

|  |
| --- |
| DATA SCIENCE & BIG DATA |
|  |
| FINAL TERM PROJECT  TEACHER:  Mr. Mazhar Javed |

PR DATA SCIENCE PROJECT

DESCRIPTION:

PARTICIPANT NAME: *AREEJ FATIMA*

PARTICIPANT ID: *F2018065146*

TOPIC: *DATA SCIENCE TECHNOLOGY & BIG DATA PROGRAMMING*

# BIG DATA

1. RDD

RDD stands for Resilient Distributed Dataset. It is collection of such data which is divided among different nodes in your cluster. It operates with an API.

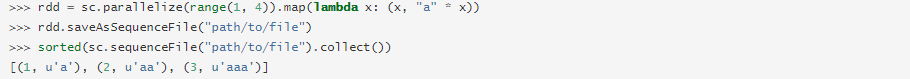
Spark revolves around the concept of a resilient distributed dataset (RDD), which is a fault-tolerant collection of elements that can be operated on in parallel. There are two ways to create RDDs: parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared filesystem, HDFS, HBase, or any data source offering a Hadoop Input Format.

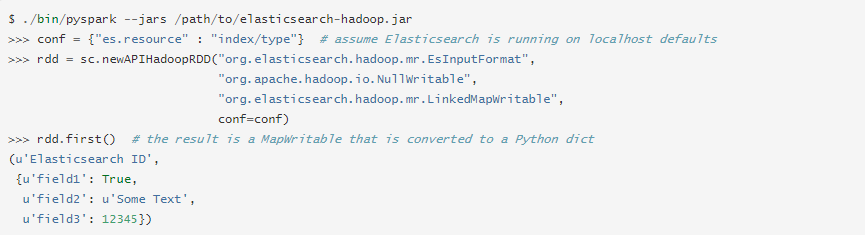
RDDs support two types of operations: transformations, which create a new dataset from an existing one, and actions, which return a value to the driver program after running a computation on the dataset. For example, map is a transformation that passes each dataset element through a function and returns a new RDD representing the results. On the other hand, reduce is an action that aggregates all the elements of the RDD using some function and returns the final result to the driver program (although there is also a parallel reduceByKey that returns a distributed dataset).



Once created, the distributed dataset (distData) can be operated on in parallel. For example, we can call distData.reduce(lambda a, b: a + b) to add up the elements of the list. We describe operations on distributed datasets later on.

#### SAVING AND LOADING SEQUENCE

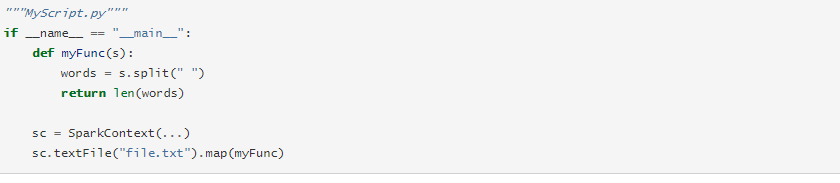




PySpark can also read any Hadoop Input Format or write any Hadoop Output Format, for both ‘new’ and ‘old’ Hadoop MapReduce APIs. If required, a Hadoop configuration can be passed in as a Python dict. Here is an example using the Elasticsearch ES Input Format:

Note that, if the Input Format simply depends on a Hadoop configuration and/or input path, and the key and value classes can easily be converted according to the above table, then this approach should work well for such cases.

#### PASSING FUNCTION TO SPARK

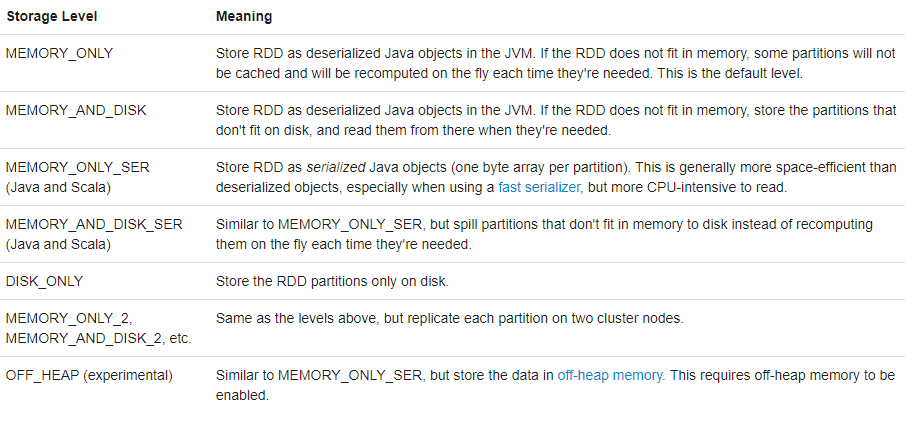


Note that while it is also possible to pass a reference to a method in a class instance (as opposed to a singleton object), this requires sending the object that contains that class along with the method. For example, consider:

#### RDD PERSISTENCE

One of the most important capabilities in Spark is persisting (or caching) a dataset in memory across operations. When you persist an RDD, each node stores any partitions of it that it computes in memory and reuses them in other actions on that dataset (or datasets derived from it). This allows future actions to be much faster (often by more than 10x). Caching is a key tool for iterative algorithms and fast interactive use.

You can mark an RDD to be persisted using the persist() or cache() methods on it. The first time it is computed in an action, it will be kept in memory on the nodes. Spark’s cache is fault-tolerant – if any partition of an RDD is lost, it will automatically be recomputed using the transformations that originally created it.



