ECE 9014: *Mitigation of Denial of Service*

Group #15 :

Kanika Gandhi

Shrutam Parmar

Karanpartap Singh Aulakh

Devashish Singh Rajput

Department of Electrical & Computer Engineering
The University of Western Ontario

Table of Content

1 Individual Deliverables -3: Predicative Analytics

1.1 Deliverables-3: Prediction of source IP based on destination port

1.1.1 Developer Info

Name: Kanika Gandhi

Student ID: 251323893

1.1.2 Predictive Data Analytics problem

In coordination to the deliberable-2 submitted, the details of the predictive data analytics are as given:

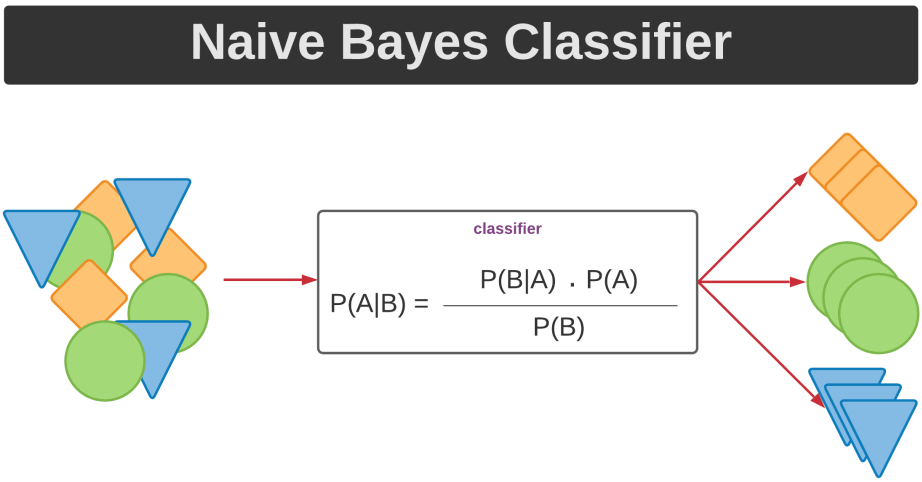
Application domain: Denial Of Service (Cyberattack) Mitigation

Predictive Analysis: *Forecast source IP for IP traceback*-To forecast a prediction using the relation between the packet emerging from source IP and the destination port at which the packet will hit the system, trying to penetrate it to carry out the DoS attack. If a packet hits the destination port that is the most vulnerable, it allows us to perform something known as 'IP Traceback'. 'IP traceback refers to when the reverse route is traced back to the malicious packet sender in order to realize the real attacker's location, to discard further packets arriving from that IP. This prediction can allow implementation of real-time mitigation techniques such as discarding of the similar packets if the packet is hitting a vulnerable port from the predicted IP, to avoid the cyber-attack.

1.1.3 Predictive Data Model

What is Naïve Bayes?

Naïve Bayes algorithm is a classification technique based on Bayes' Theorem with the assumption that features are independent of each other and are unrelated to each other. The algorithm uses prediction to forecast the highest probability of an object to classify data. This algorithm is easy and useful for large datasets. Even though the algorithm is easy, it is fast and may be used for multiclass predictions.



Based on our model:

We will be using the Source_ID and Destinationport_ID to predict the Source_ID of any packet hitting the vulnerable and open destinationport to identify which source IP packets to avoid in the transmission going further. Our model will constitute of the Source_ID and the Destinationport_ID and all attributes that are directly or indirectly dependent on them, which will be used to forecast the desired value.

1.1.4 Predictive Data Analytics Solution Pipeline

In order to perform the predictive analysis for the chosen problem, the process follows a specific workflow, as stated below:

1. Data Extraction: Post loading the data in DRUID UI for deliverable-2, the data was pulled in the form of a csv file which was then used to import data, into the notebook linked to the cluster created, in Databricks, by running the cell to interactively work with the data.

No.	Time	Source	Destination	Protocol	Length	Info	CHECKSUM	METHOD	Source_Port	Destination_Port
1	0	192.168.10.2	192.168.11.2	Modbus/TCP	78	Query: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	MD5	GET	54008	53
2	0.000245508	192.168.11.2	192.168.10.2	Modbus/TCP	81	Response: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	MD5	GET	54110	53
3	0.000626239	192.168.10.2	192.168.11.2	TCP	66	58032 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478	SHA-1	POST	54112	53
4	0.001063554	192.168.10.2	192.168.12.2	Modbus/TCP	78	Query: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	MD5	GET	54114	80
5	0.001314165	192.168.12.2	192.168.10.2	Modbus/TCP	81	Response: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	MD5	GET	54116	53
6	0.001662244	192.168.10.2	192.168.12.2	TCP	66	51000 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=1545815710 TSecr=2648963038	SHA-1	POST	54118	80
7	0.101907108	192.168.10.2	192.168.11.2	Modbus/TCP	78	Query: Trans: 47892; Unit: 1, Func: 3: Read Holding Registers	MD5	GET	54120	80

2. Data Cleaning: In order to get data ready for processing further, the command was run to drop the non-applicable values, split the data with "|" symbol and lastly change all data to lower case.

```
extract_categories = udf(lambda x: x.lower().split("|"), ArrayType(StringType()))
```

```
Command took 0.04 seconds -- by kgandh27@uwo.ca at 12/24/2022, 10:26:04 AM on My Cluster1
```

3. Feature and Label Selection: The libraries required for the process further were initially imported before moving forward in the process.

```
from pyspark.sql.functions import udf
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

Instrument ML code with MLflow: Use MLflow to track metrics, params, and models from your training co

```
Command took 0.31 seconds -- by kgandh27@uwo.ca at 12/24/2022, 10:25:55 AM on My Cluster1
```

```
df_1 = df.select(col("Time").cast('integer'),
                 "Source",
                 "Destination",
                 "Protocol",
                 "Length",
                 "Info",
                 "Source_Port",
                 "Destination_Port"
                 )
```

```
indexers = [StringIndexer (inputCol="Source",outputCol="Source_index"),
            StringIndexer (inputCol="Destination",outputCol="Destination_index"),
            StringIndexer (inputCol="Info",outputCol="Info_index"),
            StringIndexer (inputCol="Protocol",outputCol="Protocol_index"),
            StringIndexer (inputCol="Source_Port",outputCol="Source_Port_index"),
            StringIndexer (inputCol="Destination_Port",outputCol="Destination_Port_index")

            ]
```

Post importing the libraries, to develop relation between source IP and destination port, both are encoded using string indexers and vector assemblers and then to fit the machine learning model they are integrated to produce an output, namely "features".

```
vectorAssembler = VectorAssembler(inputCols = ["Destination_Port_index","Source_index"],outputCol = "features")
vindexed_df = vectorAssembler.transform(indexed_df)
vindexed_df.show(5,False)
```

(1) Spark Jobs

```
+----+-----------+-----------+----------+------+-----------------------------------------------------------------------------------------------+-----------+---
-----------+-----------+----------------+----------+--------------+-----------------+----------------------+---------+
|Time|Source     |Destination |Protocol  |Length|Info                                                                                           |Source_Port|Des
tination_Port|Source_index|Destination_index|Info_index|Protocol_index|Source_Port_index|Destination_Port_index|features |
+----+-----------+-----------+----------+------+-----------------------------------------------------------------------------------------------+-----------+---
-----------+-----------+----------------+----------+--------------+-----------------+----------------------+---------+
|0   |192.168.10.2|192.168.11.2|Modbus/TCP|78    |Query: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers              |54008      |53
|0.0         |1.0            |34.0      |0.0       |48.0          |1.0              |[1.0,0.0]|
|0   |192.168.11.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers           |54110      |53
|1.0         |0.0            |67.0      |0.0       |49.0          |1.0              |[1.0,1.0]|
|0   |192.168.10.2|192.168.11.2|TCP       |66    |58032  >  502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478|54112      |53
|0.0         |1.0            |19.0      |1.0       |50.0          |1.0              |[1.0,0.0]|
|0   |192.168.10.2|192.168.12.2|Modbus/TCP|78    |Query: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers              |54114      |80
|0.0         |2.0            |33.0      |0.0       |51.0          |0.0              |(2,[],[])|
|0   |192.168.12.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers           |54116      |53
|2.0         |0.0            |66.0      |0.0       |52.0          |1.0              |[1.0,2.0]|
+----+-----------+-----------+----------+------+-----------------------------------------------------------------------------------------------+-----------+---
-----------+-----------+----------------+----------+--------------+-----------------+----------------------+---------+
only showing top 5 rows
```

```
pipeline = Pipeline(stages=indexers)

indexed_df = pipeline.fit(df_1).transform(df_1)
indexed_df.show(5, False)

```

(13) Spark Jobs

```
+----+-----------+-----------+----------+------+-----------------------------------------------------------------------------------------------+-----------+---
-----------+-----------+----------------+----------+--------------+-----------------+----------------------+
|Time|Source     |Destination |Protocol  |Length|Info                                                                                           |Source_Port|Des
tination_Port|Source_index|Destination_index|Info_index|Protocol_index|Source_Port_index|Destination_Port_index|
+----+-----------+-----------+----------+------+-----------------------------------------------------------------------------------------------+-----------+---
-----------+-----------+----------------+----------+--------------+-----------------+----------------------+
|0   |192.168.10.2|192.168.11.2|Modbus/TCP|78    |Query: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers              |54008      |53
|0.0         |1.0            |34.0      |0.0       |48.0          |1.0              |
|0   |192.168.11.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers           |54110      |53
|1.0         |0.0            |67.0      |0.0       |49.0          |1.0              |
|0   |192.168.10.2|192.168.11.2|TCP       |66    |58032  >  502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478|54112      |53
|0.0         |1.0            |19.0      |1.0       |50.0          |1.0              |
|0   |192.168.10.2|192.168.12.2|Modbus/TCP|78    |Query: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers              |54114      |80
|0.0         |2.0            |33.0      |0.0       |51.0          |0.0              |
|0   |192.168.12.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers           |54116      |53
|2.0         |0.0            |66.0      |0.0       |52.0          |1.0              |
+----+-----------+-----------+----------+------+-----------------------------------------------------------------------------------------------+-----------+---
```

A pipeline was implemented in order to orchestrate the flow of input data through the ML model to produce output desirable data.

4. Model Training: In order to forecast it is import to first train model on a train data set and then check the prediction accuracy by predicting a test set. First, the test set was split into 70% training set and 30% testing set. The naïve bayes model was created and it's parameters were set. The created model was made to train on the dataset.

```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

```

```
nb = NaiveBayes(modelType="multinomial", labelCol = "Source_index")
nbmodel = nb.fit(train_df)
```

```
#split data testing and training (training 0,
splits = vindexed_df.randomSplit([0.7,0.3])
#optional value 34 is seed for sampling
train_df = splits[0]
test_df = splits[1]
```

5. Model Evaluation: First the testing/validation of prediction is carried out post which the accuracy of the model is defined and then the model accuracy is calculated.

```
evaluator = MulticlassClassificationEvaluator (labelCol="Source_index", predictionCol="prediction", metricName="accuracy")
nbaccuracy = evaluator.evaluate (predictions_df)
print("Test accuracy=" + str(nbaccuracy*100) , '%')
```

1.1.5 Samples of the Predictive Data Analytics Solutions

By implementing the naïve bayes model, the following predictions results were obtained:

```
predictions_df=nbmodel.transform(test_df)
predictions_df.show(5, True)
```

(1) Spark Jobs

```
+----+-------------+-------------+----------+------+--------------------+-----------+----------------+-----------+----------------+----------+----
----------+----------------+-----------------------+--------+--------------------+--------------------+--------------------+----------+
|Time|       Source| Destination|  Protocol|Length|                Info|Source_Port|Destination_Port|Source_index|Destination_index|Info_index|Prot
ocol_index|Source_Port_index|Destination_Port_index| features|       rawPrediction|         probability|prediction|
+----+-------------+-------------+----------+------+--------------------+-----------+----------------+-----------+----------------+----------+----
----------+----------------+-----------------------+--------+--------------------+--------------------+--------------------+----------+
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|      54120|              80|        0.0|             1.0|      36.0|
0.0|            54.0|                  0.0|(2,[],[])|[-0.4125322753312...|[0.66197183098591...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|      41656|              80|        0.0|             1.0|      44.0|
0.0|             0.0|                  0.0|(2,[],[])|[-0.4125322753312...|[0.66197183098591...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|      54202|             443|        0.0|             1.0|      46.0|
0.0|            82.0|                  2.0|[2.0,0.0]|[-0.4492305526676...|[0.96388747459901...|       0.0|
|   0|192.168.10.2|192.168.11.2|       TCP|    66|58032  >  502 [AC...|      54230|             443|        0.0|             1.0|      17.0|
1.0|            96.0|                  2.0|[2.0,0.0]|[-0.4492305526676...|[0.96388747459901...|       0.0|
|   0|192.168.10.2|192.168.11.2|       TCP|    66|58032  >  502 [AC...|      41702|           51640|        0.0|             1.0|      18.0|
1.0|             4.0|                  8.0|[8.0,0.0]|[-0.5593253846768...|[0.99997361093069...|       0.0|
+----+-------------+-------------+----------+------+--------------------+-----------+----------------+-----------+----------------+----------+----
----------+----------------+-----------------------+--------+--------------------+--------------------+--------------------+----------+
only showing top 5 rows
```

1.1.6 Predictive Data Analytics Solution: Accuracy & Performance

By adjusting the parameters of the split, to obtain the final result. In our data analytical problem there are five values of weather being used hence prediction is but difficult. That's why the final accuracy was 75%. While training the model, I found that the Naive Bayes model has the high efficiency, It performs better with small scale data, not very sensitive to the missing data and the algorithm is simple as well.

```
evaluator = MulticlassClassificationEvaluator (labelCol="Source_index", predictionCol="prediction", metricName="accuracy")
nbaccuracy = evaluator.evaluate (predictions_df)
print("Test accuracy=" + str(nbaccuracy*100) , '%')
```

(1) Spark Jobs

```
Test accuracy=75.75757575757575 %
```

1.2 Deliverables-3: Prediction of the destination IP using the source port

1.2.1 Developer Info

Name: Karanpartap Singh Aulakh

Student ID:251323916

1.2.2 Predictive Data Analytics problem

Details of the predictive data analytics, in accordance with the deliverable-2 are as follows:

Application domain: Denial of Service (Cyberattack) Mitigation

Predictive Analysis: I am using predictive analysis to forecast the nature of the destination IP when a packet is transmitted from a source port. This predictive analysis can be used for two things:

1. If the source port is a dangerous/malicious port, we can change the course of the packet to a new destination which could be a dummy or a wrong destination, hence dodging the attack.
2. To check whether the packets follow route, as in the information given on the packet or the destination IP on the packet (where it should reach) is exactly where the packet goes. This helps filtering out malicious packets.

Such predictions will be a value added to the mitigation techniques developed so far in the industry to prevent DOS attacks in the future.

1.2.3 Predictive Data Model

I have used Naïve Bayes Classifier Algorithm for my project. It is an algorithm which is mainly used for classification problems and belongs to the class of Supervised Learning Algorithms. It belongs in the domain of text classification and is an ideal fit for high-dimensional dataset. Naïve Bayes Classifier Algorithm is a probabilistic classifier which simply means that it predicts on object's probability.

The word 'naïve' in the algorithm is used to indicate the fact that the algorithm assumes that the occurrence of a certain feature is not dependent on occurrence of other features. The word 'Bayes' indicates that it is based on the principles of Bayes Theorem.

$$P(X \mid Y) = \frac{P(X \cap Y)}{P(Y)} \quad \Rightarrow \quad P(Y \mid X) = \frac{P(X \cap Y)}{P(X)}$$

P (Evidence | Outcome) — P (Outcome | Evidence)

$$P(Y \mid X) = \frac{P(X \mid Y) * P(Y)}{P(X)}$$

Here we can infer following:

- P(Y|X) is posterior probability that is probability of Y on event X.
- P(X|Y) is likelihood probability that is probability of evidence that the hypothesis's probability is true.
- P(Y) is prior probability that is before observation of evidence, the probability of hypothesis.
- P(X) is marginal probability is the probability of evidence P(X) is marginal probability is the probability of evidence.

Based on our model:

- I will be using the Destination IP and Sourceport to predict Destination IP to classify the packet trasnmitted as safe or unsafe and if it falls into the latter category, to try to make mitigation mechanisms to dodge/stop the transmission. For forecasting the desired values, I will be using Destination_ID and Sourceport_ID.

1.2.4 Predictive Data Analytics Solution Pipeline

I followed the following steps in order to perform predictive data analysis on my specific problem:

- Step 1: We had already loaded the data into DRUID UI for deliverable 2, so I pulled data in csv type file for my project. This csv file was then used to import data into cluster created in the notebook in Databricks and by running it to work with data. This step is known as Data Extraction.

```
# File location and type
file_location = "/FileStore/tables/Fact_table.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

No.	Time	Source	Destination	Protocol	Length	Info	CHECKSUM	METHOD	Source_Port	Destination_Port
1	0	192.168.10.2	192.168.11.2	Modbus/TCP	78	Query: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	MD5	GET	54008	53
2	0.000245508	192.168.11.2	192.168.10.2	Modbus/TCP	81	Response: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	MD5	GET	54110	53
3	0.000626239	192.168.10.2	192.168.11.2	TCP	66	58032 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478	SHA-1	POST	54112	53
4	0.001063554	192.168.10.2	192.168.12.2	Modbus/TCP	78	Query: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	MD5	GET	54114	80
5	0.001314165	192.168.12.2	192.168.10.2	Modbus/TCP	81	Response: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	MD5	GET	54116	53
6	0.001662244	192.168.10.2	192.168.12.2	TCP	66	51000 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=1545815710 TSecr=2648963038	SHA-1	POST	54118	80
7	0.101907108	192.168.10.2	192.168.11.2	Modbus/TCP	78	Query: Trans: 47892; Unit: 1, Func: 3: Read Holding Registers	MD5	GET	54120	80

- Step 2: The second step in the process is to do Data Cleaning. In order to proceed further, I dropped the non-applicable columns, split the data on “|” basis and changing the data to lower case.

```
extract_categories = udf(lambda x: x.lower().split("|"), ArrayType(StringType()))
```

- Step 3: The third step in the process is Feature and Label Selection. All the required libraries and modules were imported at the start of the implementation.

```
from pyspark.sql.functions import udf
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
df_1 = df.select(col("Time").cast('integer'),
                 "Source",
                 "Destination",
                 "Protocol",
                 "Length",
                 "Info",
                 "Source_Port"
                 )
```

After importing, String indexes and vector assemblers were used to encode the relationship between Source IP and Destination IP, to be able to fit the ML model to produce output in form of features.

```
indexers = [StringIndexer (inputCol="Source",outputCol="Source_index"),
            StringIndexer (inputCol="Destination",outputCol="Destination_index"),
            StringIndexer (inputCol="Info",outputCol="Info_index"),
            StringIndexer (inputCol="Protocol",outputCol="Protocol_index"),
            StringIndexer (inputCol="Source_Port",outputCol="Source_Port_index")

            ]
```

The implementation of the pipeline ML feature helped extract output data after regulating the flow of input data through the model that was developed.

```
pipeline = Pipeline(stages=indexers)

indexed_df = pipeline.fit(df_1).transform(df_1)
indexed_df.show(5, False)
```

Python

```
+----+------------+------------+----------+------+-----------------------------------------------------------------------------------------------------------+----------+----------+----
-------------+----------+--------------+-----------------+
|Time|Source      |Destination |Protocol  |Length|Info
|Source_Port|Source_index|Destination_index|Info_index|Protocol_index|Source_Port_index|
+----+------------+------------+----------+------+-----------------------------------------------------------------------------------------------------------+----------+----------+----
-------------+----------+--------------+-----------------+
|0   |192.168.10.2|192.168.11.2|Modbus/TCP|78    |Query: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers              |54008     |0.0       |1.0
|34.0      |0.0           |48.0            |
|0   |192.168.11.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers           |54110     |1.0       |0.0
|67.0      |0.0           |49.0            |
|0   |192.168.10.2|192.168.11.2|TCP       |66    |58032  >  502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478|54112     |0.0       |1.0
|19.0      |1.0           |50.0            |
|0   |192.168.10.2|192.168.12.2|Modbus/TCP|78    |Query: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers              |54114     |0.0       |2.0
|33.0      |0.0           |51.0            |
|0   |192.168.12.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers           |54116     |2.0       |0.0
|66.0      |0.0           |52.0            |
+----+------------+------------+----------+------+-----------------------------------------------------------------------------------------------------------+----------+----------+----
-------------+----------+--------------+-----------------+
only showing top 5 rows
```

```
vectorAssembler = VectorAssembler(inputCols = ["Source_Port_index", "Destination_index"],outputCol = "features")
vindexed_df = vectorAssembler.transform(indexed_df)
vindexed_df.show(5,False)
```

Python

```
+----+------------+------------+----------+------+-----------------------------------------------------------------------------------------------------------+----------+----------+----
-------------+----------+--------------+-----------------+----------+
|Time|Source      |Destination |Protocol  |Length|Info
|Source_Port|Source_index|Destination_index|Info_index|Protocol_index|Source_Port_index|features  |
+----+------------+------------+----------+------+-----------------------------------------------------------------------------------------------------------+----------+----------+----
-------------+----------+--------------+-----------------+----------+
|0   |192.168.10.2|192.168.11.2|Modbus/TCP|78    |Query: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers              |54008     |0.0       |1.0
|34.0      |0.0           |48.0            |[48.0,1.0]|
|0   |192.168.11.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers           |54110     |1.0       |0.0
|67.0      |0.0           |49.0            |[49.0,0.0]|
|0   |192.168.10.2|192.168.11.2|TCP       |66    |58032  >  502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478|54112     |0.0       |1.0
|19.0      |1.0           |50.0            |[50.0,1.0]|
|0   |192.168.10.2|192.168.12.2|Modbus/TCP|78    |Query: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers              |54114     |0.0       |2.0
|33.0      |0.0           |51.0            |[51.0,2.0]|
|0   |192.168.12.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers           |54116     |2.0       |0.0
|66.0      |0.0           |52.0            |[52.0,0.0]|
+----+------------+------------+----------+------+-----------------------------------------------------------------------------------------------------------+----------+----------+----
-------------+----------+--------------+-----------------+----------+
only showing top 5 rows
```

