# UE19CS322 Big Data Project 2

# Machine Learning with Spark Streaming

**Machine Learning with Spark Streaming** is one of the project titles that can be taken up as a part of the UE19CS322 Big Data course at PES University. This simulates a real world scenario where you will be required to handle an enormous amount of data for predictive modelling. The data source is a stream and your application faces the constraint of only being able to handle batches of a stream at any given point in time.

With this project, you will learn more about how applications in the real world modify their algorithms to work on large data streams and how incremental processing can be leveraged to process and analyze streams over time.

The files required for the project can be found here. **You must fill up this form and submit the link to your project's repository on GitHub** for your project's evaluation.

## Project Objectives and Outcomes

1. This project will help students obtain an in-depth understanding of how applications in the real world work with large data streams
2. At the end of this project, the student will be able to analyze large data streams and process them for machine learning tasks using Spark Streaming and Spark MLlib to draw insights and deploy models for predictive tasks

## Ethical practices

Please submit original code only. You can discuss your approach with your friends but you must write original code. All code must be submitted through the portal. **We will perform a plagiarism check on the code and you will be penalised if your code is found to be plagiarised**

## Software/Languages to be used:

1. Python `3.8.x`
2. Hadoop `3.2.2` only
3. Spark `3.1.2` only

You are allowed to use only the following libraries

1. `pyspark`
2. `numpy`

All statistical and machine learning algorithms must either be imported from `pyspark.ml` or implemented from scratch. Using `pandas`, `scikit-learn`, `tensorflow` and `pytorch` is **not permitted**.

## Marks

15 marks

## Submission Link

Portal for Big Data Assignment Submissions

## Submission Deadline

3rd December 11:59 PM, 2021

## Project Guidelines

1. Your code will **not be executed on the portal**. However, it will be used to **perform a plagiarism check later** on.
2. You are free to use any libraries and APIs to implement the project
3. **There is no fixed guideline on how you may implement the project**. The design and architecture is completely upto you. There are no constraints on the implementation, however your code must demonstrate all the required working functionalities.
4. Additionally, for this project **you are allowed to pick any dataset for this task** or use the one provided by us. If you are choosing your own dataset, please contact any of the TAs and get it verified first. The only constraints on choosing the dataset are as follows:
    - The dataset is for **classification only**
    - The dataset has *atleast* 5000 examples
5. You will be required to **create a GitHub repository and add the Teaching Assistants as collaborators** to it. You have time until your team's presentation to add us to the repository failing which your team will **not be evaluated**. You can find the usernames to add below:
    - aditeyabaral

    - anshsarkar
    - Visheeeee
    - pvn-leo
6. **You must fill up this form and submit the link to your project's repository on GitHub**. Failing to fill up this form will lead to your project not getting evaluated.
7. **You are required to make regular commits to your repository**. Your progress on the project is tracked periodically. Failing to make regular commits will lead to you being penalised.
8. You are highly suggested to structure the project well by using multiple files (but not too many) structured in different folders. You can visit this link to learn about how to structure your repository well.
9. You will be required to **present your implementation to the faculty as a part of the viva**
10. **Do note that any member of your team may be required to demonstrate any feature, or make changes to the configuration and re-run your project**. Ensure that the project runs on every single team member's system.
11. Convert line breaks in DOS format to Unix format (**this is necessary if you are coding on Windows -** your code will not run on our portal otherwise)

    ```
    dos2unix filename.py
    ```

## Task Specifications

Do ensure that during implementation, functionalities that are dependent are bundled into a process that can be executed on the shell. You may have as many helper modules as necessary.

**This project is open-ended**, which means that *there is no correct answer*. You will be evaluated on how you choose to go about each phase, what step you perform in each phase, the level and clarity of analysis performed while creating your models and on obtaining results and your conclusions you draw. **You will be graded on your approach and how you apply the tools provided by Spark MLlib** at your disposal rather than on the final scores you obtain on your dataset.

## Fetching Data and Preprocessing

In this section, you will be required to receive the stream of input and preprocess them using standard techniques. Since the data is enormous and is being generated on the fly, you will be receiving records in batches with the `batch_size` being a configurable variable.

