Big Data Project – Machine Learning with Spark Streaming

Done By -: B.Pravena - PES2UG19CS076
Rishi Patel - PES2UG19CS329
Swarnamalya A S - PES2UG19CS418
Varna Satyanarayana - PES2UG19CS448

Content:

Project Title - San Francisco Crime Classification

An open-ended project to learn and then train a model to classify crime which is a single vector of size 9 in the dataset to its corresponding label/category. There are 878k records in the train and test dataset.

Design Details

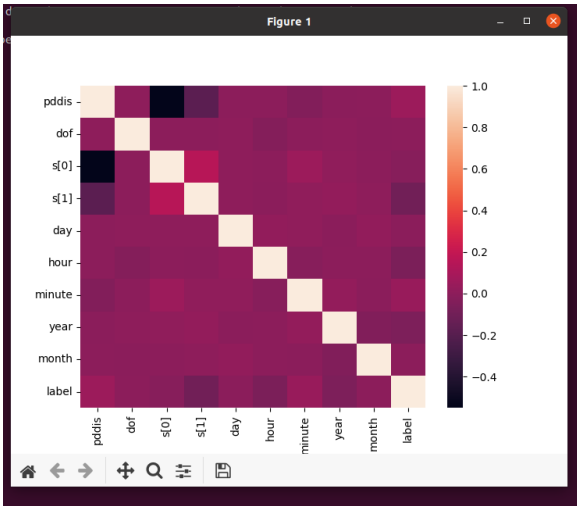
Streaming – To stream the batches, first run stream.py on a terminal and when it shows waiting run streamclient.py on the other terminal. This starts a spark context, session and then converts RDD to a dictionary and then to a dataframe.

Pre-processing - Since the dataset provided is already clean, there are no missing values. We use VectorAssembler for creating a vector of X & Y, and have normalized only the X and Y categories using MinMaxScaler as they are the only numeric categories in the dataset. This shifts and rescales the data such that they are between 0 and 1.
We then splitted the Dates column into Day, month , year, hour, minute and second.

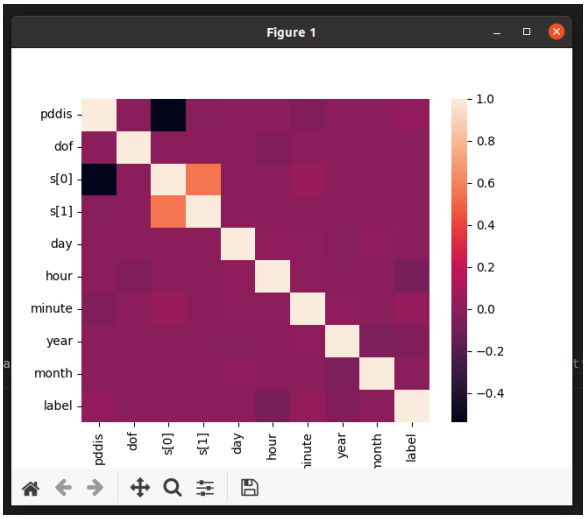
We have also removed Resolution and descript columns since they are not provided in the test dataset.

We then tried to find the correlation between the features and the target column. We did both the Pearson and Spearman correlation and obtained the following graphs as outputs.

Spearman -:



Pearson -:



We have dropped Address, seconds and month as they aren't related that much to the category of the crime committed.

Surface Level Implementation -:

Our goal is to develop (at least 3) a model that is trained incrementally and accurately identifies the type of crime in the test dataset.
We used MultiLayerPerceptron, Multinomial NaiveBayes and Stochastic Gradient Descent. We partially fit each of these models on each batch obtained while streaming.

MultiLayerPerceptron - This model uses a neural network that learns the relationship between linear and non-linear data.

Multinomial NaiveBayes - The classifier is suitable for classification with discrete features.

Stochastic Gradient Descent - It implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for classification. We have used “hinge” as loss function which supports linear SVM.