1. DATAFRAMES

Data Frame is basically a structure of data which is placed two dimensionally with two types of columns It can potentially be any of them:

* SQL TABLE
* DICT
* SPREADSHEET

It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood. Data frames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs. The Data frame API is available in Scala, Java, Python, and R. In Scala and Java, a Data Frame is represented by a Dataset of Rows. In the Scala API, Data Frame is simply a type alias of Dataset[Row]. While, in Java API, users need to use Dataset<Row> to represent a Data frame.

#### SPARK SESSION



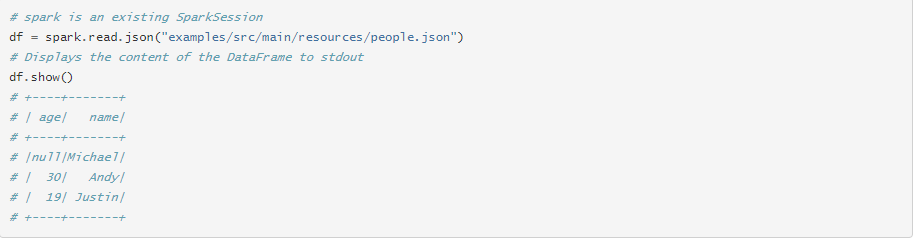
Spark Session in Spark 2.0 provides built in support for Hive features including the ability to write queries using HiveQL, access to Hive UDFs, and the ability to read data from Hive tables. To use these features, you do not need to have an existing Hive setup.

#### CREATING DATAFRAMES

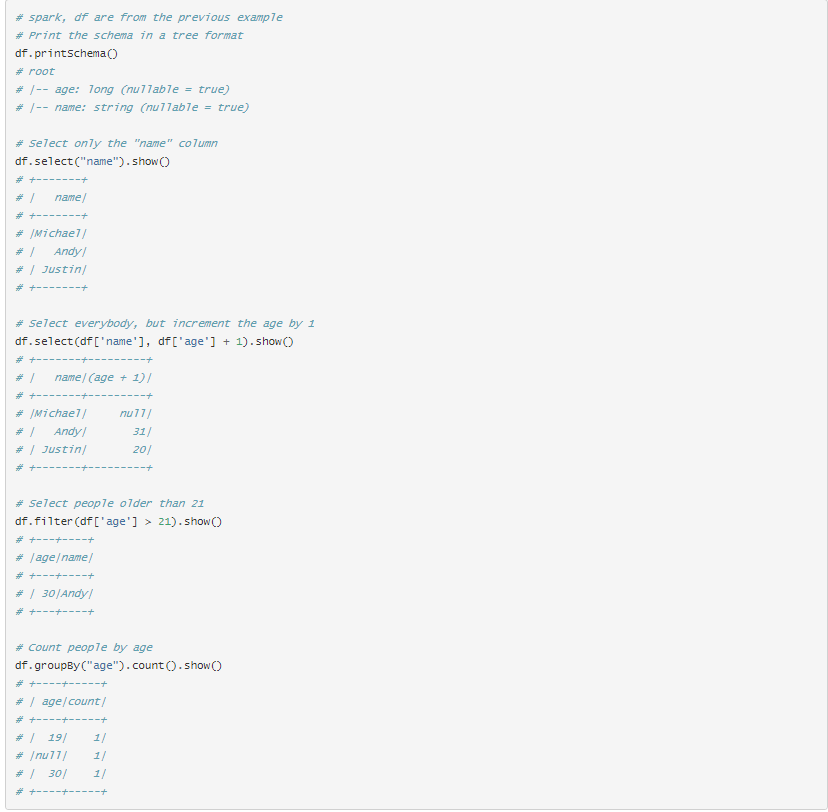
Data Frames provide a domain-specific language for structured data manipulation in Scala, Java, Python and R.

As mentioned above, in Spark 2.0, Data Frames are just Dataset of Rows in Scala and Java API. These operations are also referred as “untyped transformations” in contrast to “typed transformations” come with strongly typed Scala/Java Datasets.

Here we include some basic examples of structured data processing using Datasets:



#### DATASET OPERATIONS

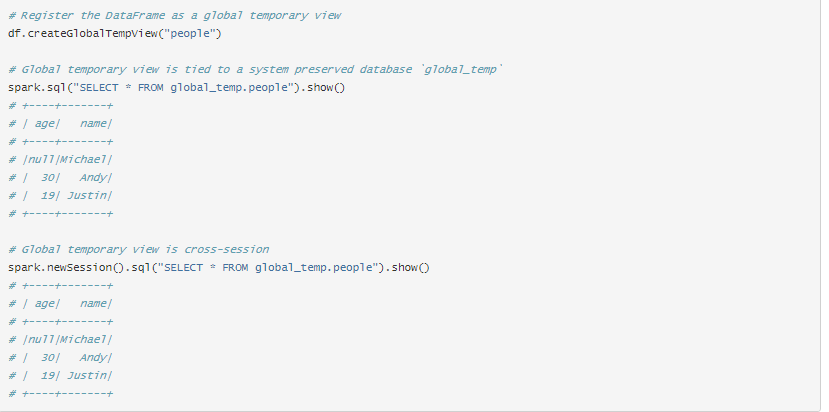


Data Frames provide a domain-specific language for structured data manipulation in Scala, Java, Python and R.

As mentioned above, in Spark 2.0, Data Frames are just Dataset of Rows in Scala and Java API. These operations are also referred as “untyped transformations” in contrast to “typed transformations” come with strongly typed Scala/Java Datasets.

Here we include some basic examples of structured data processing using Datasets:

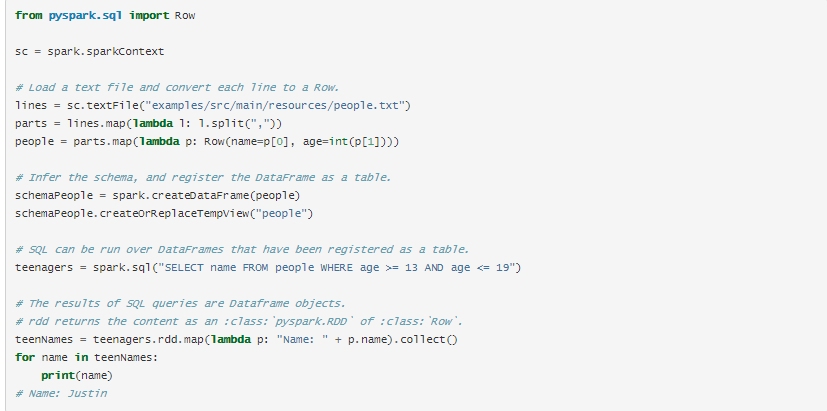
Temporary views in Spark SQL are session-scoped and will disappear if the session that creates it terminates. If you want to have a temporary view that is shared among all sessions and keep alive until the Spark application terminates, you can create a global temporary view. Global temporary view is tied to a system preserved database global\_temp, and we must use the qualified name to refer it, e.g. SELECT \* FROM global\_temp.view1.



#### CREATING DATASET

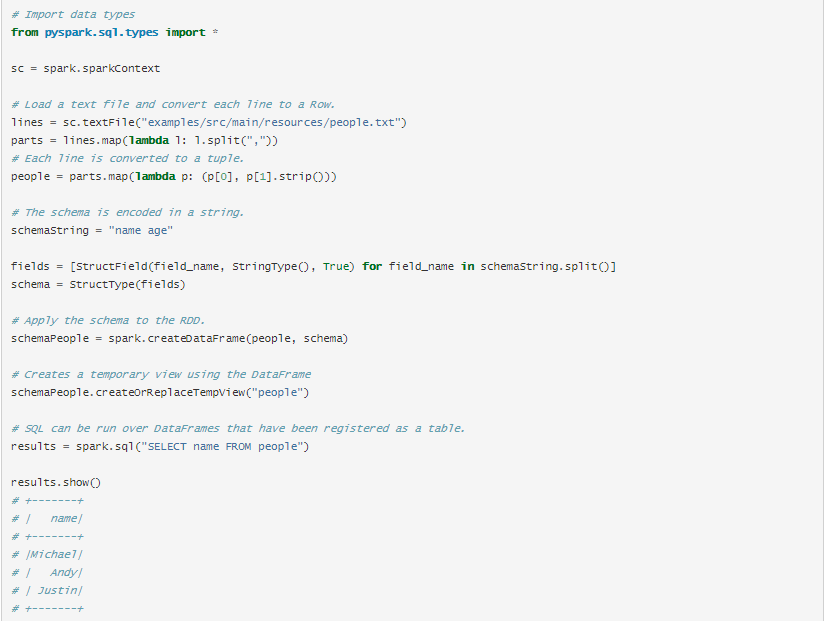


Datasets are similar to RDDs, however, instead of using Java serialization or Kryo they use a specialized Encoder to serialize the objects for processing or transmitting over the network. While both encoders and standard serialization are responsible for turning an object into bytes, encoders are code generated dynamically and use a format that allows Spark to perform many operations like filtering, sorting and hashing without deserializing the bytes back into an object.



Spark SQL can convert an RDD of Row objects to a Data Frame, inferring the datatypes. Rows are constructed by passing a list of key/value pairs as kwargs to the Row class. The keys of this list define the column names of the table, and the types are inferred by sampling the whole dataset, similar to the inference that is performed on JSON files.

#### 2.6 PROGRAMMATICALLY SPECIFING THE SCEHEMA



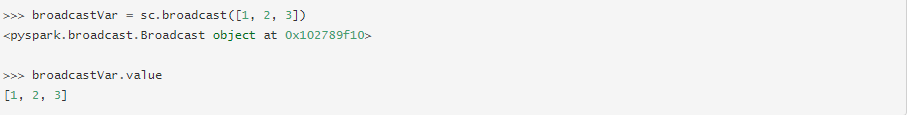
When a dictionary of kwargs cannot be defined ahead of time (for example, the structure of records is encoded in a string, or a text dataset will be parsed and fields will be projected differently for different users), a Data Frame can be created programmatically with three steps.

Create an RDD of tuples or lists from the original RDD;

Create the schema represented by a Struct Type matching the structure of tuples or lists in the RDD created in the step 1.

Apply the schema to the RDD via create Data Frame method provided by Spark Session.

