BDAT 1008 Data Collection & Curation
Network Intrusion Detection System
Final Group Project
Team 2

Shubham Chawla, Archit Sinha,
Chintan Vajani

Content

- Project Topic
- Project Summary
- About our Team
- Dataset Information
- Project Scope – Data Flow
- Tools Used
- Pyspark Code Implementation – Jupyter Notebook
- Website development– Flask
- Website Deployment – Heroku
- Fit Gap Analysis

Project Topic

We are running a consulting company specializing in Big Data. Our potential client varies from small startups to Fortune Global 500 to angel investors and venture capitals.
Our team is going to present in a global tech conference, and we will be showcasing our project – **Network Intrusion Detection System built using Apache Spark and Flask.**

Project Summary

- Performed network traffic analysis and classification of network attack using Apache spark and MLlib
- Target Dataset – KDD99
- Used different Supervised Learning Models to classify network attacks – Logistic Regression, Decision Tree, Random Forest Classification and Multinomial Naïve Bayes
- Evaluated performance both in tabular and graphical way using four performance metrics – Accuracy, Weighted Precision, Weighted Recall and F1 score
- Developed a web application using Flask Framework and deployed it on Heroku
- Conducted a Fit-Gap Analysis

Team Members

Shubham Chawla – Developer

Bachelors in Computer Science, with experience in data mining, machine learning, analysis and visualization.

Archit Sinha - Business Analyst

Bachelors in Statistics with experience in data analysis and project management.

Chintan Vajani - Project Manager

Bachelors in Computer Science, with experience as associate system engineer.

Data Source – KD99

- Developed by University of California, Irvine 1998-99
- It consists of a single CSV file containing different types of network attacks (Neptune, smurf, DoS etc) along with normal samples
- Dataset size: 4,898,431 rows and 42 features
- The derived features are divided into two main categories:
  - Content based features
  - Time based features

Project Scope – Data Flow

Fetching structured data from csv file (KD99)

Preprocessing, cleaning & feature engineering

Data cleaning using apache spark and jupyter

Performing EDA

Cleaned Data

Modelling and analytical techniques LR, DT, RF & NB

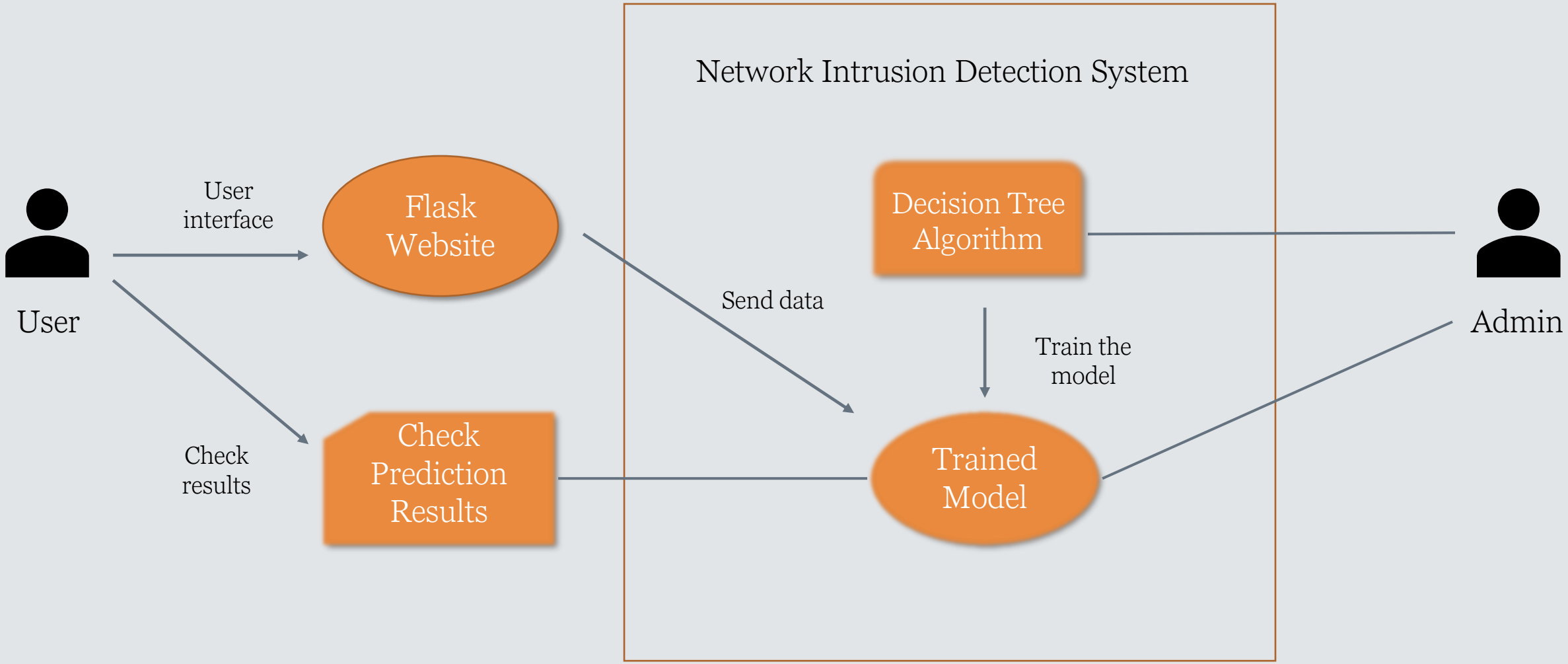
Evaluation

Evaluation Metrics – Accuracy, precision, recall and F1 score

Deployment

Deployment of model on Flask + Heroku

Use Case Diagram



Tools Used

- Apache Spark
- Mllib
- Flask
- Heroku
- Visual Studio Code

Initializing Apache Spark and loading modules

Jupyter Notebook link - https://github.com/shubhamchawla02/BDAT-1008-Final-Project

In [1]:

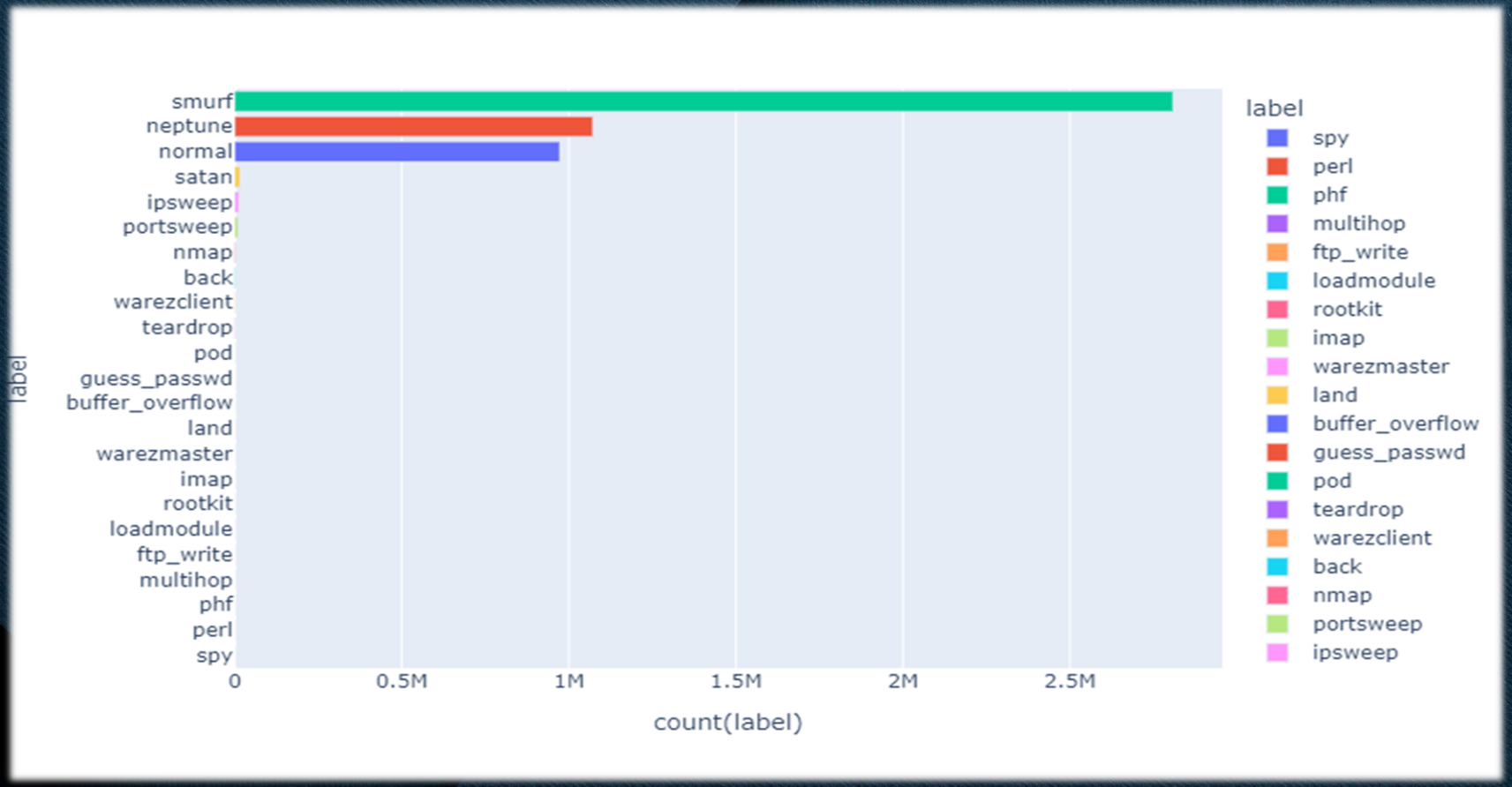
```
import pyspark
from pyspark.context import SparkContext
import warnings
warnings.filterwarnings("ignore")
import time
from pyspark.sql import SparkSession

# To perform feature engineering
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml import Pipeline

# To classify label class
from pyspark.ml.classification import LogisticRegression, DecisionTreeClassifier, \
                                      NaiveBayes, RandomForestClassifier

# Importing MulticlassClassificationEvaluator to evaluate the performance of the classifiers (models),
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

from pyspark.sql.functions import regexp_replace,lit
# Spark Session
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
spark.conf.set('spark.sql.shuffle.partitions', 6)
spark.conf.set('num-executors', 16)
spark
```

Data Exploration – label class distribution

Using Pandas API on Spark to explore data further

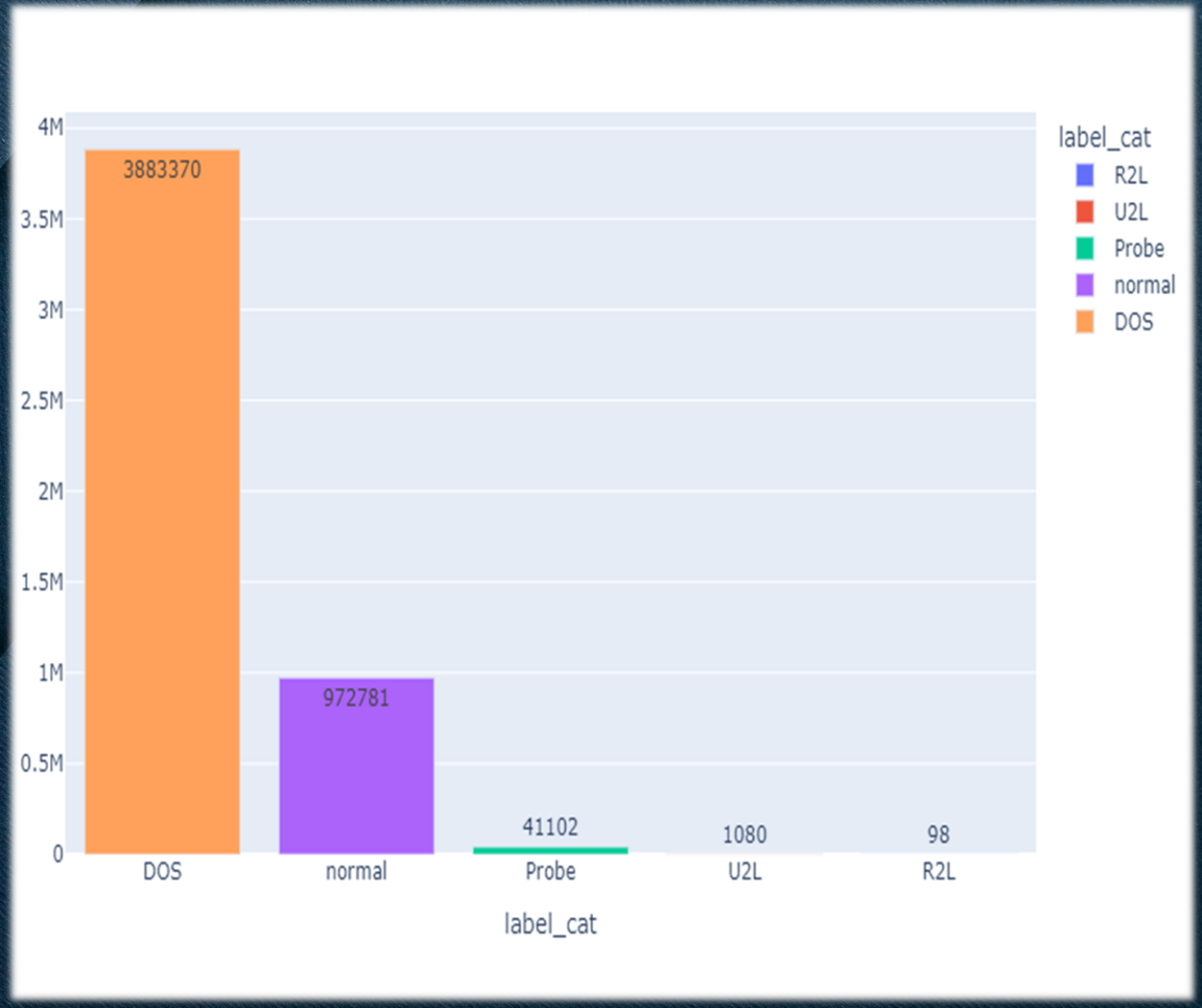
In [8]:
```
import pyspark.pandas as ps
psdf = sparkdf.to_pandas_on_spark()
```

In [12]:
```
psdf = df1.to_pandas_on_spark()
psdf.plot.bar(y = 'label', x = 'count(label)', color = 'label').update_yaxes(categoryorder="total ascending'
```



Data Exploration – encoded label class distribution

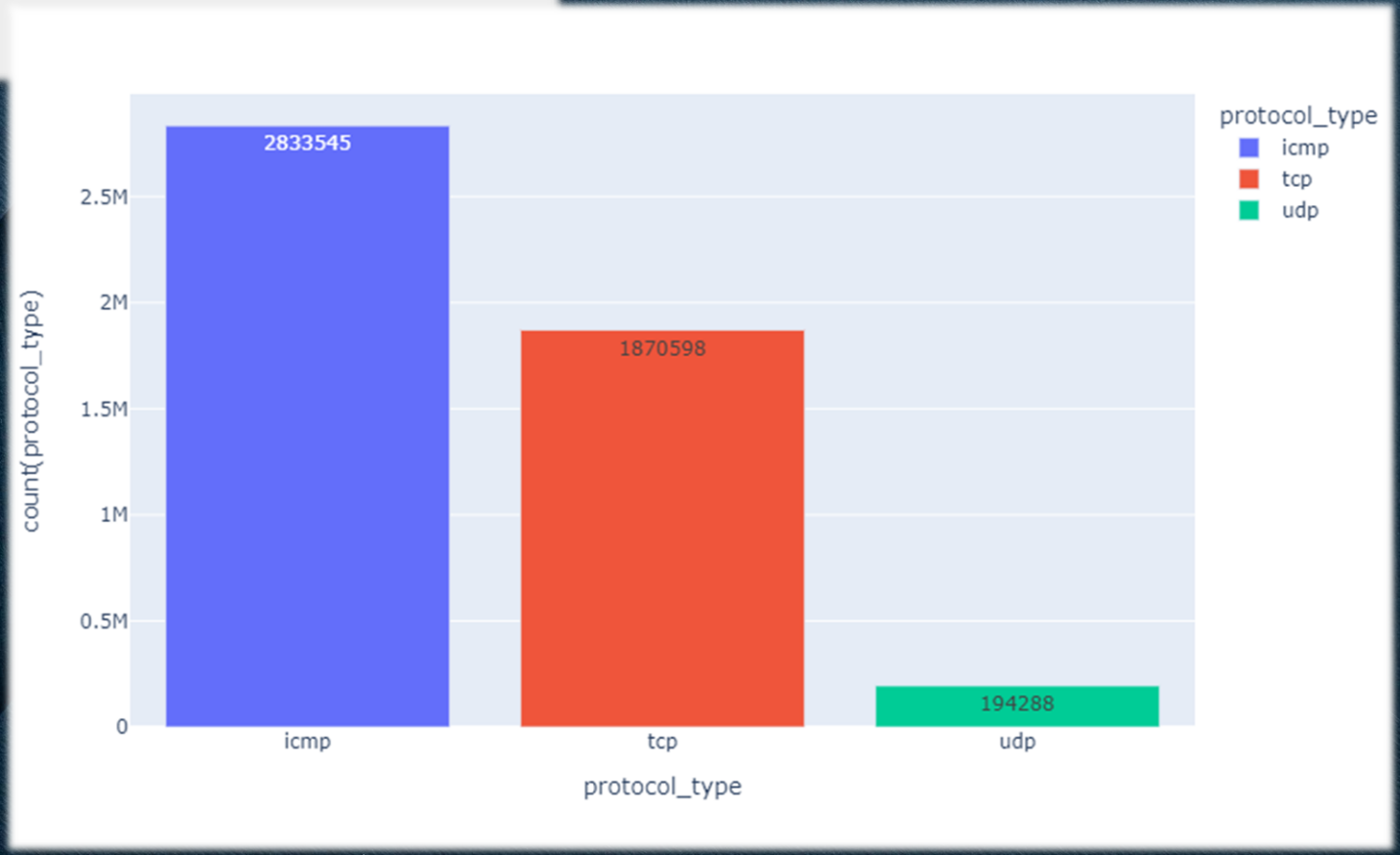
In [13]:
```
# Using pyspark sql to encode attack categories into 5 different categories
#0- normal, 1- DOS, 2- Probe, 3 - R2L(remote to user attack), 4 - U2R(User to Root attack)
# Creating a temp view
sparkdf.createOrReplaceTempView("network")

# adding label_num column
df_sql = spark.sql("""
SELECT *,
CASE
    WHEN label = 'normal' THEN 0
    WHEN label = 'back' OR label = 'smurf' OR label = 'pod' OR label = 'land' OR label = 'neptune' OR label
    WHEN label = 'satan' OR label = 'ipsweep' OR label = 'nmap' OR label = 'portsweep' THEN 2
    WHEN label = 'guess_passwd' OR label = 'ftpwrite' OR label = 'imap' OR label = 'phf' OR label = 'multih
    ELSE 4
END AS label_num,
CASE
    WHEN label = 'normal' THEN 'normal'
    WHEN label = 'back' OR label = 'smurf' OR label = 'pod' OR label = 'land' OR label = 'neptune' OR label
    WHEN label = 'satan' OR label = 'ipsweep' OR label = 'nmap' OR label = 'portsweep' THEN 'Probe'
    WHEN label = 'guess_passwd' OR label = 'ftpwrite' OR label = 'imap' OR label = 'phf' OR label = 'multih
    ELSE 'U2L'
END AS label_cat
FROM network;
""")
df_sql.select(['label_num','label_cat']).distinct().orderBy('label_num',ascending=True).show()
```



Data Exploration – Protocol type distribution

In [16]:

```
### Plotting Bar chart for protocol types

df3 = df_sql.groupby('protocol_type').agg(F.count('protocol_type')).orderBy('protocol_type',
                                                                            ascending= True)
psdf2 = df3.to_pandas_on_spark()
psdf2.plot.bar(y = 'count(protocol_type)', x = 'protocol_type', color = 'protocol_type',
               text = 'count(protocol_type)')
```



String Indexer

Creating a Pipeline to encode the qualitative variables which are of String datatype namely protocol_type, service, flag and label into variables of datatype Double by using StringIndexer Transformers.
This step is important as Logistic Regression and NaiveBayes classifiers needs to be trained on numerical values.

```
# Using String Indexer to encode Categorical variables to Numerical Variables
indexers = [StringIndexer(inputCol=column, outputCol=column + "_num") for column in qualitative_variables]


#Creating a pipeline
pipeline = Pipeline(stages=indexers)
```

Vector Assembler

Used Spark's Vector Assembler to create a feature vector for each numerical variable as we need to combine all the input columns into a single vector which would essentially act as the input feature for the classifiers.

In [29]:

```
df_assembler = VectorAssembler(inputCols=numerical_variables, outputCol="features")
df_sql = df_assembler.transform(df_sql)
#df_sql.printSchema()
```

Training and Testing Dataset

Created final dataframe ml_df having the columns features and label_num and split the dataset into Training Set (75%) and Testing Set (25%).

In [30]:
```
# Selecting vectorized features column and label_num
ml_df = df_sql.select(["features","label_num"])
ml_df.printSchema()

train_set, test_set = ml_df.randomSplit([0.75, 0.25], seed=3000)
print("Training dataset count: " + str(train_set.count()))
print("Test dataset count: " + str(test_set.count()))
```

```
root
 |-- features: vector (nullable = true)
 |-- label_num: integer (nullable = false)

Training dataset count: 3675127
Test dataset count: 1223304
```

Machine Learning Models Used

Logistic Regression

```
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0.8, featuresCol="features",
                        labelCol="label_num", family="multinomial")
```

Decision Tree

```
dt = DecisionTreeClassifier(labelCol="label_num", featuresCol="features",  maxBins=70)
```

Random Forest Classification

```
rf = RandomForestClassifier(labelCol="label_num", featuresCol="features", numTrees=20, maxBins=70)
```