- Step 4: The fourth step is Training the Model. I have used training set and test set to train my model and then using test set to predict the values and accuracy of the model. The ratio of training set to test set is 80:20 respectively. Creation of naïve bayes was done and parameters were included in as well. The model thus created was trained on dataset.

```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
nb = NaiveBayes(modelType="multinomial", labelCol = "Source_index")
nbmodel = nb.fit(train_df)
```

```
#split data testing and training (training 0,
splits = vindexed_df.randomSplit([0.7,0.3])
#optional value 34 is seed for sampling
train_df = splits[0]
test_df = splits[1]
```

Python

- Step 5: The fifth and the last step in the process is the Model Evaluation. In this step, the accuracy of model is calculated by the predictions produced by the model post testing and validation.

```
evaluator = MulticlassClassificationEvaluator (labelCol="Destination_index", predictionCol="prediction", metricName="accuracy")
nbaccuracy = evaluator.evaluate (predictions_df)
print("Test accuracy=" + str(nbaccuracy*100) , '%')
```

1.2.5 Samples of the Predictive Data Analytics Solutions

After the data preprocessing stage, data cleaning, choosing feature and labels, model prediction values yield to the following predictions.

```
predictions_df=nbmodel.transform(test_df)
predictions_df.show(5, True)
```

Python

```
+----+-------------+-------------+----------+------+--------------------+-----------+-------------+-----------------+----------+--------------+-----------------+--------
--------+--------------------+--------------------+----------+
|Time|      Source| Destination|  Protocol|Length|                Info|Source_Port|Source_index|Destination_index|Info_index|Protocol_index|Source_Port_index|
features|       rawPrediction|         probability|prediction|
+----+-------------+-------------+----------+------+--------------------+-----------+-------------+-----------------+----------+--------------+-----------------+--
--------+--------------------+--------------------+----------+
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|      54132|          0.0|              1.0|      38.0|           0.0|             60.0|
[60.0,1.0]|[-8.2234815991160...|[0.08237800001336...|       1.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|      41656|          0.0|              1.0|      44.0|           0.0|              0.0|
[0.0,1.0]|[-8.1748789233113...|[0.01114934447490...|       2.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|      54202|          0.0|              1.0|      46.0|           0.0|             82.0|
[82.0,1.0]|[-8.2413025802444...|[0.13437402747120...|       1.0|
|   0|192.168.10.2|192.168.11.2|       TCP|    66|58032  >  502 [AC...|      54230|          0.0|              1.0|      17.0|           1.0|             96.0|
[96.0,1.0]|[-8.2526432045988...|[0.17706611891739...|       1.0|
|   0|192.168.10.2|192.168.11.2|       TCP|    66|58032  >  502 [AC...|      54206|          0.0|              1.0|      31.0|           1.0|             84.0|
[84.0,1.0]|[-8.2429226694379...|[0.14000857675908...|       1.0|
+----+-------------+-------------+----------+------+--------------------+-----------+-------------+-----------------+----------+--------------+-----------------+--
--------+--------------------+--------------------+----------+
only showing top 5 rows
```

1.2.6 Predictive Data Analytics Solution: Accuracy & Performance

From the accuracy achieved, we can infer two things:

1. Using naïve bayes and accuracy of 68.5 = 69% was achieved.
2. The model built was trained well as it did not train on the data completely, avoiding generalization error (100% accuracy on test set), hence avoiding the underfitting or overfitting issue.

```
Test accuracy=68.57142857142857 %
```

1.3 Deliverables-3: Prediction of method

1.3.1 Developer Info

Name: Devashish Singh Rajput

Student ID: 251321118

1.3.2 Predictive Data Analytics problem

Hypertext Transfer Protocol (HTTP) is used for communication between clients and servers. It works as a request-response protocol. It has different types of methods for communication from which GET and POST are the most common methods. As this is a communication protocol, hacker can easily use this as a malicious mechanism. So, by using these methods hacker can attack other user devices and either get the required information or can inject viruses.

Here, we are going to see which type of method is used. As a binary classifier, we are utilizing logistic regression for this. This is the forecast for categorizing the approaches according to the dataset's features.

1.3.3 Predictive Data Model

Dataset - The dataset contains different types of denial of service attacks.

Attributes -

No - Serial Numbers

Time - In seconds

Source - Source from where the attack is performed

Destination - Destination on which the attack is performed

Protocol - Set of rules for processing data

Length - Length of the packet

Info - Query/Responses

Checksum - Types of checksum (MD5/SHA-1)

Method - Type of request (GET/POST)

Source Port - The port from where the attack is performed

Destination Port - The port on which the attacked is performed

Model:

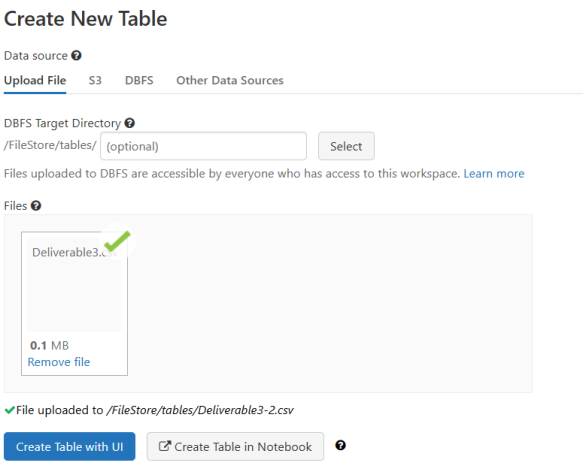
The analysis of data is to develop a classification model to classify the METHOD based info and other attributes of the dataset.

1.3.4 Predictive Data Analytics Solution Pipeline

A fact table was produced by the data warehouse. The predictive model uses Druid as its data source. Using SparkML models and the Data Brick community environment, the predictive model is created.

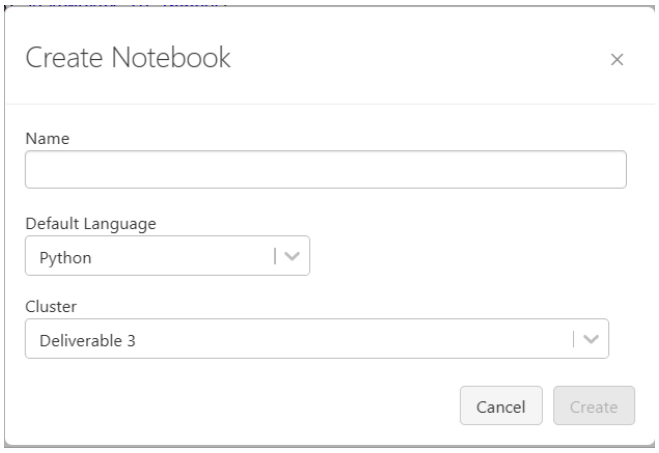
As a user, you must create an account in order to utilize the DataBrick community edition.

On successful signing in, first step is to load the data which is resulted from the Druid warehouse.