Reasons behind Design Decisions:

Streaming allows us to process data in real time and incrementally learn the model for each batch obtained.

We then compared the accuracies obtained from the three models mentioned by partially fitting the models on each batch.

Using StringIndexer, LabelEncoding etc for converting categorical to numeric assigns different values to each category for every batch that is streamed. In order to overcome this problem, we hard coded the values for converting the categorical data to continuous data so that every batch has the same numeric values assigned to each category. We then concatenated the final encoded dataset into our final dataframe.

We used MinMaxScaler as it's more useful than StandardScaler when upper and lower boundaries are known(latitude and longitude of SF) and when we need non-negative values after normalizing the dataset.

Take away from the project -:

We were able to get a deep understanding of huge data streams and process them using Machine Learning techniques in Spark to draw insights about the real world applications. We also got to experiment on many models of the SparkMLib and sklearn for incremental learning. We were also able to observe how using pyspark on multiple batches of data were streamed easily and quickly compared to using them as a single dataset in python.

Design Details

Streaming:

We execute stream file given with arguments in one terminal. Stream1.py in other terminal which will then stream the data and convert rdd to a dataframe.

Preprocessing:

Missing/Duplicate Values:

No missing values were identified in the training or test datasets.

The training dataset included 2,323 duplicate rows. It is unclear whether these values represent reporting errors or multiple instances of the same crime occurring at the same time and location. Regardless, these values were removed from the training data since they provide redundant information. The test dataset contained no duplicate rows.

Extraneous Features

Since the Descript and Resolution features are not found in the test dataset, these features will not prove useful for developing a predictive model. These features were removed from the training dataset during preprocessing.

Modeling algorithms require numerical input in order to function. Categorical data is often represented as a string of text that provides a qualitative description of the information it

represents. The process of converting a string representation of a variable to a numerical value is known as encoding.

Embeddings are vectors of numerical values that are used to represent strings of text. Word embeddings include a vector of numerical values for each individual word appearing in a string of text. These numerical values are used to translate information about the text into a format that our machine learning algorithms can understand and process.

Target Variable Encoding

Category data were processed differently depending on the algorithm used to construct the model. For tree-based algorithms, the Category feature was label encoded, while for neural networks, the Category feature was one hot encoded. One hot encoding requires creation of an individual column for each possible class of a categorical feature. For each observation, a

value of 1 is assigned to the column of the identified class while a value of 0 is assigned to all other class columns. Training and testing data were combined during the encoding process to improve the preprocessing efficiency.

Normalisation

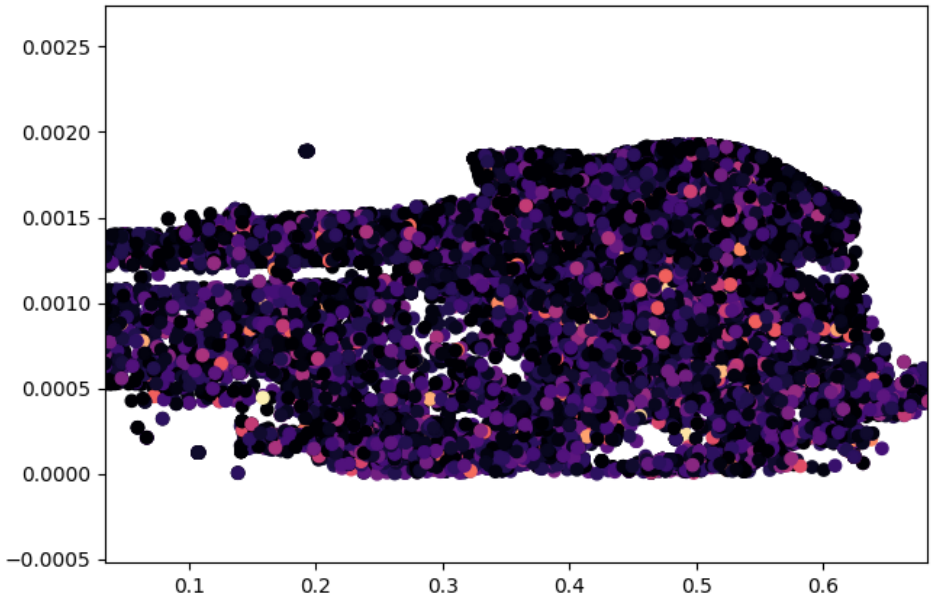
Data input to a neural network or other algorithms that use gradient descent as an optimization technique must be scaled prior to training. Two common methods of feature scaling are standardization and normalization. Normalization, also known as min-max scaling, shifts and rescales all data values so they are between 0 and 1.