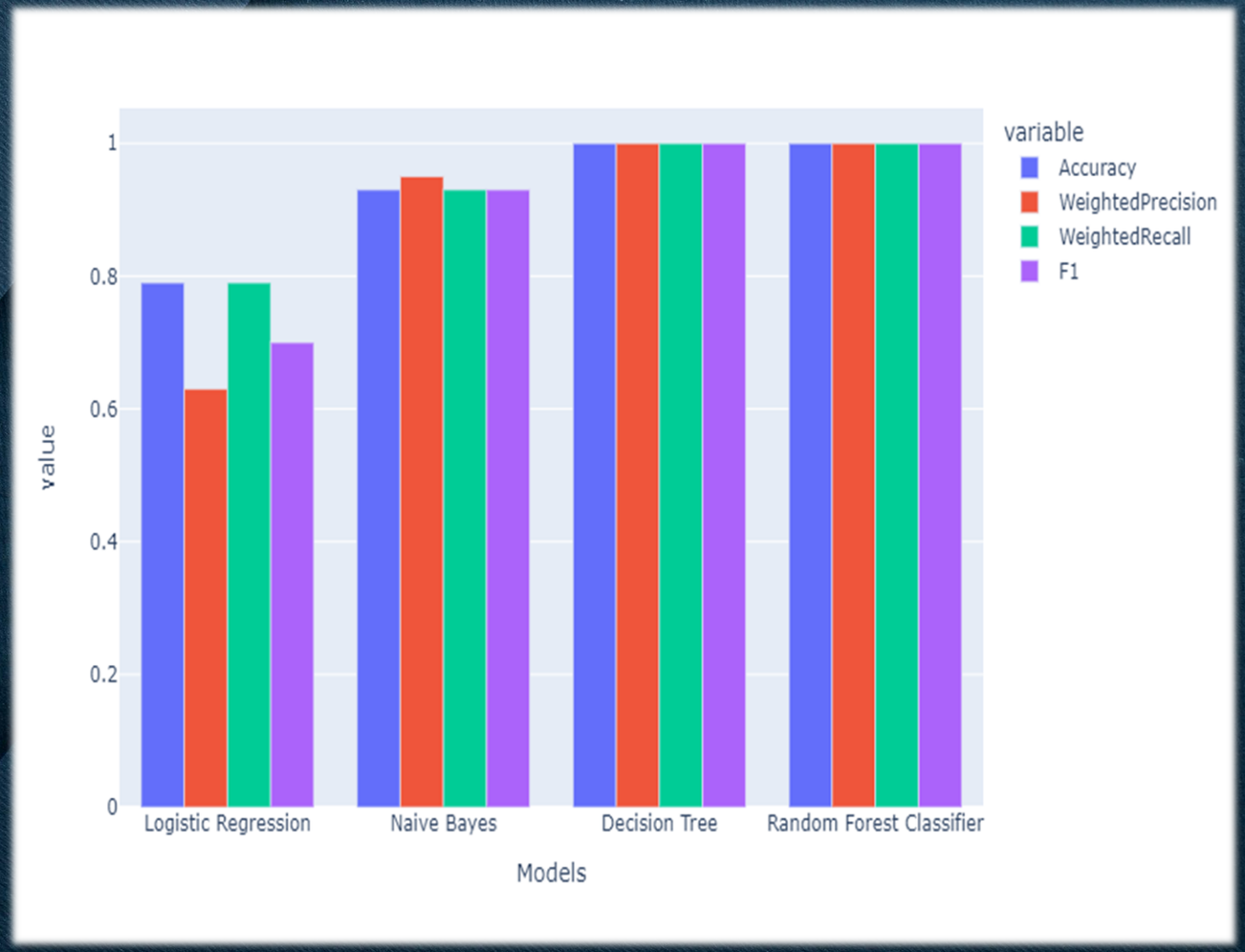
Naïve Bayes Multinomial

```
nb = NaiveBayes(labelCol="label_num", featuresCol="features", smoothing=1.0, modelType="multinomial")
```

Model Performance Evaluation

```
In [35]: metrics = ["accuracy", "weightedPrecision", "weightedRecall", "f1"]
```

Models	Accuracy	WeightedPrecision	WeightedRecall	F1
Logistic Regression	0.79	0.63	0.79	0.70
Decision Tree	1.00	1.00	1.00	1.00
Random Forest Classifier	1.00	1.00	1.00	1.00
Naive Bayes	0.93	0.95	0.93	0.93



Observations

- The best performance was achieved by Decision Tree and Random Forests classifiers which guarantees high performance scores for all the metrics.
- Naïve Bayes multinomial predicts our target variable with 93% accuracy which is not bad.
- As for the other evaluation metrics, Logistic Regression is characterized by quite good results except for the weighted precision which is rather low (63%)

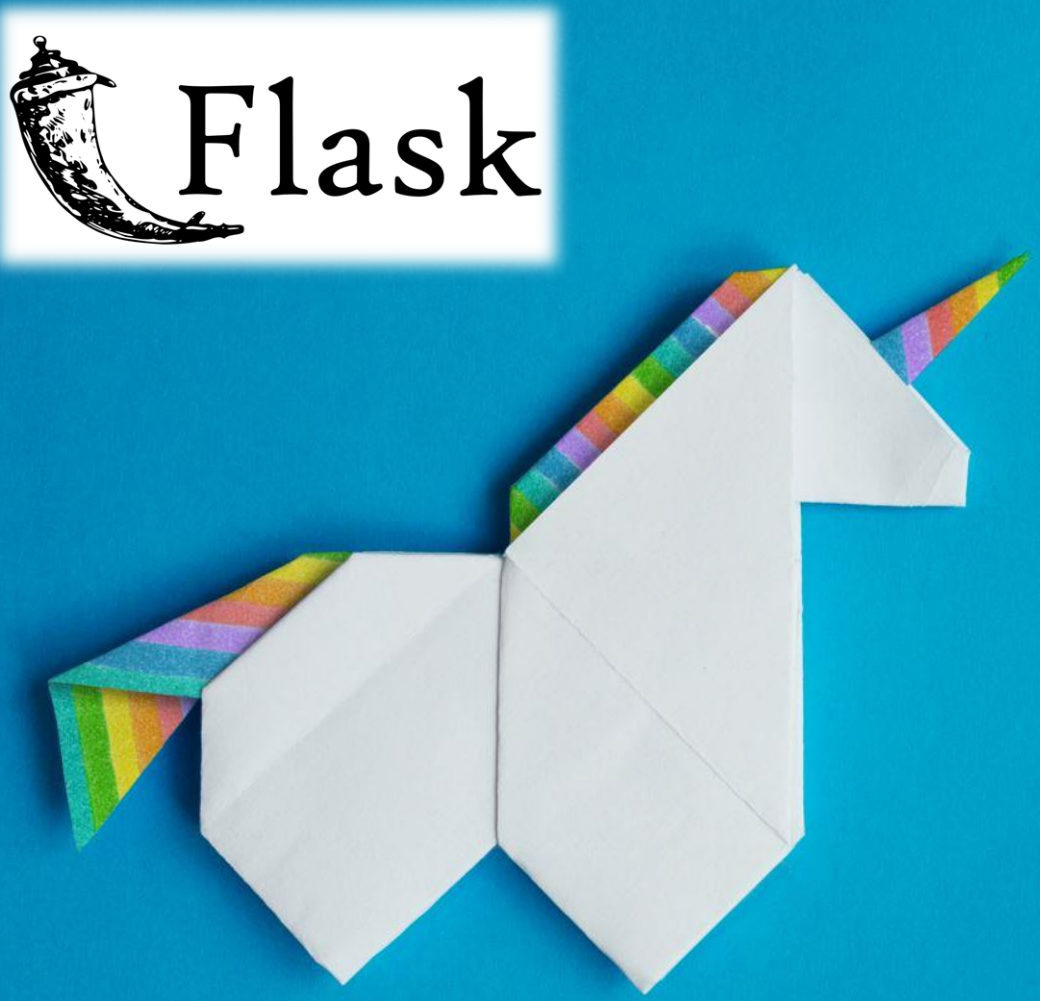
Saving our model - Reference

Using pickle module to save our best model

In [380]:

```
#!pip install prophet
from pyspark.sql.functions import *
from pyspark.sql.types import *
from prophet import Prophet
import pickle

import pickle
pkl_path = "model.pkl"
with open(pkl_path, "wb") as f:
    #Saving our decision tree model
    pickle.dump(dt_model, f)

# save the model
print("*** Data Saved ***")
```

