

Report

Implementation:

First step is to get data from guardian using Kafka. I have created guardian topic and subscribing to that topic, I have collected real-time streaming and saved the data into CSV file using pandas.

Data Preprocessing:

I have loaded the data to dataframe. Using nltk packages, I have removed stop words which will reduce the accuracy. Using stemming, I have reduced words to their word stem.

Tokenization:

As the model can't read text, we need to send numerical data to train the model. For that, I have used TF-IDF vectorizer to reflect how important a word is to a document in a collection or corpus.

Model:

I have split the data as 70% train data and 30% validation data. I have trained the classifier with tokenized data and predicted the data with validation data. I have calculated accuracy, recall, f-score.

Streaming:

I have created batches with 5 seconds and predicting the output with the trained models and send them to elastic search using API

ElasticSearch:

Using spark Streaming, we collect data from Kafka and convert them to batches, we process the data and send them to the model to predict the output.

We send the output to elastic Search using API and visualize them on Kibana.

Classifiers used:

I have used different classifiers and finalized neural network.

SVM classifier

Accuracy: 0.35798

Neural network Classifiers

Accuracy: 0.8260869565217391

Multinomial Classifiers

Accuracy: 0.5289855072463768

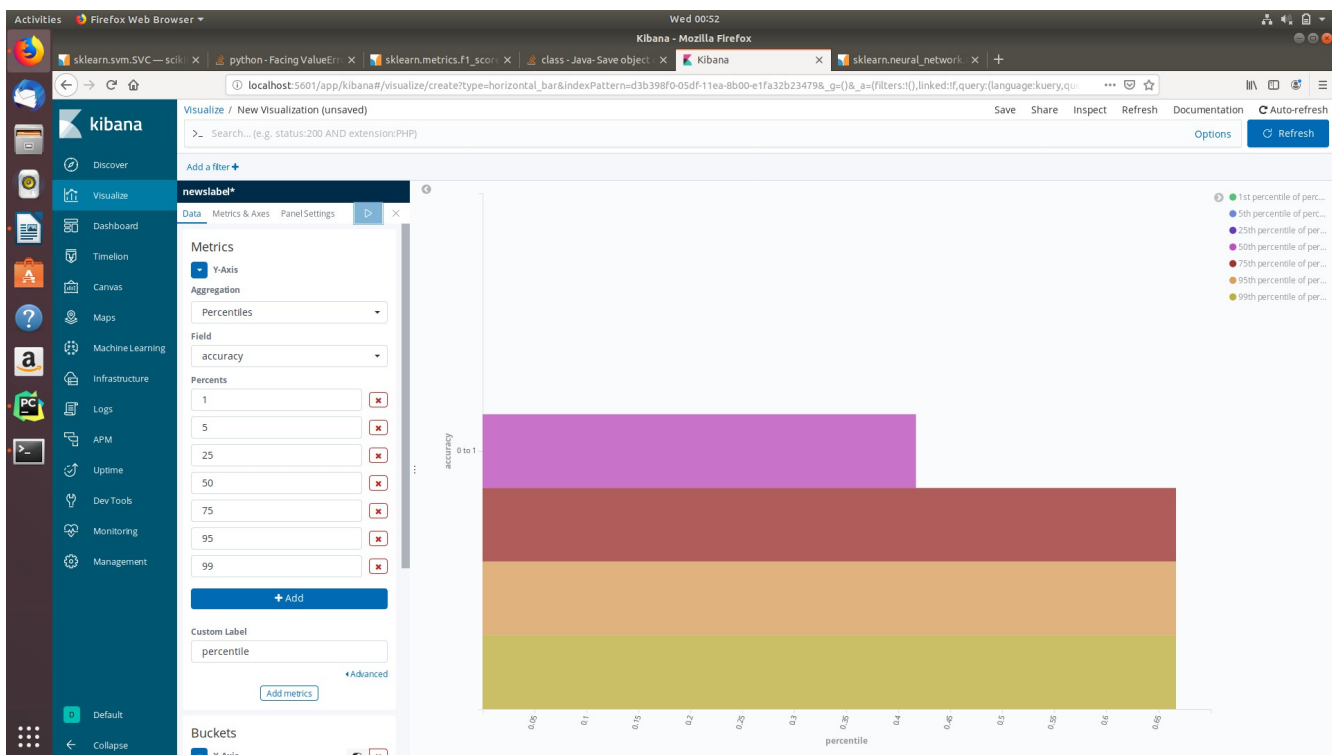
Logistic Regression Classifiers

Accuracy: 0.7318840579710145

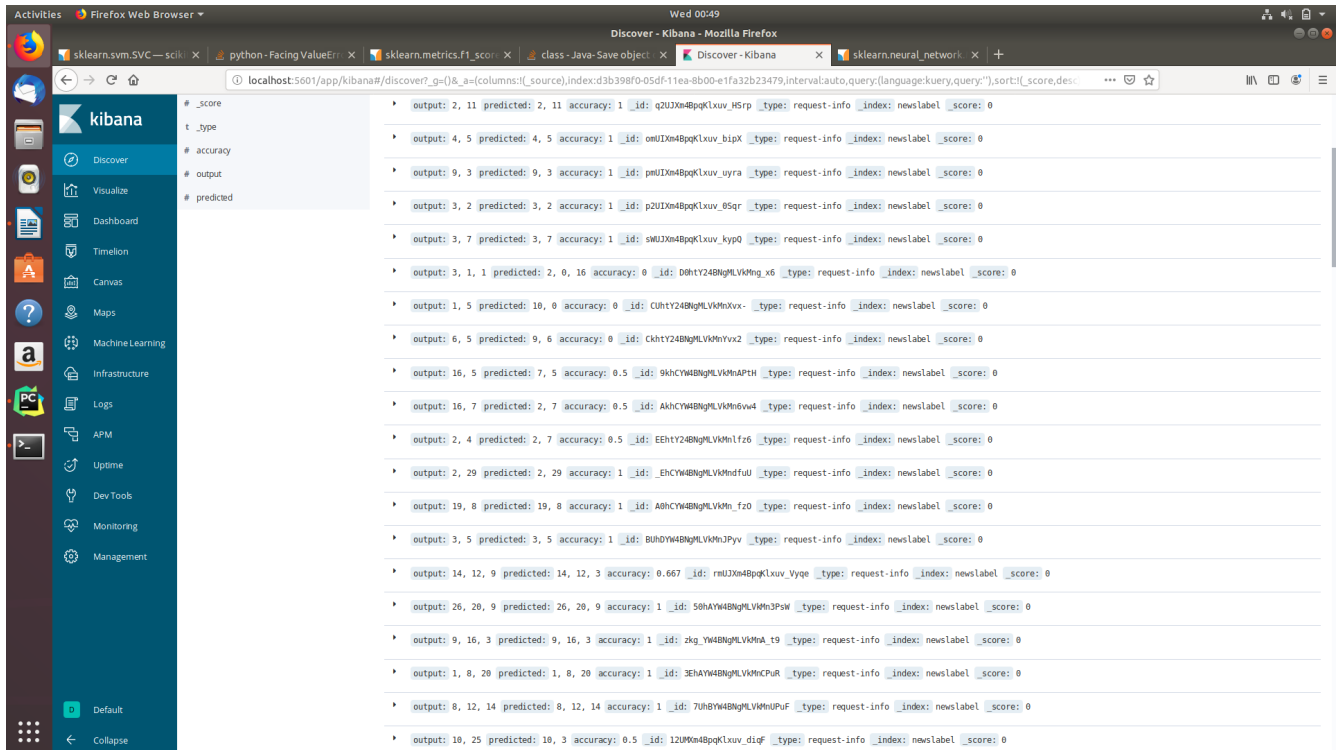
I have found few classifiers that are best fit for multiple classes and out of those, after training them multiple times and predicted the validation output, I found that neural net is the best fit with activation as logistic and max iterations with 200.