The next step is to build a cluster and run the notebook on it. The SparkML features are coded in notebook to create the prediction algorithm. To keep the notebook running, a cluster is made up of worker nodes, cluster management, and driver programs.

Once a cluster is built, the next part is to create a notebook in it.



To construct the prediction model, the necessary packages are imported. the fundamental components of the Python Spark package, such as Pipeline, VectorAssembler, and StringIndexer.

Cmd 1

```
1  from pyspark.sql.functions import udf
2  from pyspark.sql. functions import *
3  from pyspark.sql.types import *
4  from pyspark.ml import Pipeline
5  from pyspark.ml.feature import VectorAssembler
6  from pyspark.ml. feature import StringIndexer
```

After the necessary packages have been imported, the data is loaded in the workspace from the data brick's disc where it was first loaded and is imported in csv format. Using the read.format parameters from the spark package, the data is loaded.

```
file_location = "/FileStore/tables/raw_data.csv"

filetype = "csv"

infer_schema = "false"
first_row_is_header = "True"
delimiter = ","

df = spark.read.format(filetype)\
.option("infer_schema", infer_schema)\
.option("header", first_row_is_header)\
.option("sep", delimiter)\
.load(file_location)



display(df)
```

(2) Spark Jobs

Table +

	No.	Time	Source	Destination	Protocol	Length	Info	CHE
1	1	0	192.168.10.2	192.168.11.2	Modbus/TCP	78	Query: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	MD5
2	2	0.000245508	192.168.11.2	192.168.10.2	Modbus/TCP	81	Response: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	MD5
3	3	0.000626239	192.168.10.2	192.168.11.2	TCP	66	58032 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478	SHA-
4	4	0.001063554	192.168.10.2	192.168.12.2	Modbus/TCP	78	Query: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	MD5
5	5	0.001314165	192.168.12.2	192.168.10.2	Modbus/TCP	81	Response: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	MD5

To classify the Checksum, the other attributes on which the classification depends are Time, Source, Destination, Protocol, Length and Info. Those attributes are selected in the dataframe.

The few attributes like source, destination, protocol, info and checksum are in the string format whose indexing is done using the StringIndexer functionality.

```
df_1 = df.select(col("Time").cast('integer'),
                 "Source",
                 "Destination",
                 "Protocol",
                 "Length",
                 "Info",
                 "METHOD",
                 col("Source_Port").cast('integer'),
                 col("Destination_Port").cast('integer')
                )
```

```
mand took 0.09 seconds -- by sparma55@uwo.ca at 24/12/2022, 6:17:22 pm on My Cluster
```

```
L9
```

```
indexers = [StringIndexer (inputCol="Source",outputCol="Source_index"),
            StringIndexer (inputCol="Destination",outputCol="Destination_index"),
            StringIndexer (inputCol="Info",outputCol="Info_index"),
            StringIndexer (inputCol="Protocol",outputCol="Protocol_index"),
            StringIndexer (inputCol="METHOD",outputCol="METHOD_index")


           ]
```

The spark pipeline is defined with the staging as StringIndexer which is stored in the variable of indexer.

```
pipeline = Pipeline(stages=indexers)

indexed_df = pipeline.fit(df_1).transform(df_1)
indexed_df.show(5, False)

```

(11) Spark Jobs

```
+----+-------------+-------------+----------+------+-----------------------------------------------------------------------------------------------+------+-----------+----------------+------------
+-----------------+----------+-------------+------------+
|Time|Source       |Destination  |Protocol  |Length|Info                                                                                           |METHOD|Source_Port|Destination_Port|Source_index
|Destination_index|Info_index|Protocol_index|METHOD_index|
+----+-------------+-------------+----------+------+-----------------------------------------------------------------------------------------------+------+-----------+----------------+------------
+-----------------+----------+-------------+------------+
|0   |192.168.10.2|192.168.11.2|Modbus/TCP|78    |Query: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers              |GET   |54008      |53              |0.0
|1.0              |34.0      |0.0          |0.0         |
|0   |192.168.11.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers           |GET   |54110      |53              |1.0
|0.0              |67.0      |0.0          |0.0         |
|0   |192.168.10.2|192.168.11.2|TCP       |66    |58032  >  502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478|POST  |54112      |53              |0.0
|1.0              |19.0      |1.0          |1.0         |
|0   |192.168.10.2|192.168.12.2|Modbus/TCP|78    |Query: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers              |GET   |54114      |80              |0.0
|2.0              |33.0      |0.0          |0.0         |
|0   |192.168.12.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers           |GET   |54116      |53              |2.0
|0.0              |66.0      |0.0          |0.0         |
+----+-------------+-------------+----------+------+-----------------------------------------------------------------------------------------------+------+-----------+----------------+------------
+-----------------+----------+-------------+------------+
only showing top 5 rows
```

A feature transformer called VectorAssembler joins several vector columns into a single column. The feature set is assigned via the VectorAssembler.

```
vectorAssembler = VectorAssembler(inputCols = ["Source_Port","Destination_Port", "METHOD_index"],outputCol = "features")
vindexed_df = vectorAssembler.transform(indexed_df)
vindexed_df.show(5,False)
```

▸ (1) Spark Jobs

```
+----+-------------+-------------+-----------+------+----------------------------------------------------------------------------------------------------+------+-----------+----------------+------------
+----------------+----------+--------------+------------+------------------+
|Time|Source       |Destination  |Protocol   |Length|Info                                                                                                |METHOD|Source_Port|Destination_Port|Source_index
|Destination_index|Info_index|Protocol_index|METHOD_index|features          |
+----+-------------+-------------+-----------+------+----------------------------------------------------------------------------------------------------+------+-----------+----------------+------------
+----------------+----------+--------------+------------+------------------+
|0   |192.168.10.2|192.168.11.2|Modbus/TCP|78    |Query: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers                                   |GET   |54008      |53              |0.0
|1.0             |34.0      |0.0           |0.0         |[54008.0,53.0,0.0]|
|0   |192.168.11.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers                                |GET   |54110      |53              |1.0
|0.0             |67.0      |0.0           |0.0         |[54110.0,53.0,0.0]|
|0   |192.168.10.2|192.168.11.2|TCP       |66    |58032  >  502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478|POST  |54112      |53              |0.0
|1.0             |19.0      |1.0           |1.0         |[54112.0,53.0,1.0]|
|0   |192.168.10.2|192.168.12.2|Modbus/TCP|78    |Query: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers                                   |GET   |54114      |80              |0.0
|2.0             |33.0      |0.0           |0.0         |[54114.0,80.0,0.0]|
|0   |192.168.12.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers                                |GET   |54116      |53              |2.0
|0.0             |66.0      |0.0           |0.0         |[54116.0,53.0,0.0]|
+----+-------------+-------------+-----------+------+----------------------------------------------------------------------------------------------------+------+-----------+----------------+------------
+----------------+----------+--------------+------------+------------------+
only showing top 5 rows
```

The data is then split into train and test set with the split ratio 7:3.

```

#split data testing and training (training 0,
splits = vindexed_df.randomSplit([0.7,0.3])
#optional value 34 is seed for sampling
train_df = splits[0]
test_df = splits[1]
```

A logistic regression model is created to forecast checksums, and it is imported from PySpark's ML package and given the label METHOD_index along with a feature set. The train dataset is then used to fit the model.

```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(featuresCol="features", labelCol="METHOD_index", regParam=1.0)
```

Command took 0.04 seconds -- by sparma55@uwo.ca at 24/12/2022, 6:17:57 pm on My Cluster

Cmd 24

```
from pyspark.ml import Pipeline

# Define the pipeline based on the stages created in previous steps.
pipeline = Pipeline(stages=indexers)

# Define the pipeline model.
lrmodel = lr.fit(train_df)
```

▸ (12) Spark Jobs

The model is then used to predict on the test dataset.