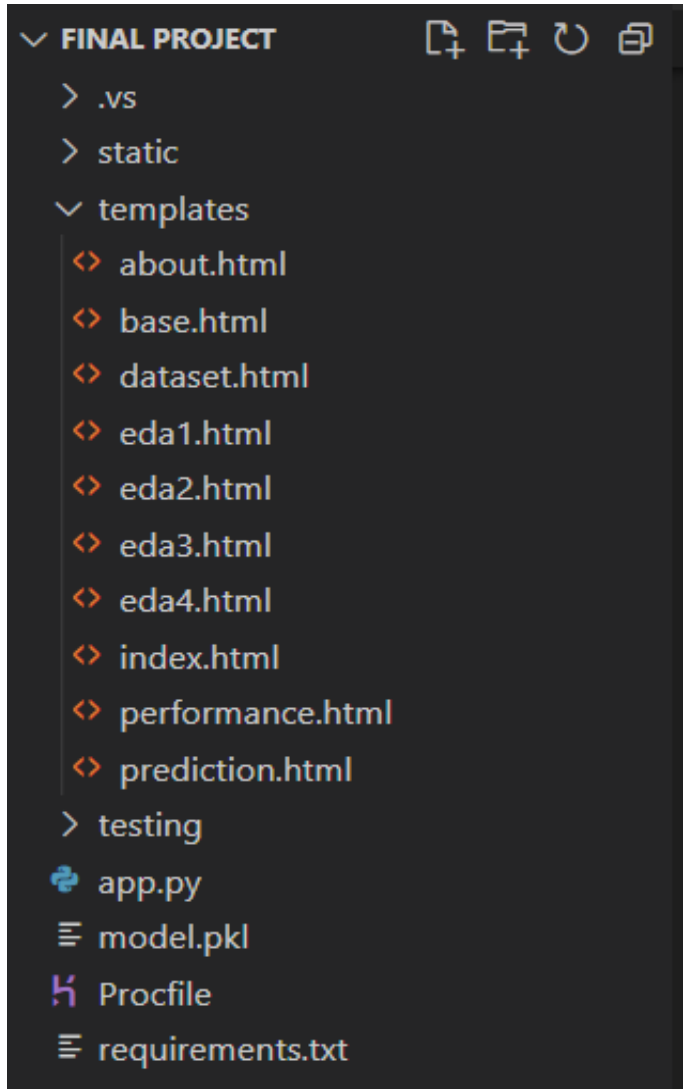
```
*** Data Saved ***
```



Website Design – Flask

Our website was created using Flask Framework.

We used HTML, CSS and JS for creating our web structure.



WEBSITE DESIGN

Website Design – Home Page

Home Dataset EDA Results Prediction Team

Network Intrusion Detection System

Big Data Project - Apache Spark - Flask

BDAT 1008 Final Project - Team 2

August 12, 2022

Website Design – Dataset

Home Dataset EDA Results Prediction Team

Information about the dataset

This data is KDD99 data set, which is widely used as one of the few publicly available data sets for network-based anomaly detection systems.

Dataset can be found here: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Github Link: https://github.com/shubhamchawla02/BDAT-1008-Final-Project

There are **4,898,431 rows** and **42 different attributes**

Label - Target Variable

- **DOS:** Denial of service is an attack category, which depletes the victim‟s resources thereby making it unable to handle legitimate requests – e.g. syn flooding. Relevant features: “source bytes” and “percentage of packets with errors”
- **robing:** Surveillance and other probing attack‟s objective is to gain information about the remote victim e.g. port scanning. Relevant features: “duration of connection” and “source bytes”
- **U2R:** unauthorized access to local super user (root) privileges is an attack type, by which an attacker uses a normal account to login into a victim system and tries to gain root/administrator privileges by exploiting some vulnerability in the victim e.g. buffer overflow attacks. Relevant features: “number of file creations” and “number of shell prompts invoked,”
- **R2L:** unauthorized access from a remote machine, the attacker intrudes into a remote machine and gains local access of the victim machine. E.g. password guessing Relevant features: Network level features – “duration of connection” and “service requested” and host level features - “number of failed login attempts”

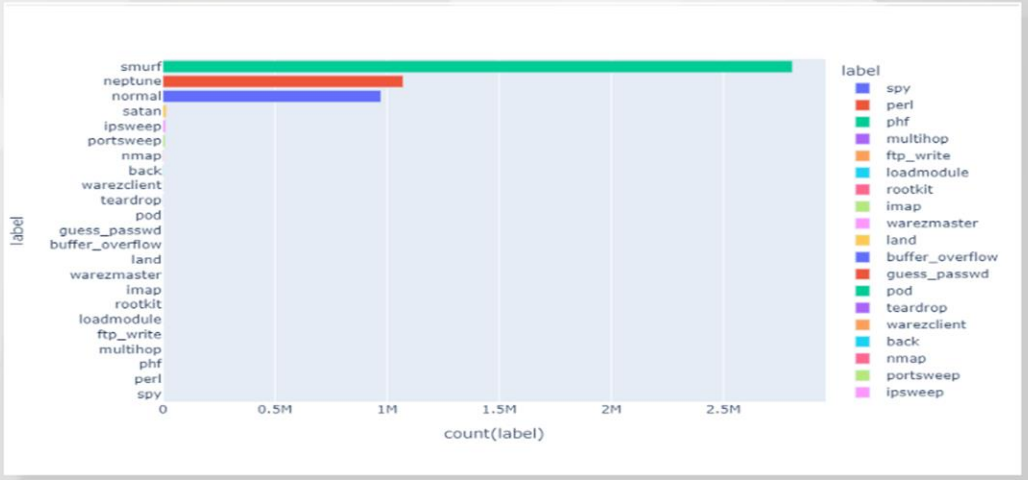
Basic Features of each network connection vector

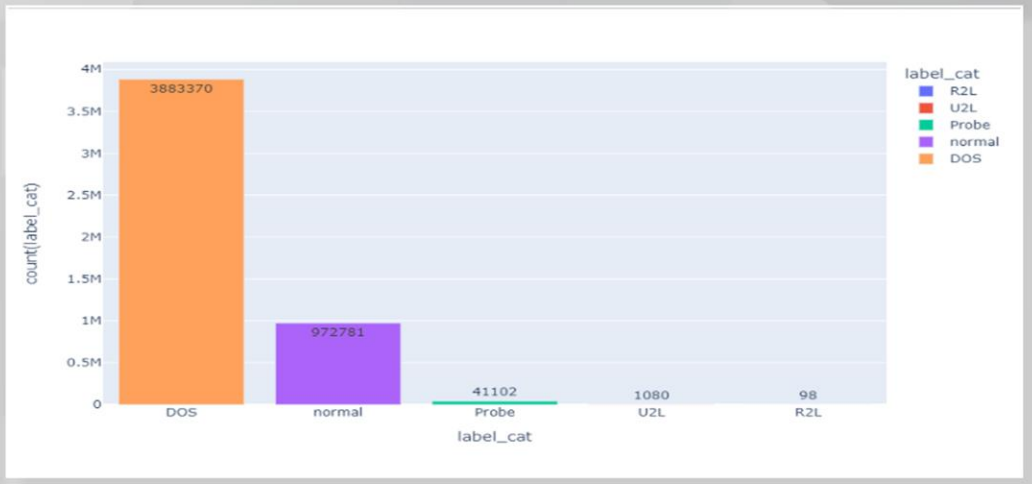
No	Feature	description	type

Website Design – EDA Attack Distribution

Home Dataset EDA Results Prediction Team

Attack Type Distribution





Observations

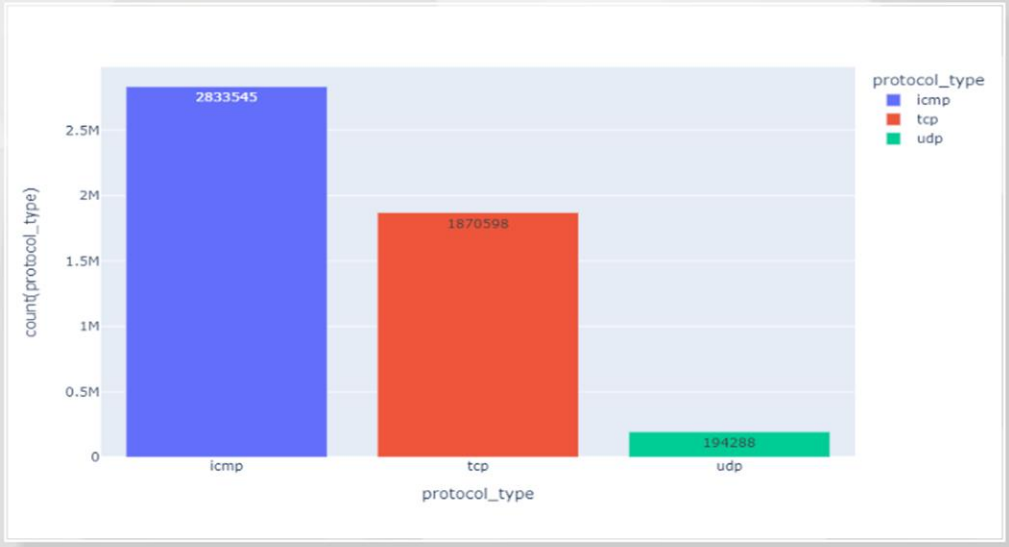
In the left plot we have 23 different kind of attacks and their distributions :

