Datasets

Explore, analyze, and share quality data.

Types of Datasets

Kaggle supports a variety of dataset publication formats, but we strongly encourage dataset publishers to share their data in an accessible, non-proprietary format if possible. Not only are open, accessible data formats better supported on the platform, they are also easier to work with for more people regardless of their tools.

Supported File Types

CSVs

The simplest and best-supported file type available on Kaggle is the "Comma-Separated List", or CSV, for tabular data. CSVs uploaded to Kaggle should have a header row consisting of human-readable field names. A CSV representation of a shopping list with a header row, for example, looks like this:

```
id,type,quantity

0,bananas,12

1,apples,7
```

CSVs are the most common of the file formats available on Kaggle and are the best choice for tabular data.

On the Data tab of a dataset, a preview of the file's contents is visible in the data explorer. This makes it significantly easier to understand the contents of a dataset, as it eliminates the need to open the data in a Notebook or download it locally.

CSV files will also have associated column descriptions and column metadata. The column descriptions allow you to assign descriptions to individual columns of the dataset, making it easier for users to understand what each column means. Column metrics, meanwhile, present high-level metrics about individual columns in a graphic format.

"The Complete Pokemon Dataset" is an example of a great CSV-type Dataset.

Data Cleaning and Preprocessing

Data preprocessing involves the transformation of the raw dataset into an understandable format. Preprocessing data is a fundamental stage in data mining to improve data efficiency. The data preprocessing methods directly affect the outcomes of any analytic algorithm.

Data preprocessing is generally carried out in 7 simple steps:

Steps In Data Preprocessing:

1. Gathering the data
2. Import the dataset & Libraries
3. Dealing with Missing Values

4. Divide the dataset into Dependent & Independent variable
5. dealing with Categorical values
6. Split the dataset into training and test sets
7. Feature Scaling

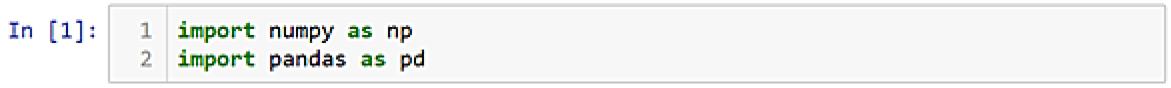
1. Gathering the data

Data is raw information, its the representation of both human and machine observation of the world. Dataset entirely depends on what type of problem you want to solve. Each problem in machine learning has its own unique approach.

Kaggle: Kaggle is my personal favorite one to get the dataset. https://www.kaggle.com/datasets

2. Import the dataset & Libraries

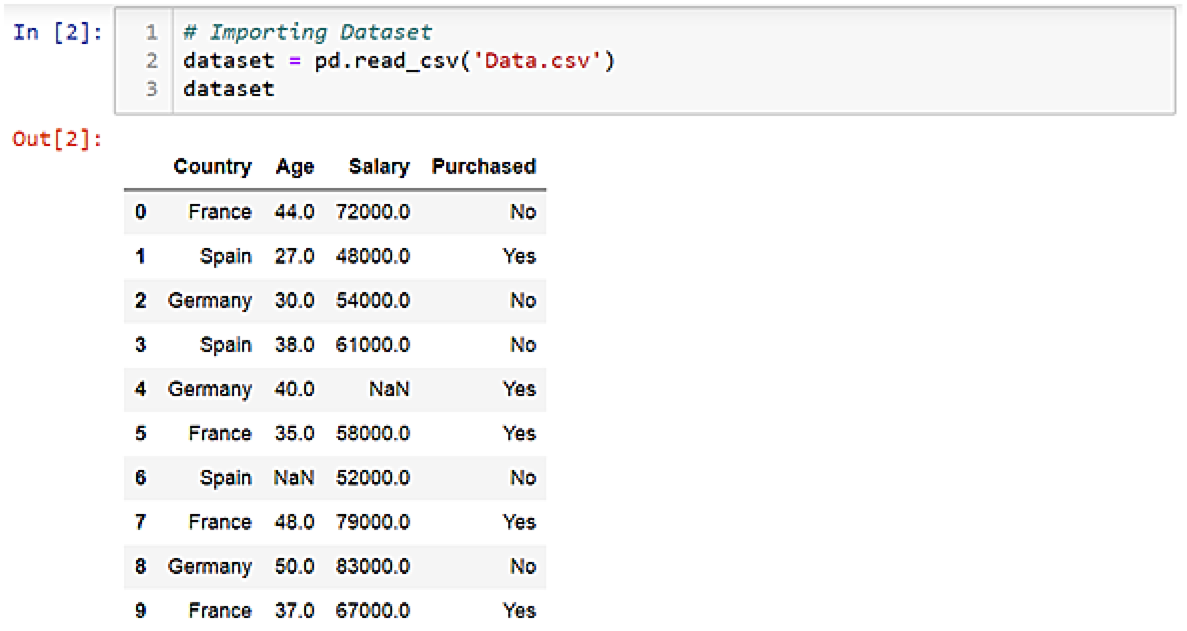
The First step is usually importing the libraries that will be needed in the program. A library is essentially a collection of modules that can be called and used.



```
In [1]:
import numpy as np
import pandas as pd
```

Importing the dataset

Loading the data using Pandas library using *read_csv()* method.



```
In [2]: # Importing Dataset
        dataset = pd.read_csv('Data.csv')
        dataset
```

Out[2]:

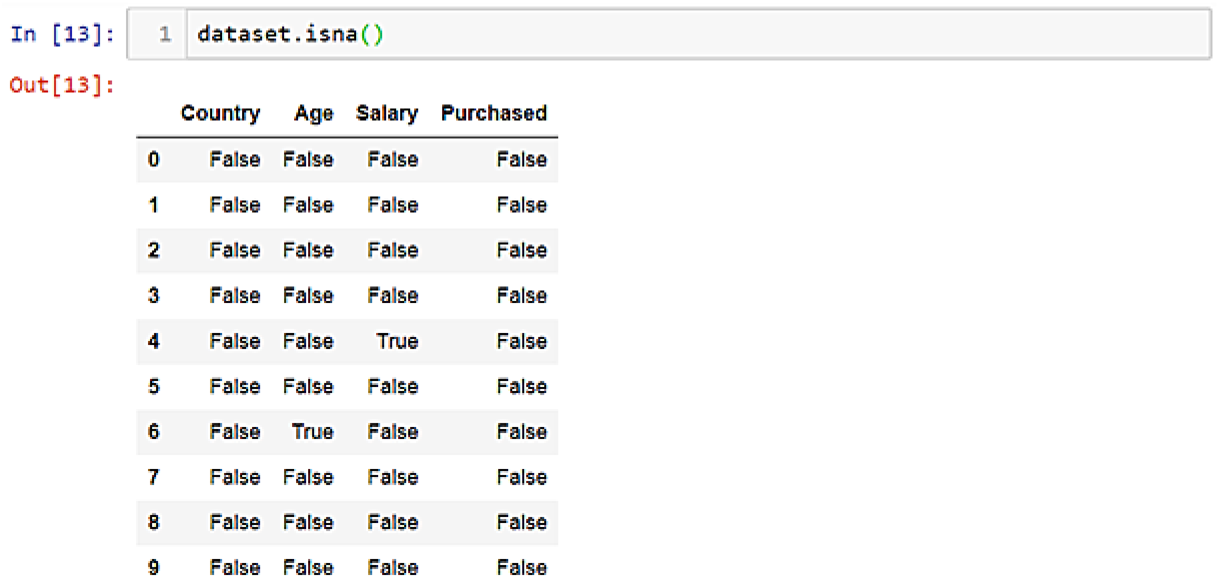
	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

3. Dealing with Missing Values

Sometimes we may find some data are missing in the dataset. if we find then we will remove those rows or we can calculate either **mean, mode or median** of the feature and replace it with missing values. This is an approximation which can add variance to the dataset.

#Check for null values:

we can check the null values in our dataset with pandas library as below.



```
In [13]:  dataset.isna()
Out[13]:
```

	Country	Age	Salary	Purchased
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	True	False
5	False	False	False	False
6	False	True	False	False
7	False	False	False	False
8	False	False	False	False
9	False	False	False	False

r

pre-processing

Preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance,[1] and is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: –100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), and missing values, etc.

Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running any analysis.[2] Often, data preprocessing is the most important phase of a machine learning project, especially in computational biology.[3] If there is a irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take a considerable amount of processing time. Examples of data preprocessing include cleaning, instance selection, normalization, one hot encoding, transformation, feature extraction and selection, etc. The product of data preprocessing is the final training set.

Algorithm

Writing a logical step-by-step method to solve the problem is called the **algorithm**. In other words, an algorithm is a procedure for solving problems. In order to solve a mathematical or computer problem, this is the first step in the process.

An algorithm includes calculations, reasoning, and data processing. Algorithms can be presented by natural languages, pseudo code, and flowcharts, etc.

Types of Algorithm

1. Recursive Algorithm

2. Divide and Conquer Algorithm

3. Dynamic Programming Algorithm

4. Greedy Algorithm

5. Brute Force Algorithm

6. Backtracking Algorithm

Save the models

Model progress can be saved during and after training. This means a model can resume where it left off and avoid long training times. Saving also means you can share your model and others can recreate your work. When publishing research models and techniques, most machine learning practitioners share:

- code to create the model, and
- the trained weights, or parameters, for the model

Building a Web Application Using Flask

If you are a data scientist or data science student, you know how machine learning works. You know the details of the algorithms, which libraries to use, and perform diagnostics. Let's say that in a business environment, you have the perfect model in your hands, with excellent results and ideal everything. Then what? It is indeed tempting to suggest that we just hand over the code to the relevant stakeholders and ask them to run it if they want to see results. But that is not how a business environment works.

One solution to this problem is by creating a machine learning web application. With this solution, stakeholders can access and run your machine learning model through a website instead of some code they don't understand. This tutorial will show you how to create a machine learning web application using Python for the machine learning model, Flask for the back-end engine, and HTML for the front-end.

As a side note, I am running this tutorial on a Mac. As far as I am aware, there should be no difference in any part of the tutorial if you are using other operating systems.