Surface Level implementation

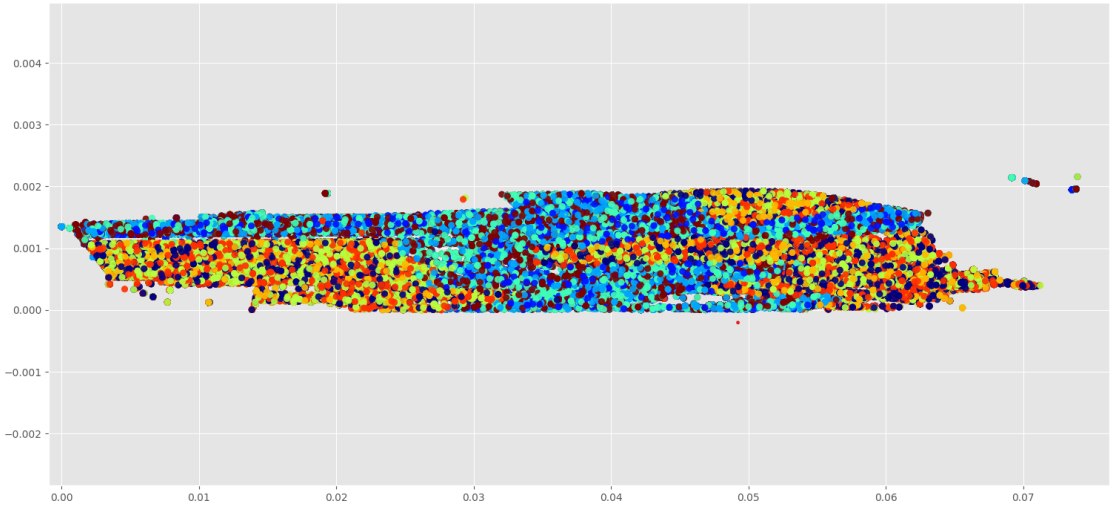
Prediction Approach

Our goal is to develop a model that accurately identifies the type of crime in the test dataset. This type of problem is considered a multiclass classification problem. In this instance, a total of 39 classes of crime category are represented in the training data.

Our model should assign a predicted probability to each of the 39 possible classes of crime category for each test observation, rather than simply predicting a single class. Selecting a single class is the equivalent of assigning a probability of 1 to that class and a probability of 0 to all other classes. If this approach is taken and the selected class is incorrect, the evaluation metric will assign the maximum penalty per observation. Using class probabilities as opposed to selecting a single class decreases the penalty we receive from the evaluation metric in the event that our prediction is incorrect. We used minmax scalar to normalize the data.


0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
−0.0005
0.1
0.2
0.3
0.4
0.5
0.6

Kmeans=8


0.004
0.003
0.002
0.001
0.000
−0.001
−0.002
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07

