# Detailed analysis of the code

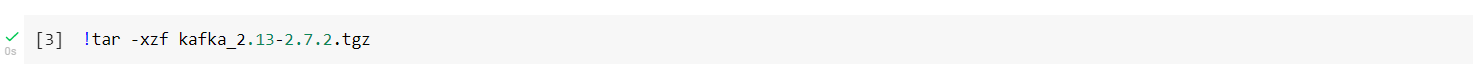
Initially, we install Kafka python to access Kafka functions from the python environment.



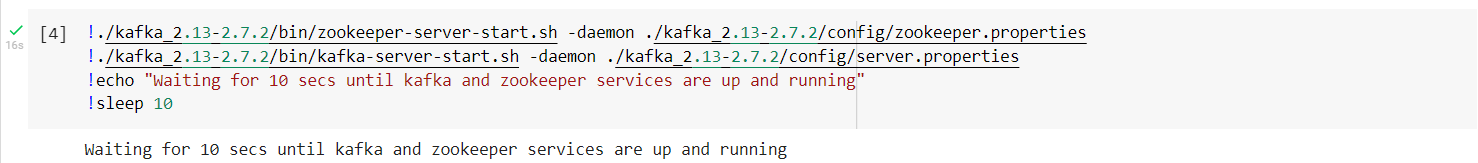
Then we install TensorFlow and TensorFlow-io packages, which are used to create deep learning models.



We use the below command to extract the Kafka file that we have downloaded from canvas.



Now, we need to set up both Kafka and Zookeeper instances by using the below commands. Kafka used the localhost:9092 port by default and zookeeper uses port 2181.



Once, the Kafka and zookeeper servers are started, we create the Kafka topics with the below specifications:

* susy-train: partitions=1, replication-factor=1
* susy-test: partitions=2, replication-factor=1

A picture containing application

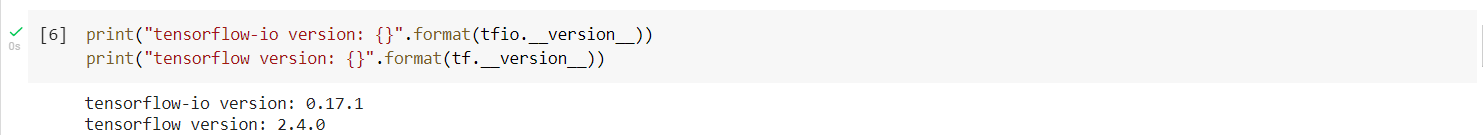
Description automatically generated

Now we import all the required libraries needed, mainly the Kafka Producer, pandas, and TensorFlow libraries.

Graphical user interface, application

Description automatically generated with medium confidence

We print the TensorFlow versions.



Here, below are the column names of SUSY data, that we shall use for classification. It consists of 19 attributes.

Text

Description automatically generated with medium confidence

Now we load the data from the SUSY file into the pandas data structure.

A picture containing graphical user interface

Description automatically generated

We print the number of rows and columns of SUSY data.

Graphical user interface

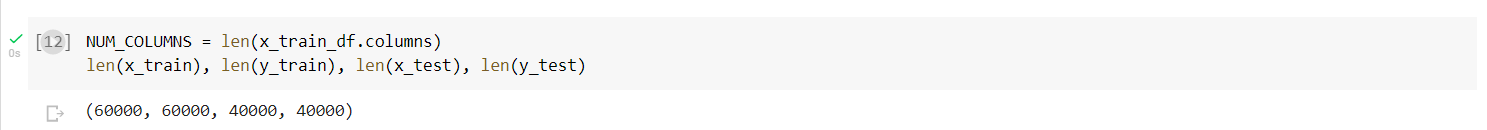
Description automatically generated with medium confidence

Now we split the dataset and drop the column that we predict. Now the data is stored in partitions which can later be used by the consumer groups for retrieving data efficiently.

Text

Description automatically generated

Now we print the size of the training and test data.



Now we store the training and testing data in Kafka. Storing the data in Kafka simulates an environment for continuous remote data retrieval for training and inference purposes using the producer.

Graphical user interface, text

Description automatically generated

Here, we are streaming the data from Kafka into TensorFlow.

Text

Description automatically generatedWe now build and model to train. We are using adam optimizer here with 10 epochs.