#### 3. SHARED & BROADCAST VARIABLES

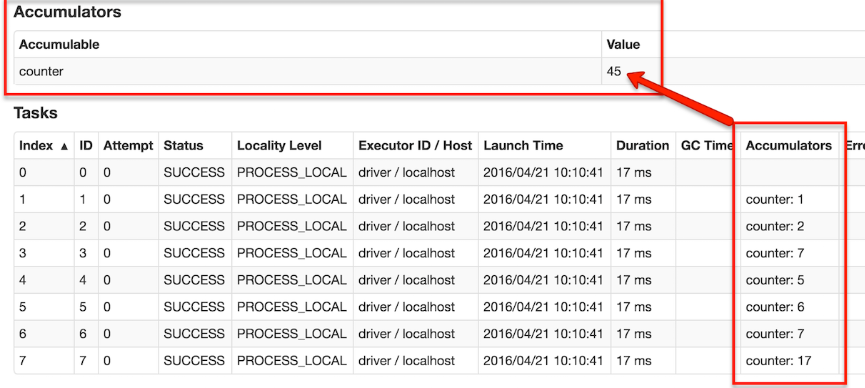


After the broadcast variable is created, it should be used instead of the value v in any functions run on the cluster so that v is not shipped to the nodes more than once. In addition, the object v should not be modified after it is broadcast in order to ensure that all nodes get the same value of the broadcast variable (e.g. if the variable is shipped to a new node later).

#### ACCUMULATORS

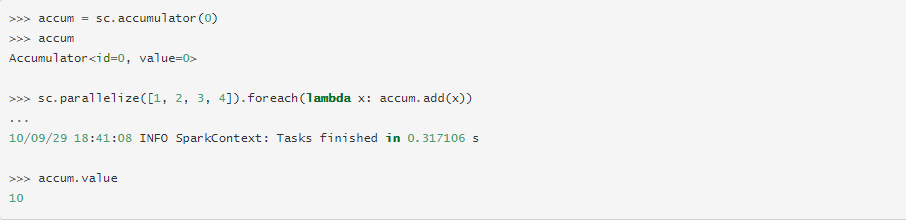
Accumulators are variables that are only “added” to through an associative and commutative operation and can therefore be efficiently supported in parallel. They can be used to implement counters (as in MapReduce) or sums. Spark natively supports accumulators of numeric types, and programmers can add support for new types.

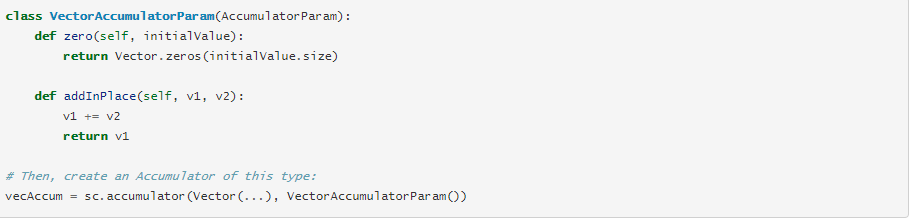
As a user, you can create named or unnamed accumulators. As seen in the image below, a named accumulator (in this instance counter) will display in the web UI for the stage that modifies that accumulator. Spark displays the value for each accumulator modified by a task in the “Tasks” table.



An accumulator is created from an initial value v by calling SparkContext.accumulator(v). Tasks running on a cluster can then add to it using the add method or the += operator. However, they cannot read its value. Only the driver program can read the accumulator’s value, using its value method.

The code below shows an accumulator being used to add up the elements of an array:

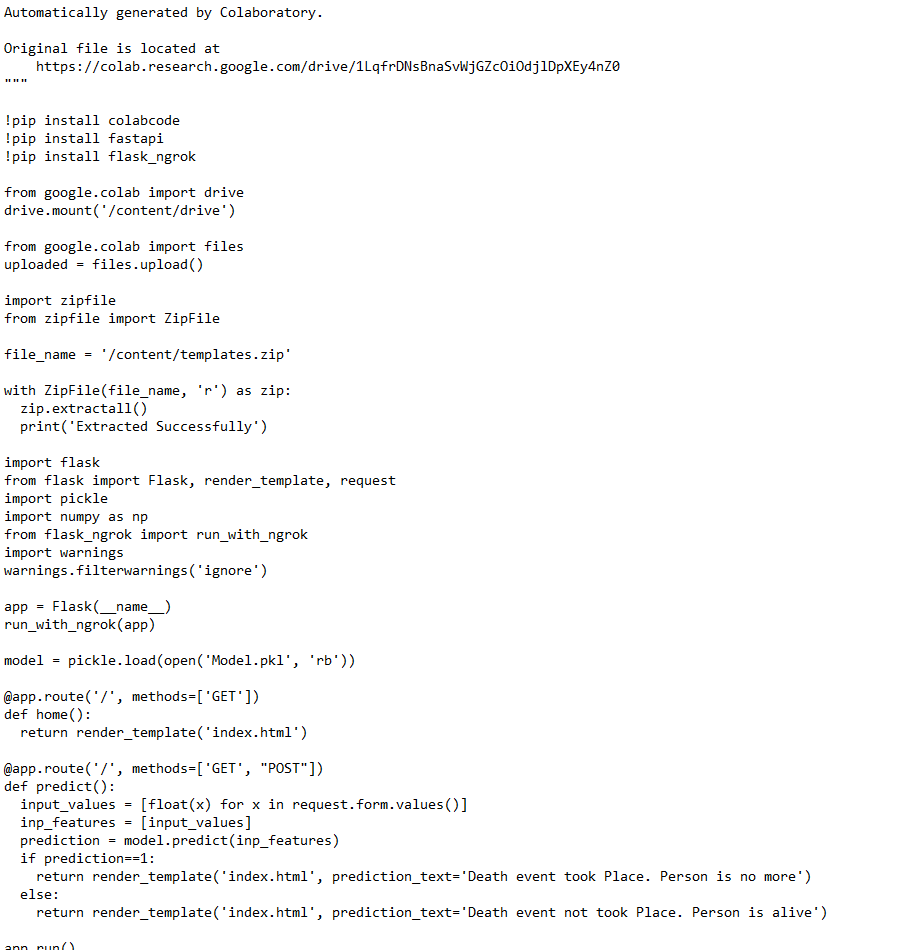




For accumulator updates performed inside actions only, Spark guarantees that each task’s update to the accumulator will only be applied once, i.e. restarted tasks will not update the value. In transformations, users should be aware of that each task’s update may be applied more than once if tasks or job stages are re-executed.