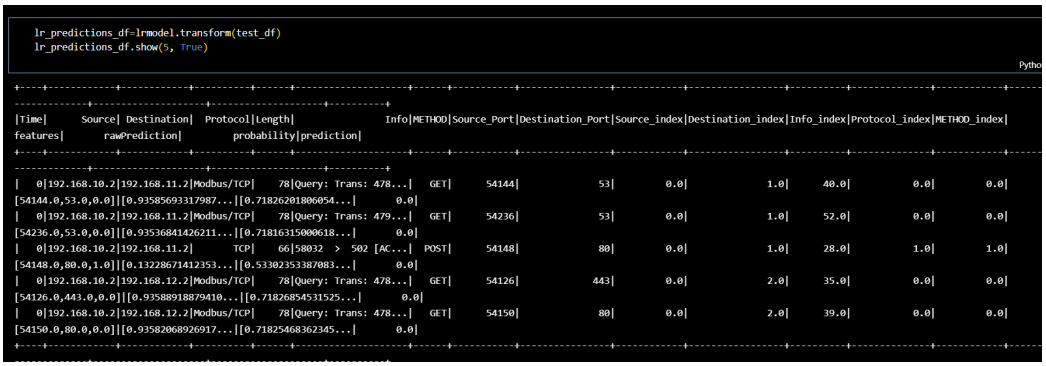
```
lr_predictions_df=lrmodel.transform(test_df)
lr_predictions_df.show(5, True)
```

▸ (1) Spark Jobs

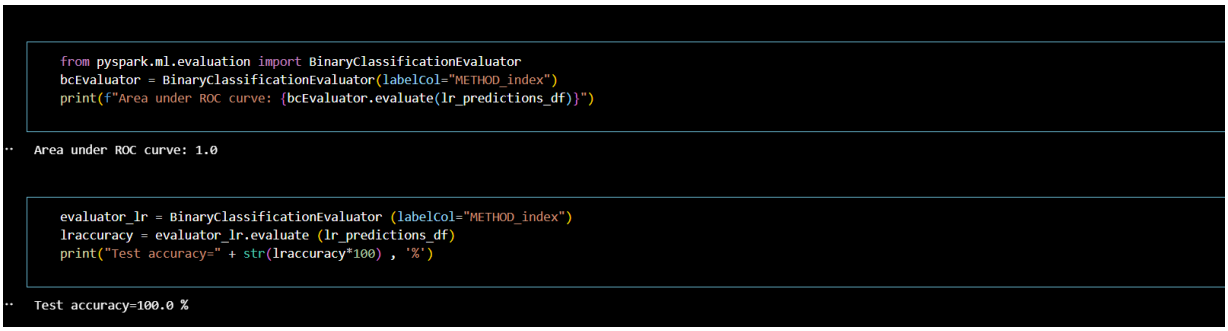
```
+----+----------+----------+--------+------+--------------------+------+-----------+----------------+----------+-----------------+----------+--------------+-----------+--------------------+--------------------+----------+
|Time|    Source|Destination|Protocol|Length|                Info|METHOD|Source_Port|Destination_Port|Source_index|Destination_index|Info_index|Protocol_index|METHOD_index|            features|       rawPrediction|probability|prediction|
+----+----------+----------+--------+------+--------------------+------+-----------+----------------+----------+-----------------+----------+--------------+-----------+--------------------+--------------------+----------+
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|   GET|      54144|              53|       0.0|              1.0|      40.0|           0.0|        0.0|[54144.0,53.0,0.0]|[0.93585693317987...|[0.718262...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 479...|   GET|      54236|              53|       0.0|              1.0|      52.0|           0.0|        0.0|[54236.0,53.0,0.0]|[0.93536841426211...|[0.718163...|       0.0|
|   0|192.168.10.2|192.168.11.2|       TCP|    66|58032 > 502 [AC...|  POST|      54148|              80|       0.0|              1.0|      28.0|           1.0|        1.0|[54148.0,80.0,1.0]|[0.13228671412353...|[0.533023...|       0.0|
|   0|192.168.10.2|192.168.12.2|Modbus/TCP|    78|Query: Trans: 478...|   GET|      54126|             443|       0.0|              2.0|      35.0|           0.0|        0.0|[54126.0,443.0,0.0]|[0.93588918879410...|[0.718268...|       0.0|
|   0|192.168.10.2|192.168.12.2|Modbus/TCP|    78|Query: Trans: 478...|   GET|      54150|              80|       0.0|              2.0|      39.0|           0.0|        0.0|[54150.0,80.0,0.0]|[0.93582068926917...|[0.718254...|       0.0|
+----+----------+----------+--------+------+--------------------+------+-----------+----------------+----------+-----------------+----------+--------------+-----------+--------------------+--------------------+----------+
only showing top 5 rows
```

1.3.5 Samples of the Predictive Data Analytics Solutions

The prediction algorithm yielded the following results:



1.3.6 Predictive Data Analytics Solution: Accuracy & Performance



To conclude it is that the final test accuracy of the logistic regression is shown below. Post analyzing the data prediction findings yielded a result of 100%. This is a really positive outcome, which might indicate overfitting because of the zero error ratio in train vs test set but the results can be improved.

All in all, logistic regression is a good tool which can be used to predict the packet GET or POST info accurately.

1.4 Deliverables-3: Type of Checksums

1.4.1 Developer Info

Name: Shrutam Parmar

Student ID: 251312991

1.4.2 Predictive Data Analytics problem

Checksum is a sequence of letters and numbers used to check the integrity of data. With the help of checksum, we can confirm that our data is identical. In the DOS attack, hacker tries to manipulate the data. If the data is manipulated then the checksum is also changed. So, by calculating checksum we can identify that if the data modification is done or not. We are having two types of checksums, MD5 and SHA-1.

Here, we are going to predict which type of checksum is used in the system. For this we are using Logistic Regression as a Binary Classifier. This is the prediction to classify the checksums based on the features of the dataset.

1.4.3 Predictive Data Model

Dataset:- The dataset contains different types of denial of service attacks.

Attributes:-

- No :- Serial Numbers
- Time :- In seconds
- Source :- Source from where the attack is performed
- Destination :- Destination on which the attack is performed
- Protocol :- Set of rules for processing data
- Length :- Length of the packet
- Info :- Query/Responses
- Checksum :- Types of checksum (MD5/SHA-1)
- Source Port :- The port from where the attack is performed
- Destination Port :- The port on which the attacked is performed

Model:

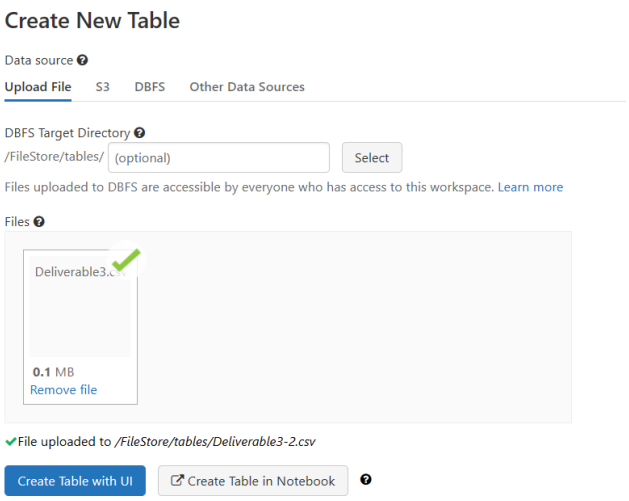
The analysis of data is to develop a classification model to classify the checksums-based info and other attributes of the dataset.