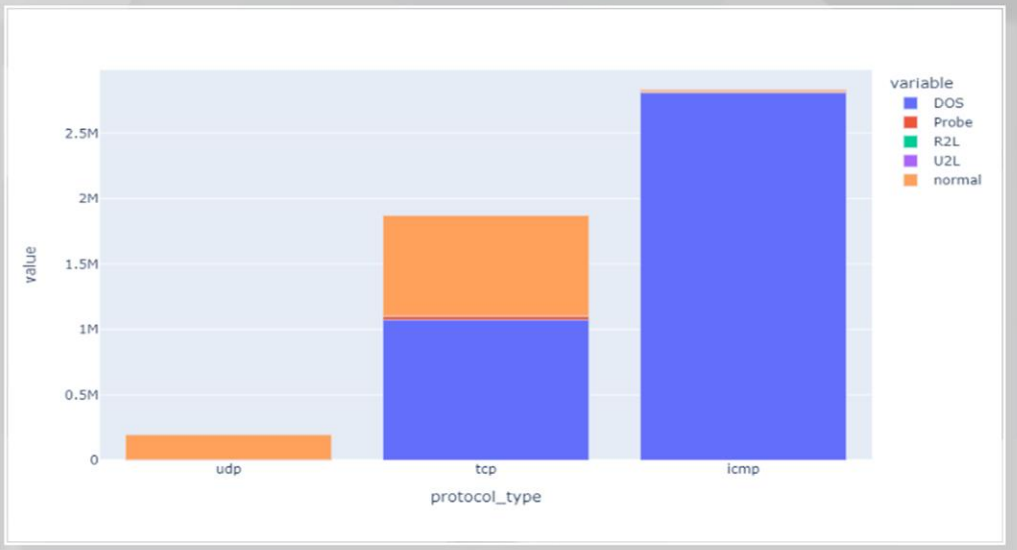
- Data set is not uniformly distributed as we can see in the above plots
- There are lots of attacks where data points are very few and some of the attacks like Normal, Neptune and Smurf 85% data points out of 100% data points
- There are 16 attacks out of 23 attacks where the data points are less then 1%
- There are way too many categories in the label to predict. So we encoded them according to 5 main categories - Normal, DOS, Probe, R2L and U2R which can be seen in the right plot

Website Design – EDA Protocol Distribution

Home Dataset EDA Results Prediction Team

Protocol Type Distribution





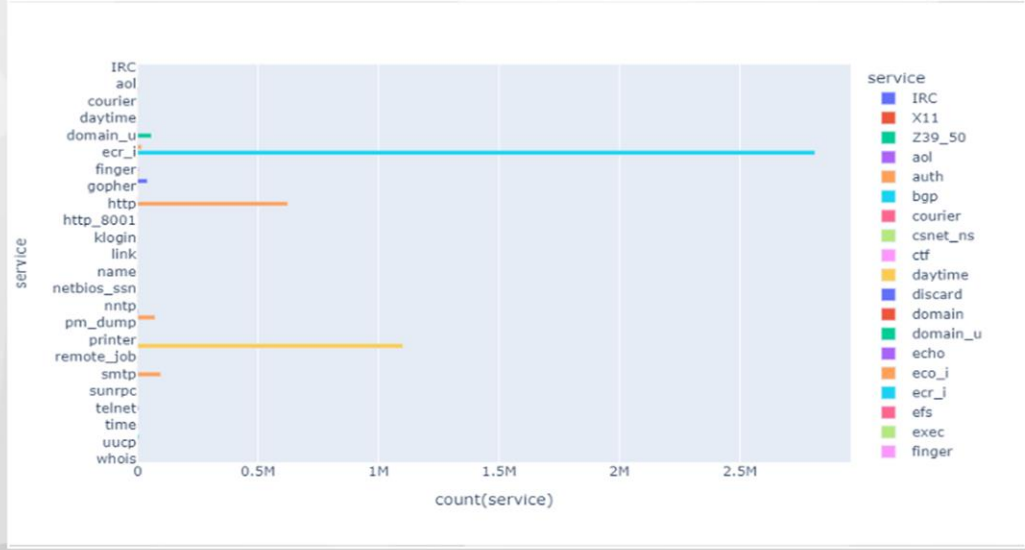
Observations

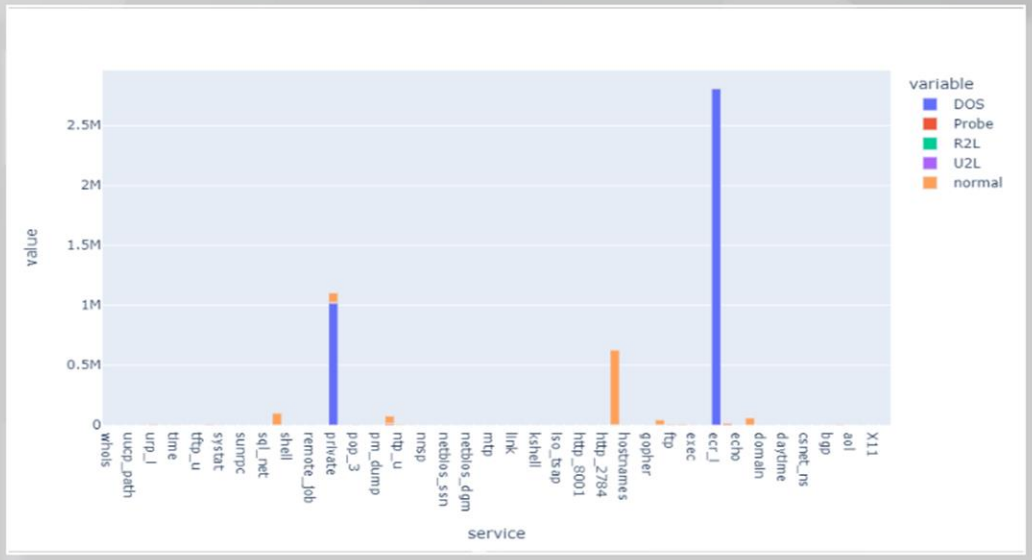
- We have 3 different category of protocols in the training data namely : TCP , UDP and ICMP
- ICMP has the highest number of data points followed by TCP and UDP
- On the right side we have plotted protocol types against our target variable
- Normal and DOS classes both are uniform only in term of TCP whereas UDP only shows up in normal class
- Majority of the ICMP protoype belongs to DOS class while there are only a few points belongs to normal class also

Website Design – EDA Service Distribution

Home Dataset EDA Results Prediction Team

Service Type Distribution





Observations

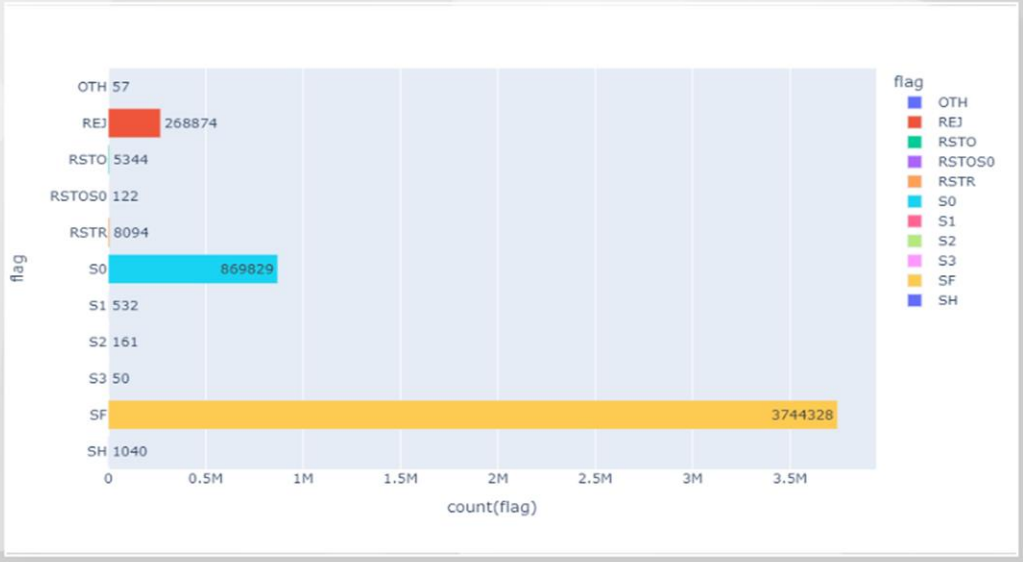
In the left plot we have 24 different kind of services and their distributions :

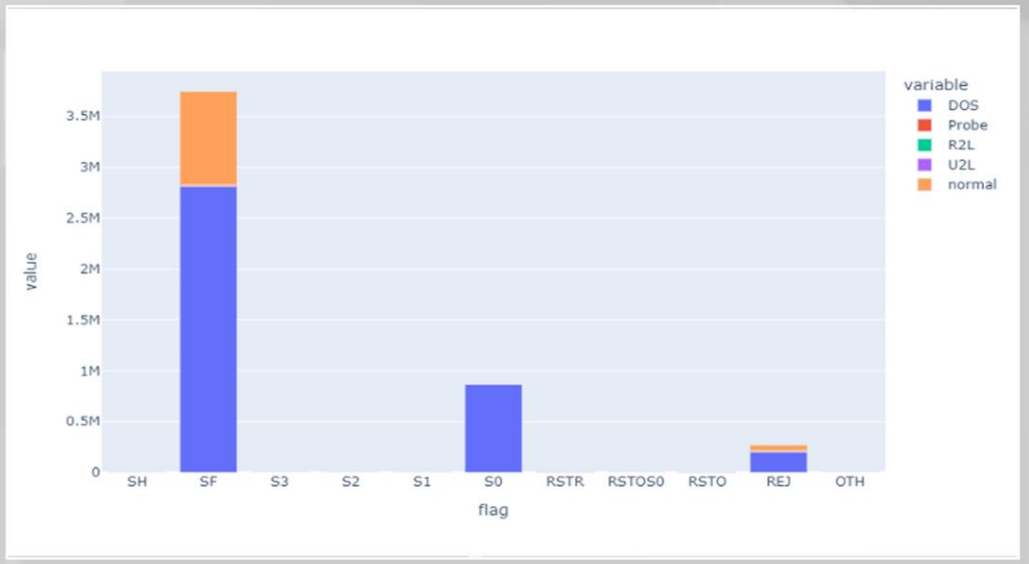
- There are three services ecr_i, http and private that dominates the dataset.
- On the right side we have plotted service type against our target variable
- Private service and ecr_i is dominated by DOS attack class
- http service is dominated by normal class which makes sense

Website Design – EDA Flag Distribution

Home Dataset EDA Results Prediction Team

Flag Type Distribution





Observations

In the left plot we have 11 different kind of flag types and their distributions :

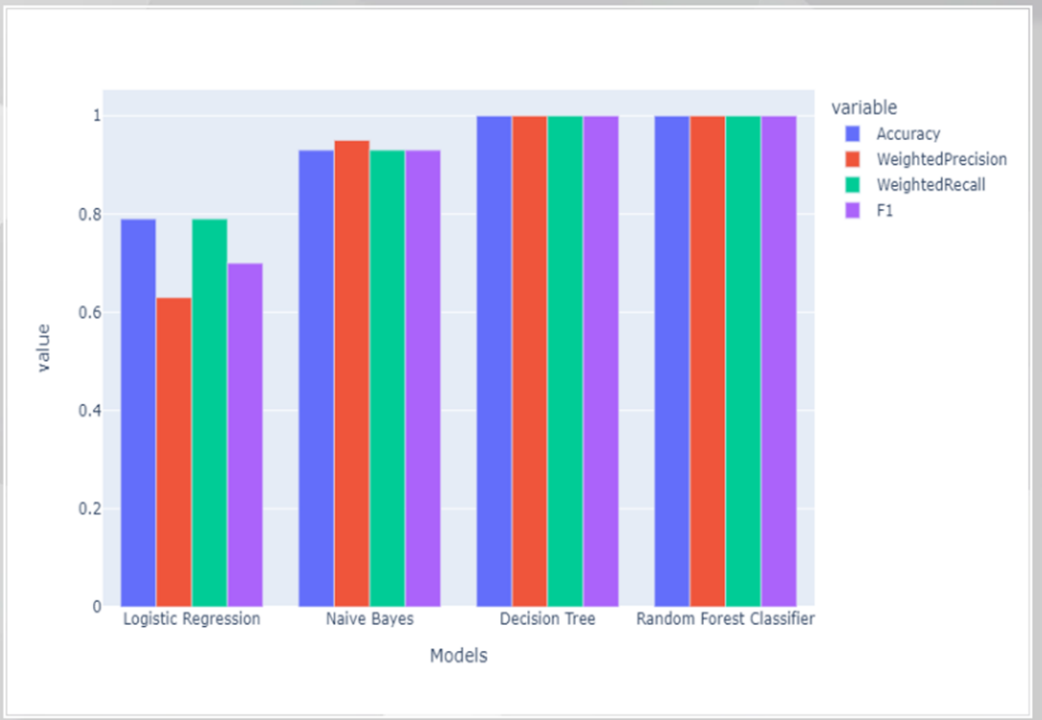
- There are three flags with most datapoints that are SF, SO and REj. Although, SF flag constitutes more than 80% pf the data
- On the right side we have plotted flag type against our target variable
- SF and SO flag is dominated by DOS attack
- REJ shows approximately equal distribution of normal and DOS attack class