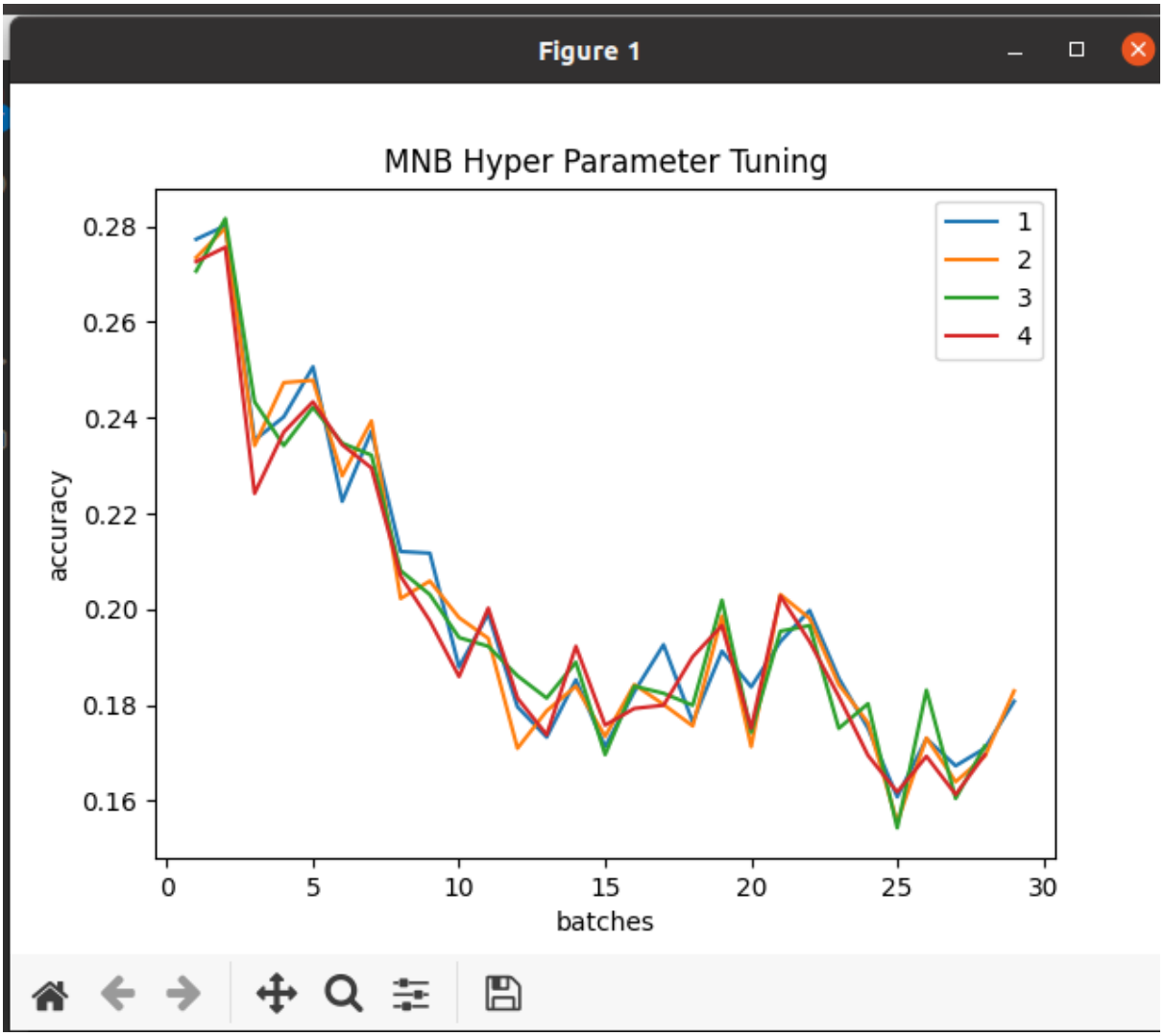


Figure 1
MNB Hyper Parameter Tuning
accuracy
batches
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0
5
10
15
20
25
30
1
2
3
4

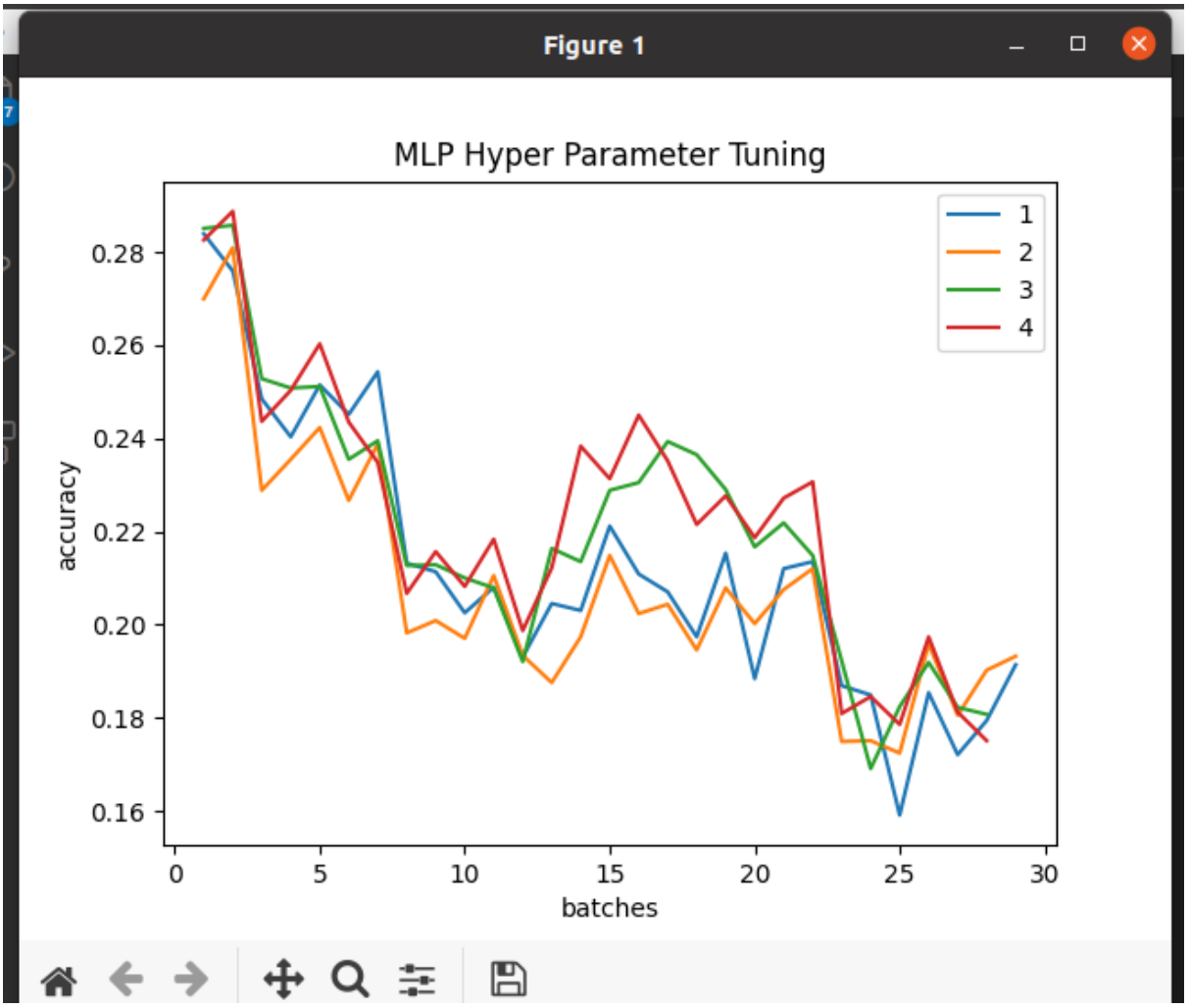


Figure 1
MLP Hyper Parameter Tuning
accuracy
batches
1
2
3
4
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0
5
10
15
20
25
30

