3.1 Project title chosen: **Machine Learning with Spark MLlib.**

Dataset chosen: spam/ham classification

3.2 Design details: sklearn is the main module used to perform batch wise processing. The modules, required for preprocessing techniques, we use are Tokenizer, StopWordsRemover, RegexTokenizer and CountVectorizer imported from pyspark.ml. We use GaussianNB, SGDClassifier, MultinomialNB, MiniBatchKmeans and PassiveAggressiveClassifier modules from sklearn for various classification and clustering techniques. All These models can perform online updates to model parameters via partial\_fit. We implement joblib in order to save our trained model. Joblib is a set of tools to provide lightweight pipelining in Python. In particular, joblib offers: transparent disk-caching of the output values and lazy re-evaluation (memorize pattern) easy simple parallel computing.

The design idea consists of dividing the processing technique into two files, train file and the test file. The train file processes the data in batches, does preprocessing and trains the data using the specific model. We utilize the joblib dump functionality to save our trained model. Then we perform the testing process through the test file wherein we utilize the dumped joblib file and perform prediction. Various models of classification and clustering follow the same procedure

3.3 Surface level implementation details about each unit:

Preprocessing: Preprocessing involves removing the null values, tokenization of the messages, removal of stop words. We also run iteration to remove numeric, alphanumeric and punctuational words. Then we perform count vectorization to complete our preprocessing. This preprocessing scheme is generalized for all models.

Models:

1.Gaussian Naïve Bayes: The dataframe obtained from preprocessing is utilized. The GaussianNB model is initialized using the function and is stored in a variable. A joblib file called ‘gaussianNB’ is created for storing the model and is loaded to the variable. We now implement the partial fit function and save it onto the joblib file. The partial fit performs the training and this trained model is stored on the disk through the joblib file. This file is then loaded and used for prediction in the test file

1.Multinominal Naïve Bayes: The dataframe obtained from preprocessing is utilized. The MultinomialNB model is initialized using the function and is stored in a variable. A joblib file called ‘multinomialNB’ is created for storing the model and is loaded to the variable. We now implement the partial fit function and save it onto the joblib file. The partial fit performs the training and this trained model is stored on the disk through the joblib file. This file is then loaded and used for prediction in the test file

2.Gaussian Naïve Bayes: The dataframe obtained from preprocessing is utilized. The GaussianNB model is initialized using the function and is stored in a variable. A joblib file called ‘gaussianNB’ is created for storing the model and is loaded to the variable. We now implement the partial fit function and save it onto the joblib file. The partial fit performs the training and this trained model is stored on the disk through the joblib file. This file is then loaded and used for prediction

3.Passive Aggressive: The dataframe obtained from preprocessing is utilized. The PassiveAggressiveClassifier model is initialized with parameters as max\_iter=50, using the function and is stored in a variable. A joblib file called ‘passiveAggresive’ is created for storing the model and is loaded to the variable. We now implement the partial fit function and save it onto the joblib file. The partial fit performs the training and this trained model is stored on the disk through the joblib file. This file is then loaded and used for prediction

4.K-means clustering: The dataframe obtained from preprocessing is utilized. The MiniBatchKMeans model is initialized (n\_clusters=2, init='k-means++', n\_init=1, init\_size=250, batch\_size=250, )using the function and is stored in a variable. A joblib file called ‘km’ is created for storing the model and is loaded to the variable. We now implement the partial fit function and save it onto the joblib file. The partial fit performs the training and this trained model is stored on the disk through the joblib file. This file is then loaded and used for prediction

3.4 Reason behind design decisions: The various functions utilized are those that can be processed through partial fit. Through partial fit, All These models can perform online updates to model. Joblib is utilized for structuring the processing performed

3.5 Takeaway from the project: Streaming and processing are gargantuan procedures. It was possible for us to understand how these processes work through practical implications. We were also able to understand the various advantages that streaming has over normal processing such as real time analytics, low latency and faster crisis response. Instead of storing huge chunks of data, which is practically not feasible many a times, this functionality makes our life easier.

The project improved our understanding of various machine learning models and also helped us appreciate modules like sklearn better.