1.4.4 Predictive Data Analytics Solution Pipeline

The fact table resulted from the data warehouse Druid is the data source for the predictive model. The predictive model is developed on the Data Brick community environment and using the SparkML models.

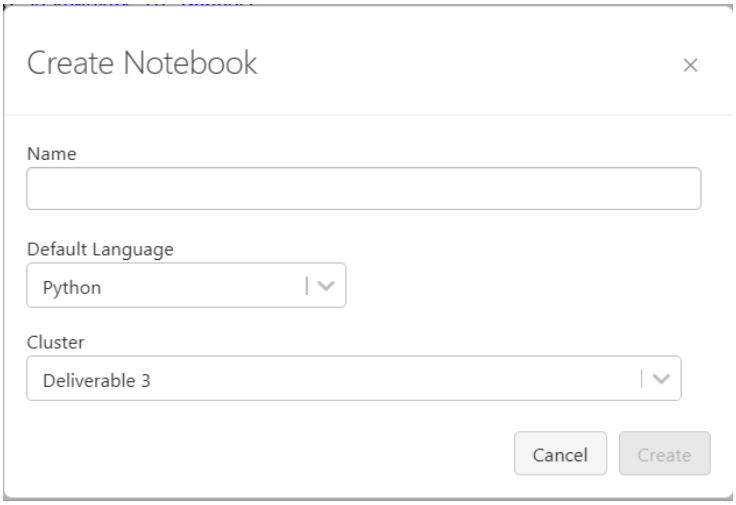
To use the DataBrick community edition, as a user the account is created.

On successful signing in, first step is to load the data which is resulted from the Druid warehouse.



The next step is to create a cluster to run the notebook in it. In notebook, the SparkML functionalities are coded to build the predictive algorithm. A cluster is a combination of Driver Programs, Cluster manager and worker nodes running together to keep the notebook running.

Once a cluster is built, the next part is to create a notebook in it.



The relevant packages are imported required to build the predictive model. The basic packages like Pipeline, VectorAssembler and StringIndexer from the pyspark package.

Cmd 1

```
1  from pyspark.sql.functions import udf
2  from pyspark.sql. functions import *
3  from pyspark.sql.types import *
4  from pyspark.ml import Pipeline
5  from pyspark.ml.feature import VectorAssembler
6  from pyspark.ml. feature import StringIndexer
```

Once the required packages are imported, the data is loaded in the workspace from the filesystem of the data brick where it was loaded initially and it is imported in the csv format. To load the data, the read.format options are used from the spark package.

```
file_location = "/FileStore/tables/raw_data.csv"

filetype = "csv"

infer_schema = "false"
first_row_is_header = "True"
delimiter = ","

df = spark.read.format(filetype)\
.option("infer_schema", infer_schema)\
.option("header", first_row_is_header)\
.option("sep", delimiter)\
.load(file_location)



display(df)
```

(2) Spark Jobs

Table +

	No.	Time	Source	Destination	Protocol	Length	Info	CHE
1	1	0	192.168.10.2	192.168.11.2	Modbus/TCP	78	Query: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	MD5
2	2	0.000245508	192.168.11.2	192.168.10.2	Modbus/TCP	81	Response: Trans: 47891; Unit: 1, Func: 3: Read Holding Registers	MD5
3	3	0.000626239	192.168.10.2	192.168.11.2	TCP	66	58032 > 502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478	SHA-
4	4	0.001063554	192.168.10.2	192.168.12.2	Modbus/TCP	78	Query: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	MD5
5	5	0.001314165	192.168.12.2	192.168.10.2	Modbus/TCP	81	Response: Trans: 47890; Unit: 3, Func: 3: Read Holding Registers	MD5

To classify the Checksum, the other attributes on which the classification depends are Time, Source, Destination, Protocol, Length and Info. Those attributes are selected in the dataframe.

The few attributes like source, destination, protocol, info and checksum are in the string format whose indexing is done using the StringIndexer functionality.

Cmd 3

```
df_1 = df.select(col("Time").cast('integer'),
                 "Source",
                 "Destination",
                 "Protocol",
                 "Length",
                 "Info",
                 "CHECKSUM")
```

Command took 0.07 seconds -- by sparma55@uwo.ca at 24/12/2022, 5:03:00 pm on My Cluster

Cmd 4

```
indexers = [StringIndexer (inputCol="Source",outputCol="Source_index"),
            StringIndexer (inputCol="Destination",outputCol="Destination_index"),
            StringIndexer (inputCol="Info",outputCol="Info_index"),
            StringIndexer (inputCol="Protocol",outputCol="Protocol_index"),
            StringIndexer (inputCol="CHECKSUM",outputCol="CHECKSUM_index")
            ]
```

Command took 0.10 seconds -- by sparma55@uwo.ca at 24/12/2022, 5:04:03 pm on My Cluster

Cmd 5

The spark pipeline is defined with the staging as StringIndexer which is stored in the variable of indexer.

Python

```
pipeline = Pipeline(stages=indexers)

indexed_df = pipeline.fit(df_1).transform(df_1)
indexed_df.show(5, False)

```

(11) Spark Jobs

```
+----+------------+------------+----------+------+--------------------------------------------------------------------------------------------------------+--------+------------+-----------------+----------
--+--------------+--------------+
|Time|Source      |Destination |Protocol  |Length|Info                                                                                                    |CHECKSUM|Source_index|Destination_index|Info_ind
ex|Protocol_index|CHECKSUM_index|
+----+------------+------------+----------+------+--------------------------------------------------------------------------------------------------------+--------+------------+-----------------+----------
--+--------------+--------------+
|0   |192.168.10.2|192.168.11.2|Modbus/TCP|78    |Query: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers                                       |MD5     |0.0         |1.0              |34.0
|0.0           |0.0           |
|0   |192.168.11.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers                                    |MD5     |1.0         |0.0              |67.0
|0.0           |0.0           |
|0   |192.168.10.2|192.168.11.2|TCP       |66    |58032  >  502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478|SHA-1   |0.0         |1.0              |19.0
|1.0           |1.0           |
|0   |192.168.10.2|192.168.12.2|Modbus/TCP|78    |Query: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers                                       |MD5     |0.0         |2.0              |33.0
|0.0           |0.0           |
|0   |192.168.12.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers                                    |MD5     |2.0         |0.0              |66.0
|0.0           |0.0           |
+----+------------+------------+----------+------+--------------------------------------------------------------------------------------------------------+--------+------------+-----------------+----------
--+--------------+--------------+
```

VectorAssembler is a feature transformer that combines multiple columns in a single vector column. Using the VectorAssembler, the feature set is assigned.

```
vectorAssembler = VectorAssembler(inputCols = ["Protocol_index", "CHECKSUM_index"],outputCol = "features")
vindexed_df = vectorAssembler.transform(indexed_df)
vindexed_df.show(5,False)
```