#### DEPLOYMENT IN MACHINE LEARNING MODELS

Model deployment is the process of putting machine learning models into production. This makes the model's predictions available to users, developers or systems, so they can make business decisions based on data, interact with their application (like recognize a face in an image) and so on



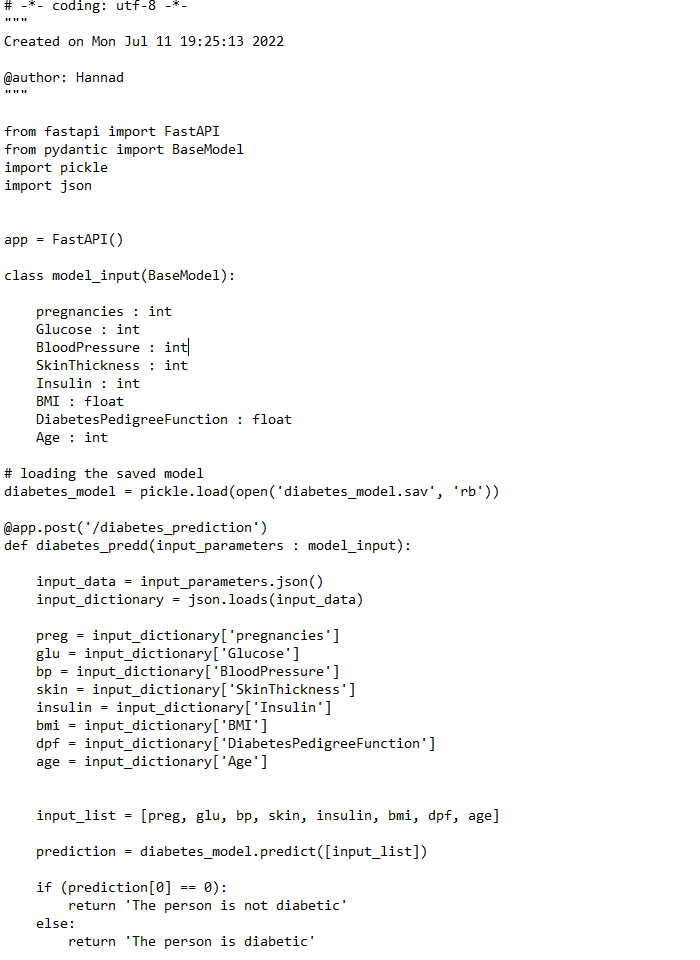
The simplest way to deploy a machine learning model is to create a web service for prediction. In this example, we use the Flask web framework to wrap a simple random forest classifier built with scikit-learn

It usually required four steps:

1. Develop and create a model in a training environment.
2. Test and clean the code ready for deployment.
3. Prepare for container deployment.
4. Plan for continuous monitoring and maintenance after machine learning.
5. 

Once a model has been trained and its results have been deemed successful, it needs to be validated to ensure that its one-time success was not an anomaly. Validation includes testing the model on a fresh data set and comparing the results to its initial training. In most cases, several different models are trained, but only a handful are successful enough to be validated. Of those that are validated, usually only the most successful model is deployed.

Validation also includes reviewing the training documentation to ensure that the methodology was satisfactory for the organization and that the data used corresponds to the requirements of end users. Much of this validation is often for regulatory compliance or organizational governance requirements, which may, for example, dictate what data can be used and how it must be processed, stored and documented.