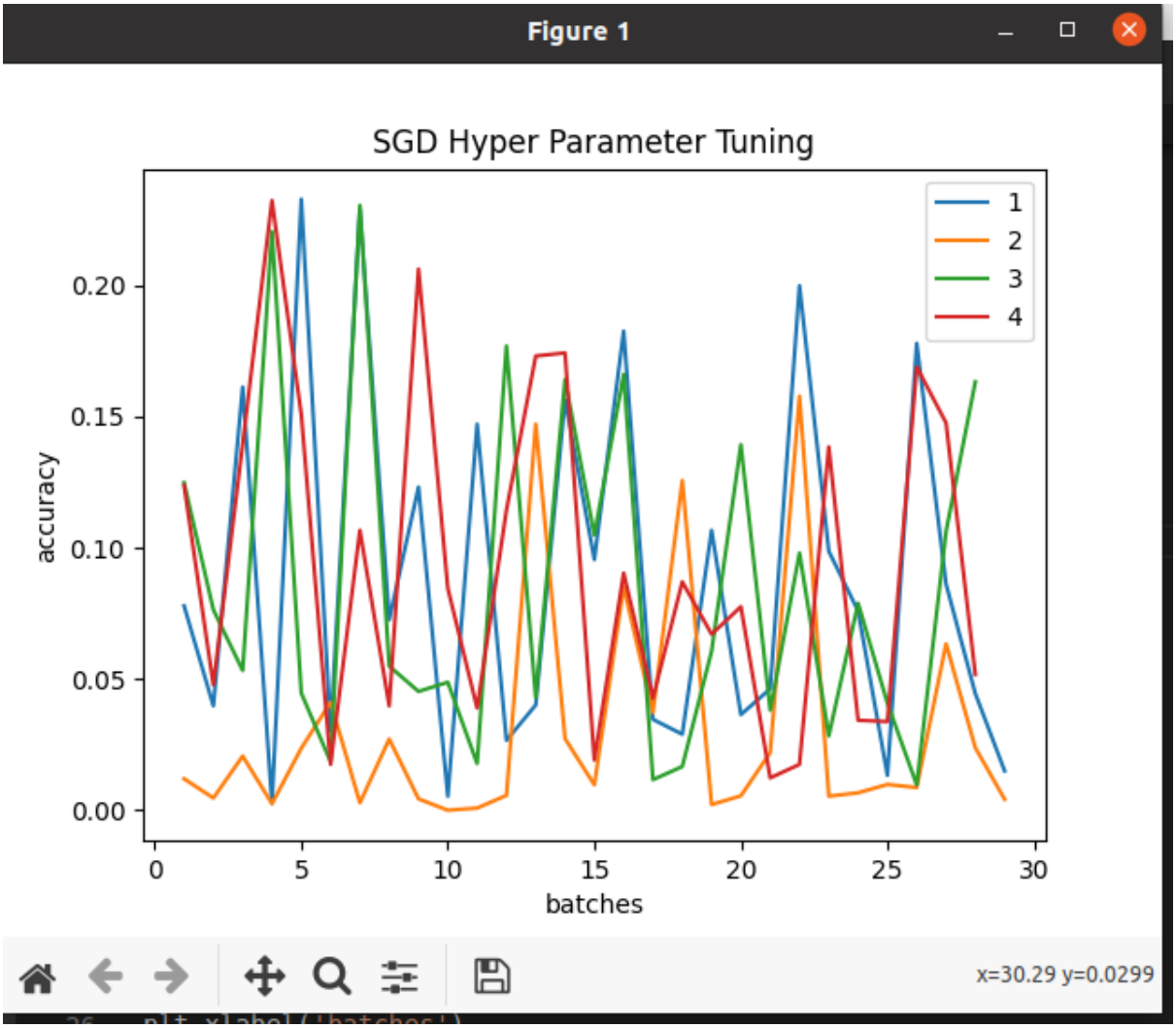


Figure 1
SGD Hyper Parameter Tuning
accuracy
batches
1
2
3
4
0.00
0.05
0.10
0.15
0.20
0
5
10
15
20
25
30
x=30.29 y=0.0299