1. The size of each batch, `batch_size`, will be known to you at runtime and will be provided as a command line argument to your process. This value will remain constant throughout a single run of the process but can be changed for different experimental runs.
2. Data can be streamed using the `stream.py` file. The script will contain a parameter defined named `BATCH_SIZE` which will control the number of examples in each batch. You are suggested to experiment with the batch size and find **how the batch size affects performance at each stage of the project** – preprocessing, model building and so on.
3. **After every batch of data is received, you are required to preprocess them using standard statistical techniques** to scale and normalize the values. These include the following (but not limited to):
    - MinMax Scaling
    - Normalization
4. The **type of the dataset will determine the type as well as the level preprocessing** being performed. For example, if you choose to work with the dataset (Fashion MNIST) provided, you will have to look into computer vision based techniques such as
    - Convolutions
    - Pooling
    - Masking
    - Greyscale
5. You can either write custom code snippets to perform preprocessing, or use the functions already included as a part of Spark MLlib.
6. Depending on the dataset chosen, you also need to look into the respective **feature selection techniques and dimension reduction** that Spark MLlib supports to extract features that offer the most importance
    - Compute the variance explained by each feature in your dataset
    - Extract and retain only those features that provide the most variance
7. Depending on the dataset chosen, you may also be required to carry out feature transformation techniques to convert non-numerical data types in numerical data.
8. For each preprocessing step performed, **you will be required to explain the reason for carrying it out, as well as provide examples of how it improved performance** in the models you build in the later section.
9. **Optional**: Use plots to show how each preprocessing step makes a difference in the spread nd distribution of your data and compare performance of your models across varying levels of preprocessing performed

## Building the Model

In this section, you will be required to build machine learning models and fit them on the dataset. Since the entire dataset is being streamed in batches, you will have to resort to incremental learning to allow your model to learn effectively.

## Learning from Data

1. After each batch has been preprocessed, it needs to be fed into a machine learning model for classification
2. **You cannot store all batches in memory**. At every instant of time, the model must be re-fit on the input batch such that it learns incrementally, one batch at a time.
3. You must create different classifiers, and fit all of them at the same time on the input batch. **You are required to implement atleast 3 classifiers** and analyse the performance provided by each. You are further required to use different types of classifiers to understand which type of classifier suits your data the best.
4. **Hyperparameter tuning must be performed to obtain the best possible score on your dataset**. Which hyperparameters provide maximum changes to your model's performance? Why? These are questions your experiments must answer. You may even use suitable plots to show how your performance improved with each tuning experiment.
5. **Experiment with the training batch size** to understand if it plays an important role in predictive modelling. Does increasing the batch size increase performance? Obtain the performance for each of your classifiers on different batch sizes and present them.
6. **Optional**: Evaluate on the training set and find out how the performance on that training batch varies as the batch size increases. Vary the batch size and obtain the performance of your models on each training batch. Present your performance using suitable plots.

## Testing your Model

1. Similar to how the input data is streamed, the test data will also be streamed in batches. **The size of the input stream may not be the same as that of the training data**.
2. For each batch, perform predictions using all the models that you have built.
3. At the end of the stream, compute the performance of each classifier using appropriate metrics such as:
    - Confusion Matrix
    - F-1
    - Accuracy
    - Precision

    - Recall

4. **Analyze the difference in the performance between the different types of classifiers** (such as linear vs non-linear) and explain the differences. You may use suitable plots to explain the predictions made by your classifiers
5. **Make plots of your predictions to compare across different classifiers, hyperparameter choices and batch sizes**. While there is no hard limit on the number of plots you create, you are required to create 3 plots per classifier to compare performance and 3 plots showing 3 experiments with the training batch size.
6. **Optional**: Experiment with the batch size of the test data to understand if the batch size plays a role while making predictions. You will be required to explain your findings, if any
7. **Optional**: Create random shuffles of the entire dataset to set up cross-validation sets consisting of train and test batches. Evaluate and tune your models on each set of train and test batches and analyse the performance across them.

## Clustering

In this section, you will be required to implement a clustering algorithm on your dataset and analyze the clusters obtained.

1. Similar to the previous sections, your data will be streamed in batches. However, this time **the data will be streamed in an endless loop**, i.e the first batch will again be streamed after all batches have been streamed.
2. You must **fit your clustering algorithm on each batch of input**, such that it learns incrementally.
3. Once all batches have been streamed once, compare the centroids obtained in the current iteration with the previous iteration. If the centroids shift by a very small value, stop receiving input from the stream.
4. Set the value of the number of clusters to be the number of classes present in the dataset. Analyse if all the examples in the same class from the test dataset also fall into the same cluster. Answer the question, “Is clustering a form of unsupervised classification?”
5. **Optional**: Experiment with different batch sizes and different values for the hyperparameters (such as the number of clusters). Do you observe any interesting patterns? Draw conclusions if any.