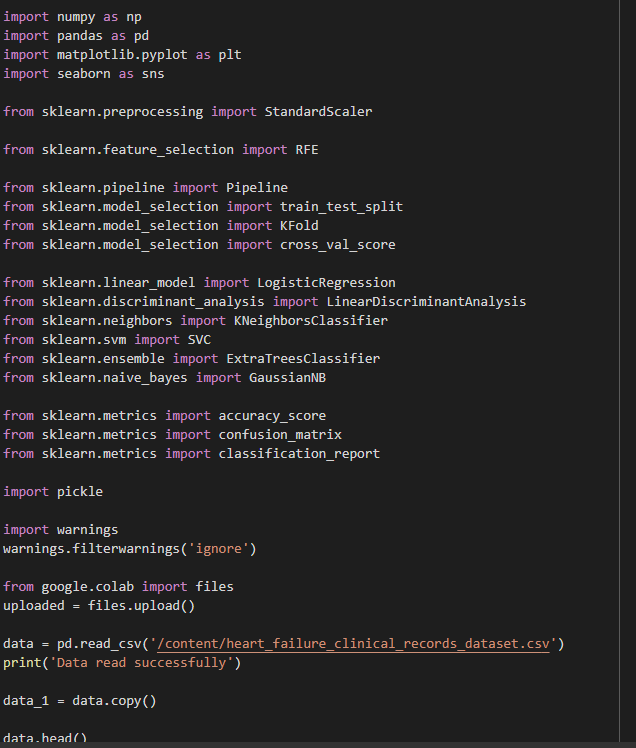


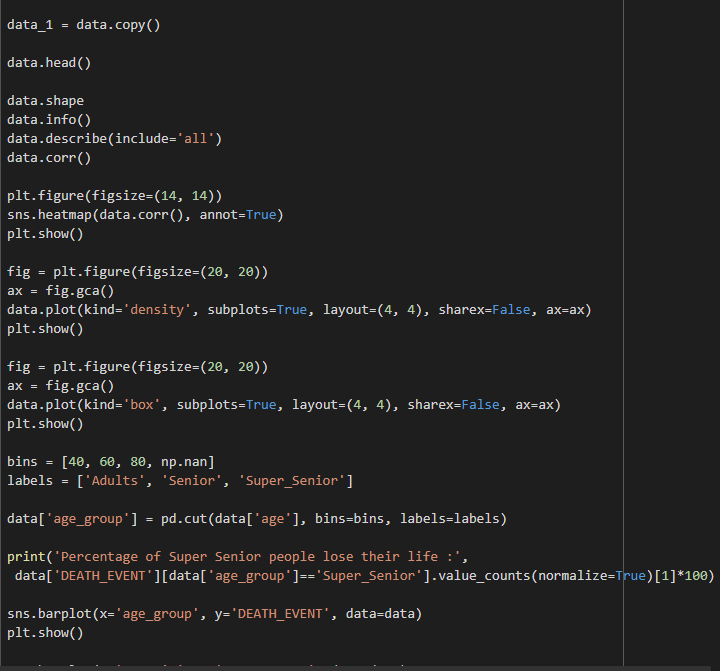
# DATA SCIENCE

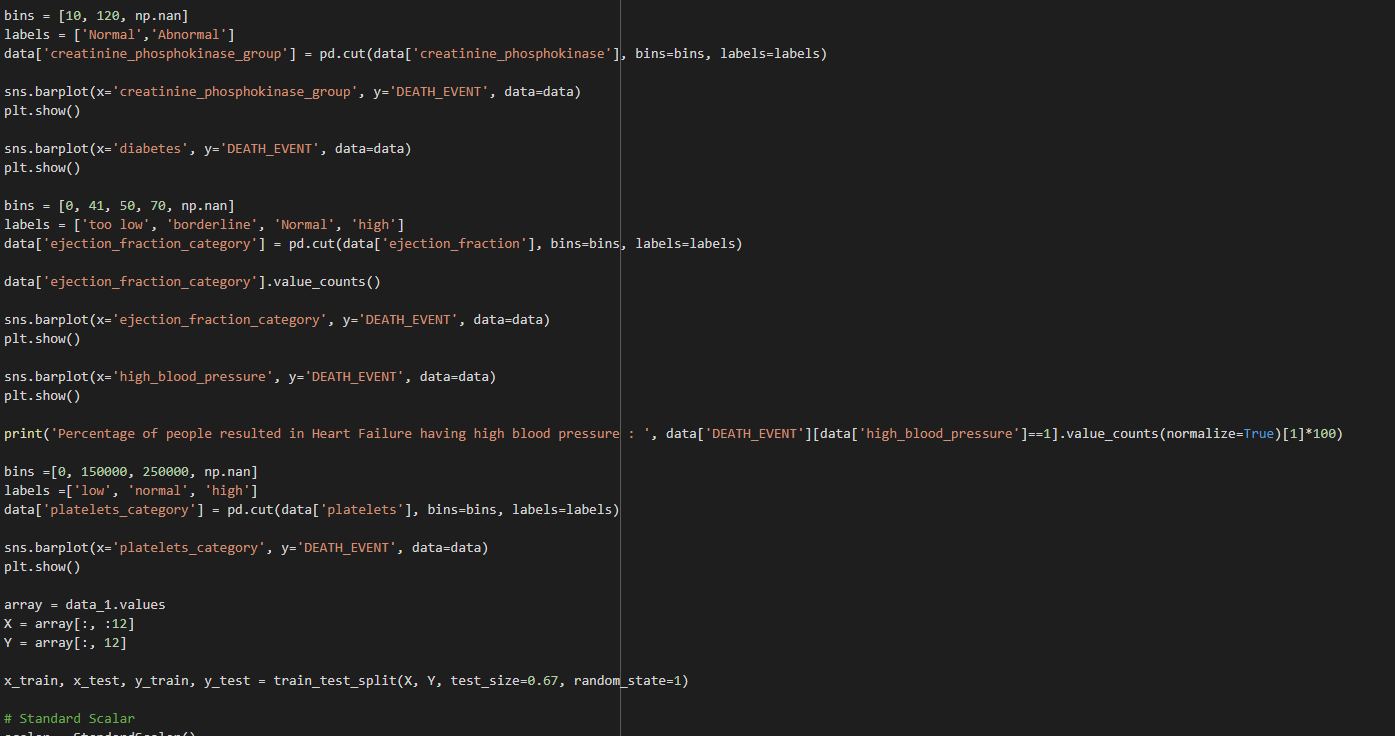
Data science is defined as applying advanced techniques the raw data in order to generate useful information from it.

The information which is extracted from that raw data comes handy in making decision making.

Apart from these uses, data science is used in marketing, finance, and human resources, healthcare, government programmers, and any other industry that generates data. Marketing departments use data science to determine which product is most likely to sell



NumPy is a library for Python that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Pandas is a high-level data manipulation tool that is built on 

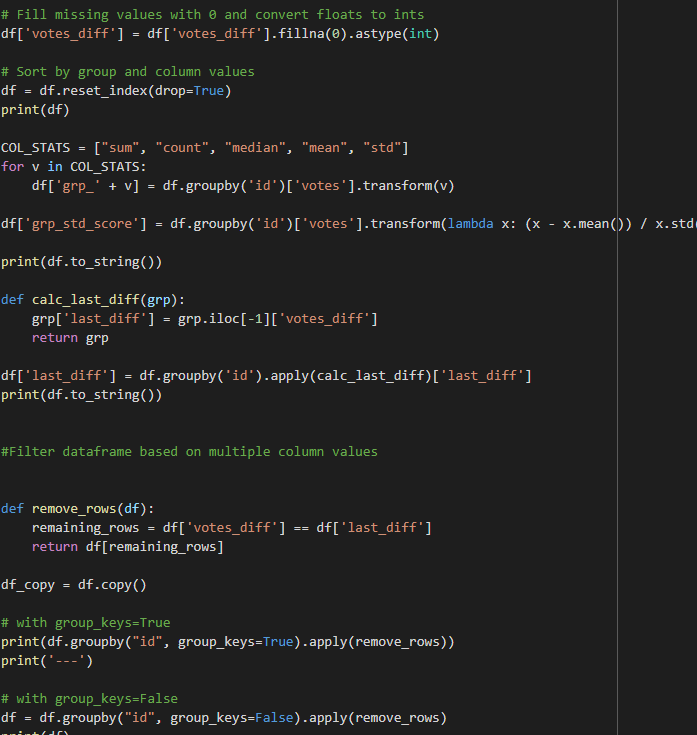


Basically, there are 2 rules of Broadcasting to remember:

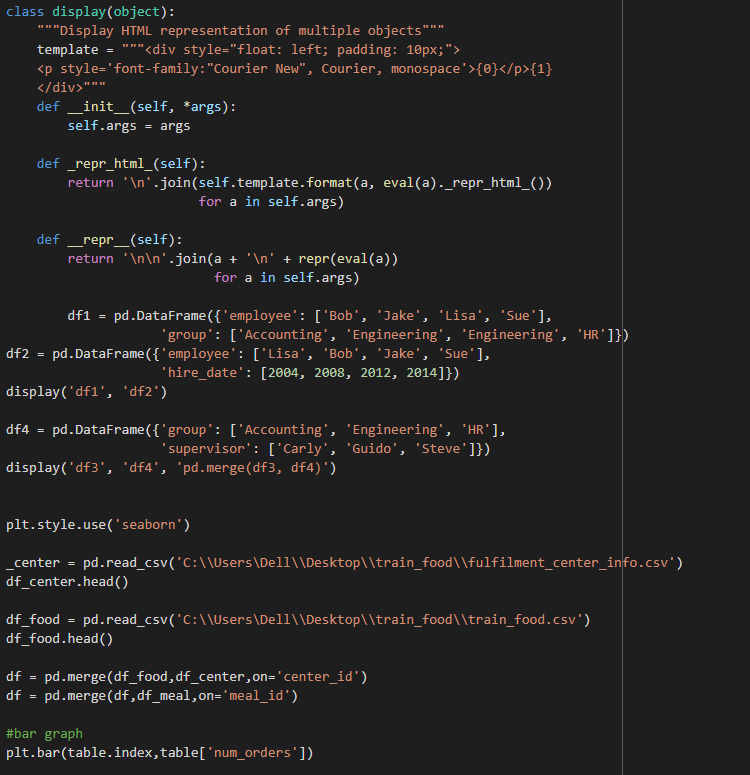
For the arrays that do not have the same rank, then a 1 will be prepended to the smaller ranking arrays until their ranks match. For example, when adding arrays, A and B of sizes (3,3) and (,3) [rank 2 and rank 1], 1 will be prepended to the dimension of array B to make it (1,3) [rank=2]. The two sets are compatible when their dimensions are equal or either one of the dimensions is 1.

When either of the dimensions compared is one, the other is used. In other words, dimensions with size 1 are stretched or “copied” to match the other. For example, upon adding a 2D array A of shape (3,3) to a 2D array B of shape (1, 3). NumPy will apply the above rule of broadcasting. It shall stretch the array B and replicate the first row 3 times to make array B of dimensions (3,3) and perform the operation.

NumPy provides basic mathematical and statistical functions like mean, min, max, sum, prod, std, var, summation across different axes, transposing of a matrix, etc.



As data comes in many shapes and forms, pandas aims to be flexible with regard to handling missing data. While NaN is the default missing value marker for reasons of computational speed and convenience, we need to be able to easily detect this value with data of different types: floating point, integer, boolean, and general object. In many cases, however, the Python None will arise and we wish to also consider that “missing” or “not available” or “NA”.



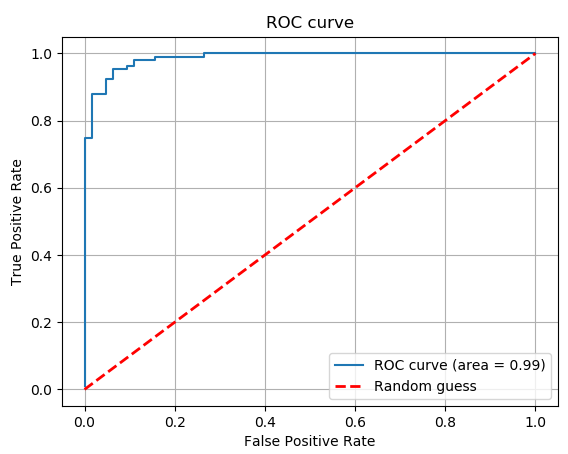
If you are using Matplotlib from within a script, the function plt.show() is your friend. plt.show() starts an event loop, looks for all currently active figure objects, and opens one or more interactive windows that display your figure or figures.

ROC curves typically feature true positive rate on the Y axis, and false positive rate on the X axis. This means that the top left corner of the plot is the “ideal” point - a false positive rate of zero, and a true positive rate of one. This is not very realistic, but it does mean that a larger area under the curve (AUC) is usually better.

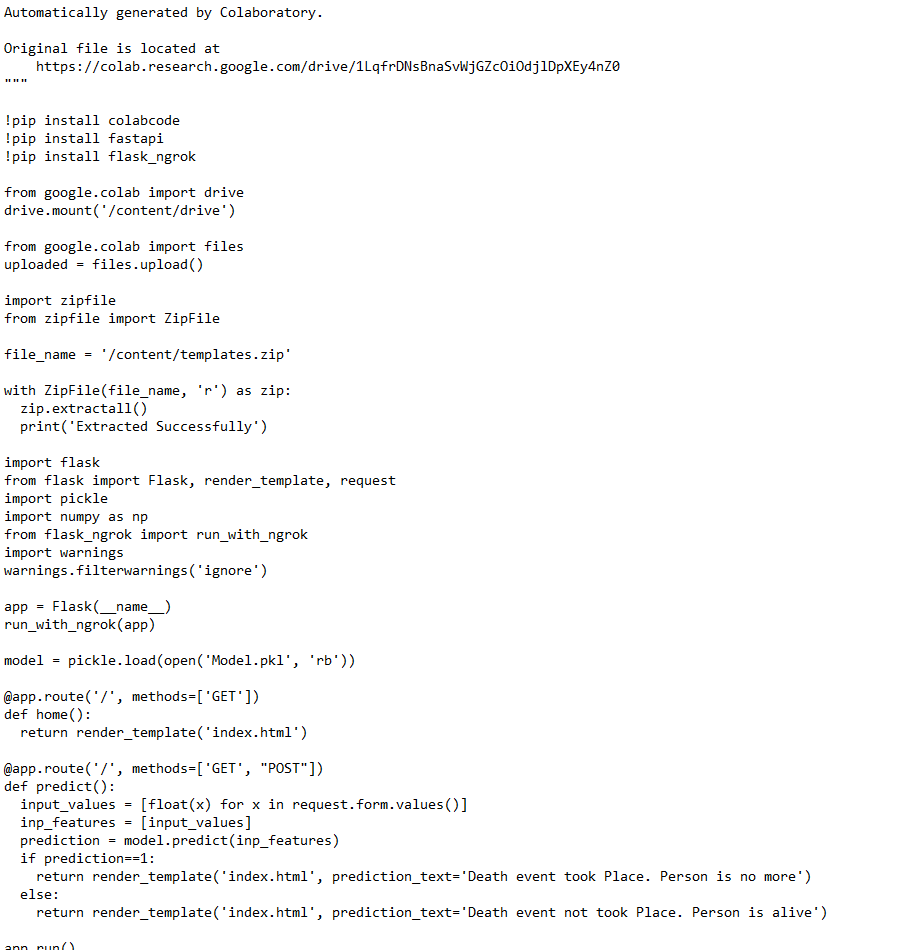
The “steepness” of ROC curves is also important, since it is ideal to maximize the true positive rate while minimizing the false positive rate.

ROC curves are typically used in binary classification to study the output of a classifier. In order to extend ROC curve and ROC area to multi-label classification, it is necessary to binarize the output. One ROC curve can be drawn per label, but one can also draw a ROC curve by considering each element of the label indicator matrix as a binary prediction (micro-averaging).

* ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.
* Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.
* ROC curves are appropriate when the observations are balanced between each class, whereas precision-recall curves are appropriate for imbalanced datasets.



Model deployment is the process of putting machine learning models into production. This makes the model's predictions available to users, developers or systems, so they can make business decisions based on data, interact with their application (like recognize a face in an image) and so on



The simplest way to deploy a machine learning model is to create a web service for prediction. In this example, we use the Flask web framework to wrap a simple random forest classifier built with scikit-learn

It usually required four steps:

1. Develop and create a model in a training environment.
2. Test and clean the code ready for deployment.
3. Prepare for container deployment.
4. Plan for continuous monitoring and maintenance after machine learning.
5. 

Once a model has been trained and its results have been deemed successful, it needs to be validated to ensure that its one-time success was not an anomaly. Validation includes testing the model on a fresh data set and comparing the results to its initial training. In most cases, several different models are trained, but only a handful are successful enough to be validated. Of those that are validated, usually only the most successful model is deployed.

Validation also includes reviewing the training documentation to ensure that the methodology was satisfactory for the organization and that the data used corresponds to the requirements of end users. Much of this validation is often for regulatory compliance or organizational governance requirements, which may, for example, dictate what data can be used and how it must be processed, stored and documented.