A picture containing application

Description automatically generated

A less complex neural network has been used with 4 layers, 3 with activation function relu and the last layer with activation function sigmoid, the complexity of the model can be increased by modifying the learning strategy, tuning hyper-parameters, etc.

Table

Description automatically generated

The model is now compiled. For losses, binarycorsenthropy is used and metrics are ‘accuracy’.

Graphical user interface, text

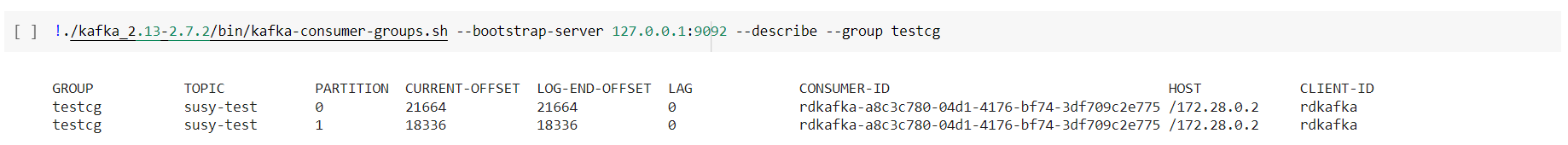
Description automatically generated

To infer the test data streaming. The kafkaFroupIODataset function is used. Once all the messages are read from Kafka and the latest offsets are committed, the consumer does not read the messages from the beginning. It is possible to train for a single epoch only with data inputs during the training phase, once a data point is consumed by the model it can be discarded.

We also evaluate the performance of the test data. Because it is an ‘exactly-once’ semantics, test data cannot be reused, to run inference on test data new consumer groups should be used every time.

Graphical user interface, text, application, email

Description automatically generated



In online learning, the data once consumed by the model may not be available for training again.

When all of the messages are consumed from the topics, the dataset disconnects from Kafka after waiting for 10sec, within these 10 sec time frames if any new data is received, the processes of training and data consumption continues.

Graphical user interface, application

Description automatically generated

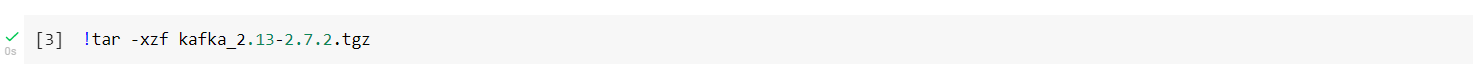
The incrementally trained model can be saved in a periodic fashion and can be utilized to infer the test data in either online or offline modes.

Graphical user interface, text, application

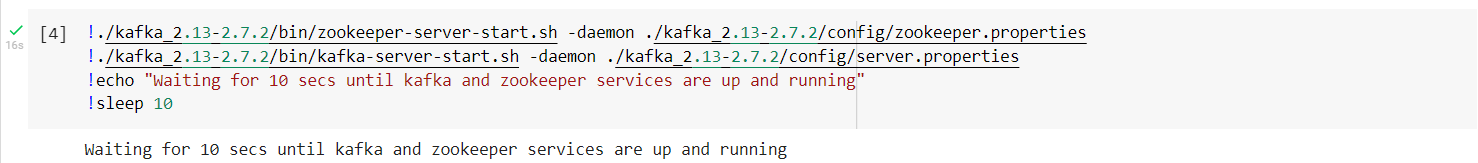
Description automatically generated

# How did you execute the task using Kafka, and why is Kafka important in this machine-learning model?

We use the below command to extract the Kafka file that we have downloaded from canvas.



Now, we need to set up both Kafka and Zookeeper instances by using the below commands. Kafka used the localhost:9092 port by default and zookeeper uses port 2181.



Once, the Kafka and zookeeper servers are started, we create the Kafka topics with the below specifications:

* susy-train: partitions=1, replication-factor=1
* susy-test: partitions=2, replication-factor=1

A picture containing application

Description automatically generated

Kafka allows data from various sources to be written into it, for the above code we have used the SUSY dataset, we have used Kafka and created topics to store training and testing data, to infer the test data streaming and we have used Kafka to stream online. Online learning is different from the traditional training of the models where the model learns stepwise by repeating the process with fixed datasets and the model iterating over the same dataset multiple times. Whereas in online learning, the data once consumed by the model may not be available for training again.

The Kafka ecosystem helps in different ML use cases for**model training, model serving, and model monitoring.** Kafka is a middle layer between the datasets, the environment where the model fits, and the actual application that is used for real-time predictions.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated