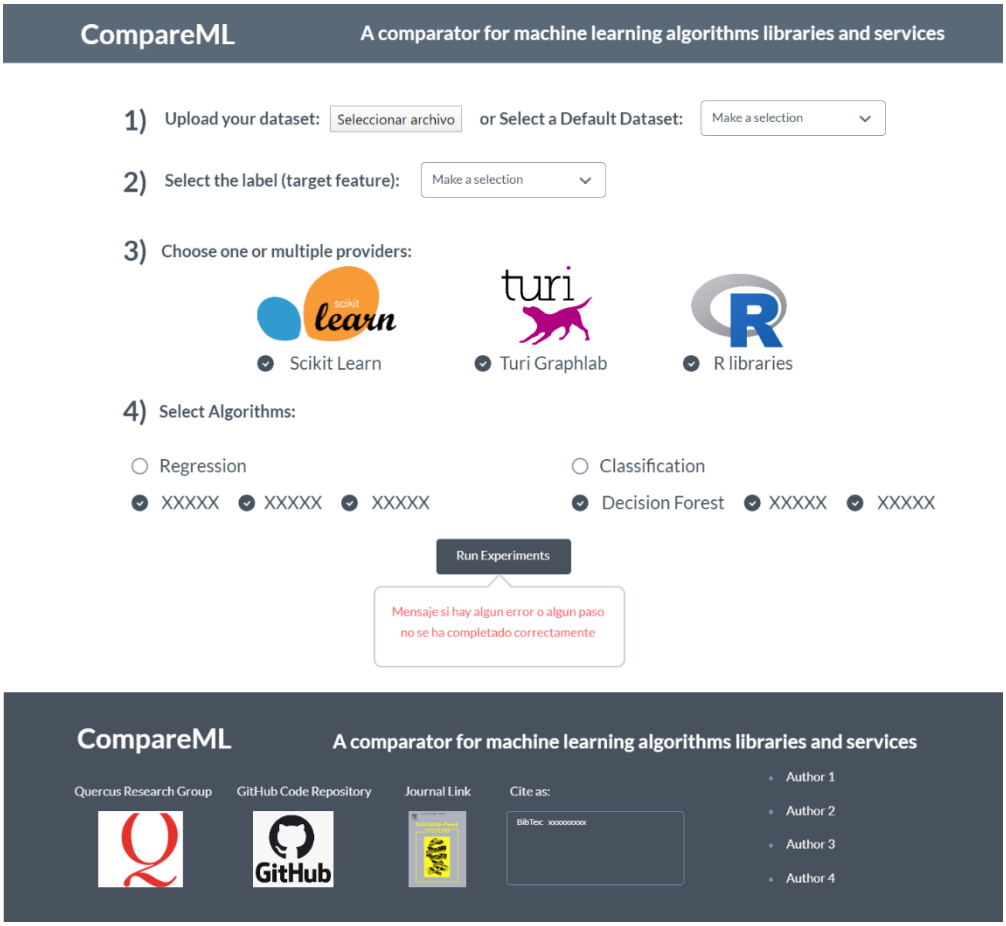


Figure 1: 'Experiments set up' user interface (actualizar)

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:

Figure 2: ‘Experiments Results’ user interface (actualizar)

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.

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.

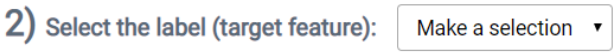


Figure 4: 'Label Selection' functionality screenshot.

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.

Figure 5: 'Providers Selection' functionality screenshot

In *CompareML* version 1.0, the services, libraries and tools available are Turi Graphlab 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.

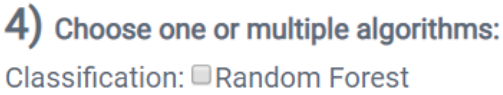


Figure 6: 'Algorithms Selection' functionality screenshot (actualizar)

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 Neural Network 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	RMSE is a measure of the differences between the values predicted by a model and the values observed. It can be defined as: $$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$ Where: • 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	The Max-Error metric is the worst case error between a predicted value and a true value. It can be defined as: $$Max\ Error\ (y, \hat{y}) = \max\ (\lvert y_i - \hat{y}_i \rvert)$$ Where: • $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$, are the values predicted by the model, • $y_1, y_2, \dots, y_n$ are the values observed.

Table 1: Regression algorithms results.

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

Metric	Description
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.
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.

Table 2: Classification algorithms results.

6. Illustrative Example (pendiente implementación)

As an illustrative example, we are going to use *CompareML* to choose the provider that, a priory, is best suited for dealing with a dataset. The example dataset, called Graduates.csv, includes data obtained across several years of 6867 students of the University of Almería (Spain) during their first year in college labeled as graduated or not.

This real-world dataset was generated by the authors with the aim of creating a decision model that could be used to predict if a university student will graduate after finishing the first year of his/her college studies.

The features that make up this dataset are:

Feature Name	Description	Type
Age	Age	Numeric
BirthProvince	Place of Birth	String
Nationality1	First Nationality	String
DoubleNationality	Has the student more than a nationality?	Boolean
Degree	Degree Name	String
DegreeField	Degree Field	String
Faculty	Faculty Name	String
UniversityAccessType	How the student access to the degree String	
CreditsEnrolled	Number of Credits Enrolled	Numeric
AverageScore	Average Score in the Subjects Enrolled (0-10) Numeric	
SuccessRate	Percentage of Credits Pass	Numeric
Graduated	Has the student graduated? *(Label)*	Boolean

The first step consists of providing the dataset to *CompareML*. To do this, the option of selecting a default dataset is ignored, and the 'Graduates' dataset is directly upload from our computer.

In the second step, we choose the label feature that we are interested to predict, which is the 'Graduated' feature. This Boolean feature has two classes (Y, N), that indicates whether a student has finished a concrete university degree.

In the third step, the libraries and services supported by *CompareML* are enumerated. In version 1.0 Scikit Learn, Turi Graphlab and R providers are supported. At least one of them must be selected. In our example we are interested in comparing the 3 providers, so we select all of them.

The fourth step consists of selecting the algorithms that will be used to build the models. There are two options available, *Regression* or *Classification*. Depending on the type of algorithm selected we can choose from a variety of algorithms. In our example, we are interested in predicting a categorical value and, for that reason, we mark the *Classification* algorithms checkbox. If we were interested on predicting a numerical value, we would have marked the *Regression* algorithms checkbox.

After selecting this option, we must select at least one *Classification* algorithm. In our example we want to perform an experiment including all possible algorithms, so we select all of them (Random Forest, Logistic Regression, and Neural Network).

If everything is properly setup and the parameters are set correctly, we can run the experiments by pressing the "Start" button. If not, an error message will be shown describing how to solve the problem.

% Cuando este operativa la plataforma poner los resultados

Appendix I: List of Machine Learning libraries and services supported

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

Turi Graphlab Create

Turi Graphlab 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.

GraphLab 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.

This module, if deployed following a microservice-architecture, encloses a web service that communicates with the Back-end receiving the algorithms that need to be used to build models and the inputs of the experiment. When the experiments are carried out, the results are sent back to the Back-end main module.

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.

As happens in the Turi Graphlab Create module, if deployed following a microservice-architecture, it encloses a web service that communicates with the Back-end receiving the algorithms that need to be used to build models and the inputs of the experiment. When the experiments are carried out, the results are sent back to the Back-end main module.

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.

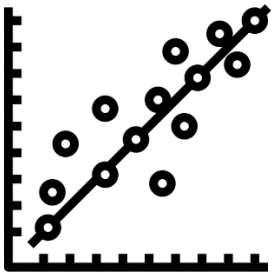
The algorithms available in *CompareML* makes use of the following R libraries:

As happens in the Scikit Learn Module and the Turi Graphlab Create module, if deployed following a microservice-architecture, it encloses a web service that communicates with the Back-end receiving the algorithms that need to be used to build models and the inputs of the experiment. When the experiments are carried out, the results are sent back to the Back-end main module.

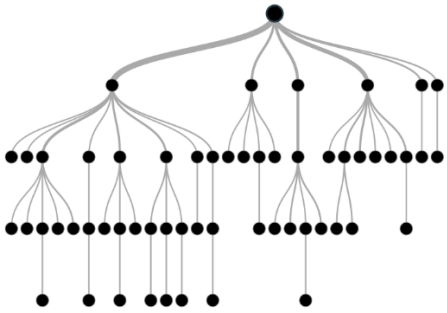
Appendix II: List of Machine Learning algorithms supported

Regression Algorithms

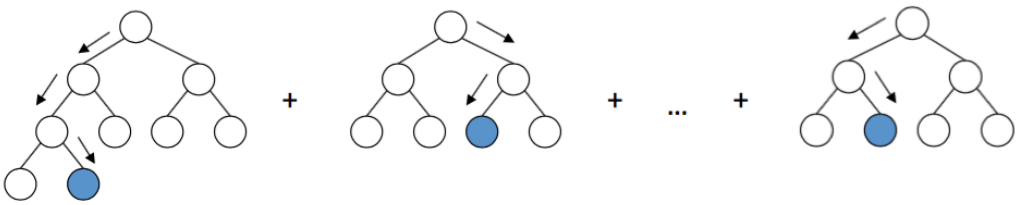
- Linear Regression. It is a linear model that assumes a linear relationship between the input variables and the single output variable. The model learns estimating the values of the coefficients used in the representation with the data that we have available. Linear regression can be defined as $y =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

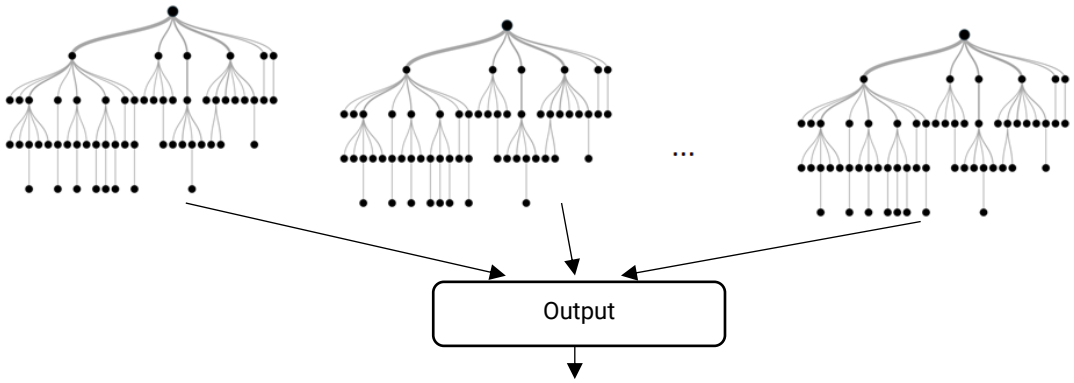


- Boosted Decision Tree. It is a general method, not limited to decision trees, that consists of combining many classifiers into a new more stable one with smaller error applying a boosting method. In a boosting the predictors are not made independently, but sequentially, applying the logic that the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

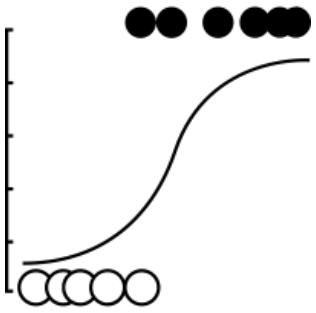


Classification Algorithms

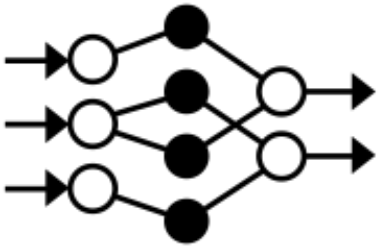
- Random Forest. The *Random Forest* algorithm is an improvement that creates several decision trees, using bagging or other technique, and votes the most popular output of them. Usually, most of the implementation does not directly count the output of them but sum the normalized frequency of each output 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: $f(x) = \frac{1}{1+e^{-(x)}}$.



- Neural Network. The Artificial Neural Network algorithm creates a set of interconnected levels, where each level consists of a set of nodes (neurons) that receives input and produces weighted outputs. The nodes of layer 1 are the inputs, the nodes of the last layer are the output and the nodes in between are called “hidden nodes”. A neural network can be seen as a weighted directed acyclic graph.



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