I created batches for 5 seconds and predicted it with the model that I have trained and send the data to elastic search by creating index **newslabel** and then I created pie chart with percentiles to see the accuracy of prediction.

Percentile of accuracy:



Data that has been sent to Kibana:



The screenshot shows a Kibana Discover interface in a Firefox browser. The left sidebar contains navigation options: Discover, Visualize, Dashboard, Timeline, Canvas, Maps, Machine Learning, Infrastructure, Logs, APM, Uptime, Dev Tools, Monitoring, and Management. The main area displays a table of data points. The table has columns for #, _score, _type, accuracy, output, and predicted. The data points are sorted by _score in descending order. Each row represents a data point with various fields including output, predicted, accuracy, _id, _type, request-info, _index, newLabel, and _score.

#	_score	_type	accuracy	output	predicted
1	0	request-info	0	output: 2, 11	predicted: 2, 11
2	0	request-info	0	output: 4, 5	predicted: 4, 5
3	0	request-info	0	output: 9, 3	predicted: 9, 3
4	0	request-info	0	output: 3, 2	predicted: 3, 2
5	0	request-info	0	output: 3, 7	predicted: 3, 7
6	0	request-info	0	output: 5, 1, 1	predicted: 2, 0, 16
7	0	request-info	0	output: 1, 5	predicted: 10, 0
8	0	request-info	0	output: 6, 5	predicted: 9, 6
9	0	request-info	0.5	output: 16, 5	predicted: 7, 5
10	0.5	request-info	0.5	output: 16, 7	predicted: 2, 7
11	0.5	request-info	0.5	output: 2, 4	predicted: 2, 7
12	1	request-info	1	output: 2, 29	predicted: 2, 29
13	1	request-info	1	output: 19, 8	predicted: 19, 8
14	1	request-info	1	output: 3, 5	predicted: 3, 5
15	0.667	request-info	0.667	output: 14, 12, 9	predicted: 14, 12, 3
16	1	request-info	1	output: 26, 20, 9	predicted: 26, 20, 9
17	1	request-info	1	output: 9, 16, 3	predicted: 9, 16, 3
18	1	request-info	1	output: 1, 8, 20	predicted: 1, 8, 20
19	1	request-info	1	output: 8, 12, 14	predicted: 8, 12, 14
20	0.5	request-info	0.5	output: 10, 25	predicted: 10, 3