(1) Spark Jobs

```
+----+------------+------------+----------+------+--------------------------------------------------------------------------------------------------+--------+------------+-----------------+--------+
--+--------------+--------------+----------+
|Time|Source      |Destination |Protocol  |Length|Info                                                                                              |CHECKSUM|Source_index|Destination_index|Info_ind
ex|Protocol_index|CHECKSUM_index|features  |
+----+------------+------------+----------+------+--------------------------------------------------------------------------------------------------+--------+------------+-----------------+--------+
--+--------------+--------------+----------+
|0   |192.168.10.2|192.168.11.2|Modbus/TCP|78    |Query: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers                                 |MD5     |0.0         |1.0              |34.0
|0.0           |0.0           |(2,[],[])|
|0   |192.168.11.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47891; Unit:   1, Func:   3: Read Holding Registers                              |MD5     |1.0         |0.0              |67.0
|0.0           |0.0           |(2,[],[])|
|0   |192.168.10.2|192.168.11.2|TCP       |66    |58032  >  502 [ACK] Seq=13 Ack=16 Win=229 Len=0 TSval=340602092 TSecr=1065398478|SHA-1   |0.0         |1.0              |19.0
|1.0           |1.0           |[1.0,1.0]|
|0   |192.168.10.2|192.168.12.2|Modbus/TCP|78    |Query: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers                                 |MD5     |0.0         |2.0              |33.0
|0.0           |0.0           |(2,[],[])|
|0   |192.168.12.2|192.168.10.2|Modbus/TCP|81    |Response: Trans: 47890; Unit:   3, Func:   3: Read Holding Registers                              |MD5     |2.0         |0.0              |66.0
|0.0           |0.0           |(2,[],[])|
+----+------------+------------+----------+------+--------------------------------------------------------------------------------------------------+--------+------------+-----------------+--------+
--+--------------+--------------+----------+
only showing top 5 rows
```

The data is then split into train and test set with the split ratio 7:3.

Cmd 8

```
#split data testing and training (training 0,
splits = vindexed_df.randomSplit([0.7,0.3])
#optional value 34 is seed for sampling
train_df = splits[0]
test_df = splits[1]
```

(2) Spark Jobs

A logistic Regression model is defined to predict the checksums where the model is imported from the pyspark's ML package and assigned with feature set along with the label as checksum_index. The model is then fitted on the train dataset.

Cmd 9

```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(featuresCol="features", labelCol="CHECKSUM_index", regParam=1.0)
```

Command took 0.06 seconds -- by sparma55@uwo.ca at 24/12/2022, 5:21:00 pm on My Cluster

Cmd 10

```
from pyspark.ml import Pipeline

# Define the pipeline based on the stages created in previous steps.
pipeline = Pipeline(stages=indexers)

# Define the pipeline model.
lrmodel = lr.fit(train_df)
```

(8) Spark Jobs

Command took 3.68 seconds -- by sparma55@uwo.ca at 24/12/2022, 5:25:41 pm on My Cluster

The model is used to predict on the test dataset.

```
lr_predictions_df=lrmodel.transform(test_df)
lr_predictions_df.show(5, True)
```

(1) Spark Jobs

```
+----+------------+------------+----------+------+--------------------+--------+------------+-----------------+----------+--------------+----------+--------+-----------------
---+--------------------+----------+
|Time|      Source| Destination|  Protocol|Length|                Info|CHECKSUM|Source_index|Destination_index|Info_index|Protocol_index|CHECKSUM_index|features|       rawPredict
ion|         probability|prediction|
+----+------------+------------+----------+------+--------------------+--------+------------+-----------------+----------+--------------+----------+--------+-----------------
---+--------------------+----------+
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|     MD5|         0.0|              1.0|      38.0|           0.0|          0.0|(2,[],[])|[1.1546270937018
0...|[0.76035505937416...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|     MD5|         0.0|              1.0|      40.0|           0.0|          0.0|(2,[],[])|[1.1546270937018
0...|[0.76035505937416...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|     MD5|         0.0|              1.0|      42.0|           0.0|          0.0|(2,[],[])|[1.1546270937018
0...|[0.76035505937416...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|     MD5|         0.0|              1.0|      44.0|           0.0|          0.0|(2,[],[])|[1.1546270937018
0...|[0.76035505937416...|       0.0|
|   0|192.168.10.2|192.168.11.2|       TCP|    66|58032  >  502 [AC...|   SHA-1|         0.0|              1.0|      17.0|           1.0|          1.0|[1.0,1.0]|[-0.203360901786
4...|[0.44933426388462...|       1.0|
+----+------------+------------+----------+------+--------------------+--------+------------+-----------------+----------+--------------+----------+--------+-----------------
---+--------------------+----------+
```

1.4.5 Samples of the Predictive Data Analytics Solutions

Post completion of validation via prediction, the predicted values received were as shown below:

```
lr_predictions_df=lrmodel.transform(test_df)
lr_predictions_df.show(5, True)
```

```
+----+----------+----------+---------+------+--------------------+--------+----------+----------------+---------+-------------+-------------+--------+--
------------+--------------------+----------+
|Time|    Source| Destination|  Protocol|Length|                Info|CHECKSUM|Source_index|Destination_index|Info_index|Protocol_index|CHECKSUM_index|  features|
rawPrediction|         probability|prediction|
+----+----------+----------+---------+------+--------------------+--------+----------+----------------+---------+-------------+-------------+--------+--
------------+--------------------+----------+
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|     MD5|       0.0|             1.0|     38.0|          0.0|          0.0|(2,[],[])|
[1.15462709370180...|[0.76035505937416...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|     MD5|       0.0|             1.0|     40.0|          0.0|          0.0|(2,[],[])|
[1.15462709370180...|[0.76035505937416...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|     MD5|       0.0|             1.0|     42.0|          0.0|          0.0|(2,[],[])|
[1.15462709370180...|[0.76035505937416...|       0.0|
|   0|192.168.10.2|192.168.11.2|Modbus/TCP|    78|Query: Trans: 478...|     MD5|       0.0|             1.0|     44.0|          0.0|          0.0|(2,[],[])|
[1.15462709370180...|[0.76035505937416...|       0.0|
|   0|192.168.10.2|192.168.11.2|      TCP|    66|58032  >  502 [AC...|   SHA-1|       0.0|             1.0|     17.0|          1.0|          1.0|[1.0,1.0]|
[-0.2033609017864...|[0.44933426388462...|       1.0|
+----+----------+----------+---------+------+--------------------+--------+----------+----------------+---------+-------------+-------------+--------+--
------------+--------------------+----------+
only showing top 5 rows
```

1.4.6 Predictive Data Analytics Solution: Accuracy & Performance2

```
evaluator_lr = BinaryClassificationEvaluator (labelCol="CHECKSUM_index")
lraccuracy = evaluator_lr.evaluate (lr_predictions_df)
print("Test accuracy=" + str(lraccuracy*100) , '%')
```

```
Exception ignored in: <function JavaWrapper.__del__ at 0x7f2a4b29a8b0>
Traceback (most recent call last):
  File "/databricks/spark/python/pyspark/ml/wrapper.py", line 39, in __del__
    if SparkContext._active_spark_context and self._java_obj is not None:
AttributeError: 'BinaryClassificationEvaluator' object has no attribute '_java_obj'
Test accuracy=95.65217391304348 %
```

Inference: Below is a display of the logistic regression's final test accuracy. A result of 95.6% was obtained after evaluating the data prediction findings. This is a highly favorable result, and the outcomes seem to be enhanced. The generalization error might be curbing to 0 as the test set performs well post training.