Website Design – Results

Home Dataset EDA Results Prediction Team

Model Performance Comparison

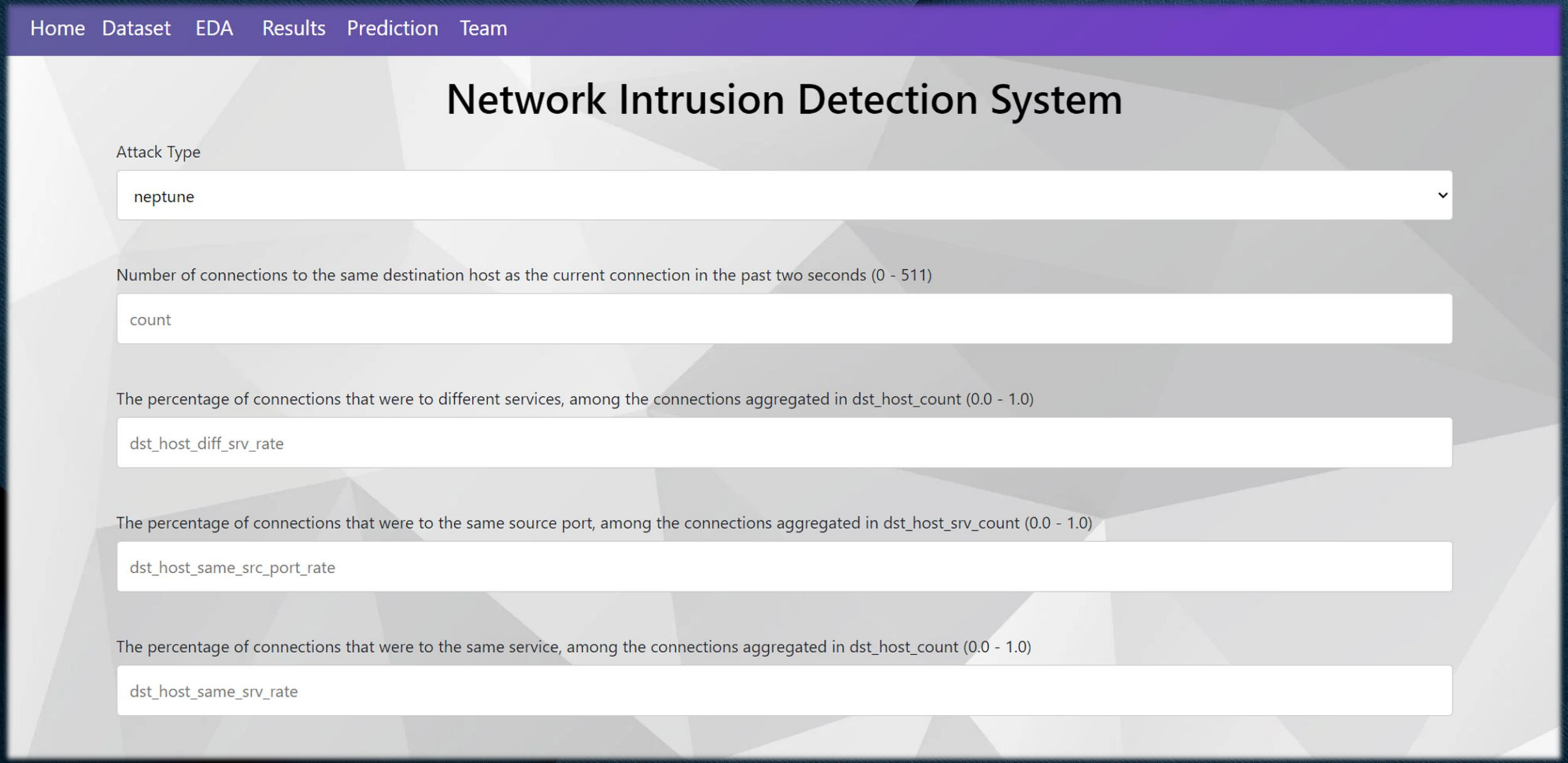
Models	Accuracy	WeightedPrecision	WeightedRecall	F1
Logistic Regression	0.79	0.63	0.79	0.70
Decision Tree	1.00	1.00	1.00	1.00
Random Forest Classifier	1.00	1.00	1.00	1.00
Naive Bayes	0.93	0.95	0.93	0.93



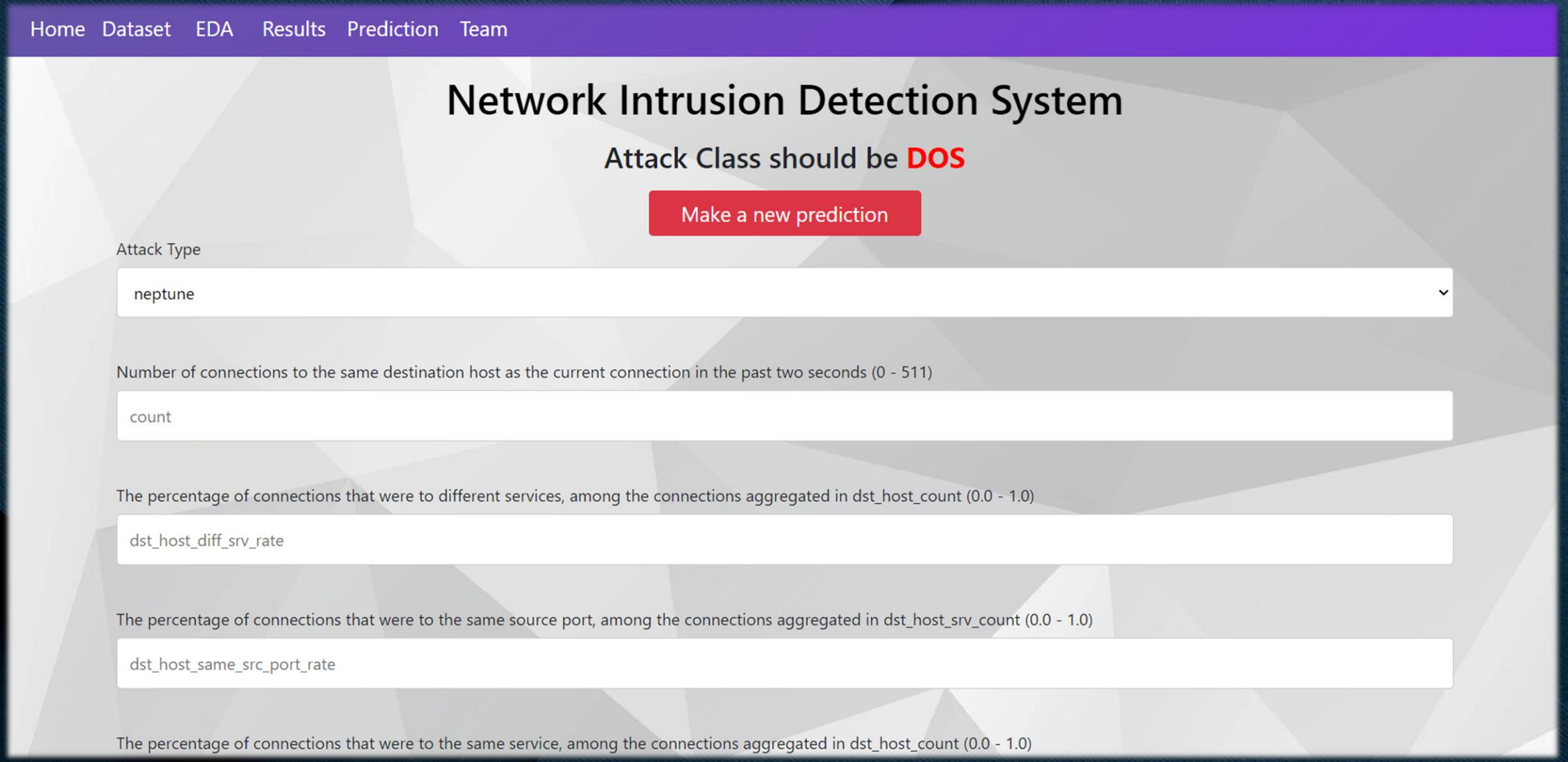
Observations

- The best performance was achieved by Decision Tree and Random Forests classifiers which guarantees high performance scores for all the metrics.
- As for the other evaluation metrics, Logistic Regression is characterized by quite good results except for the weighted precision which is rather low (63%)

Website Design – Prediction



Website Design – After making a prediction



Website Design – Prediction Code Implementation

```
app = Flask(__name__)
model = joblib.load('model.pkl')
```

```
@app.route('/results',methods=['POST'])
def results():

    data = request.get_json(force=True)
    predict = model.predict([np.array(list(data.values()))])

    if predict==0:
        output='Normal'
    elif predict==1:
        output='DOS'
    elif predict==2:
        output='PROBE'
    elif predict==3:
        output='R2L'
    else:
        output='U2R'

    return jsonify(output)
```

Using joblib library to load our saved model file and using POST method to send HTML form data to the server and then rendering our prediction page after predicting the attack class.

Website Design – Team

Home Dataset EDA Results Prediction Team

BDAT 1008 Data Collection & Curation Final Project Team 2

Shubham Chawla

200493036

Developer

shubham.chawla@mygeorgian.ca

LinkedIn

Archit Sinha

200505416

Business Analyst

archit.sinha2@mygeorgian.ca

LinkedIn

Chintan Vajani

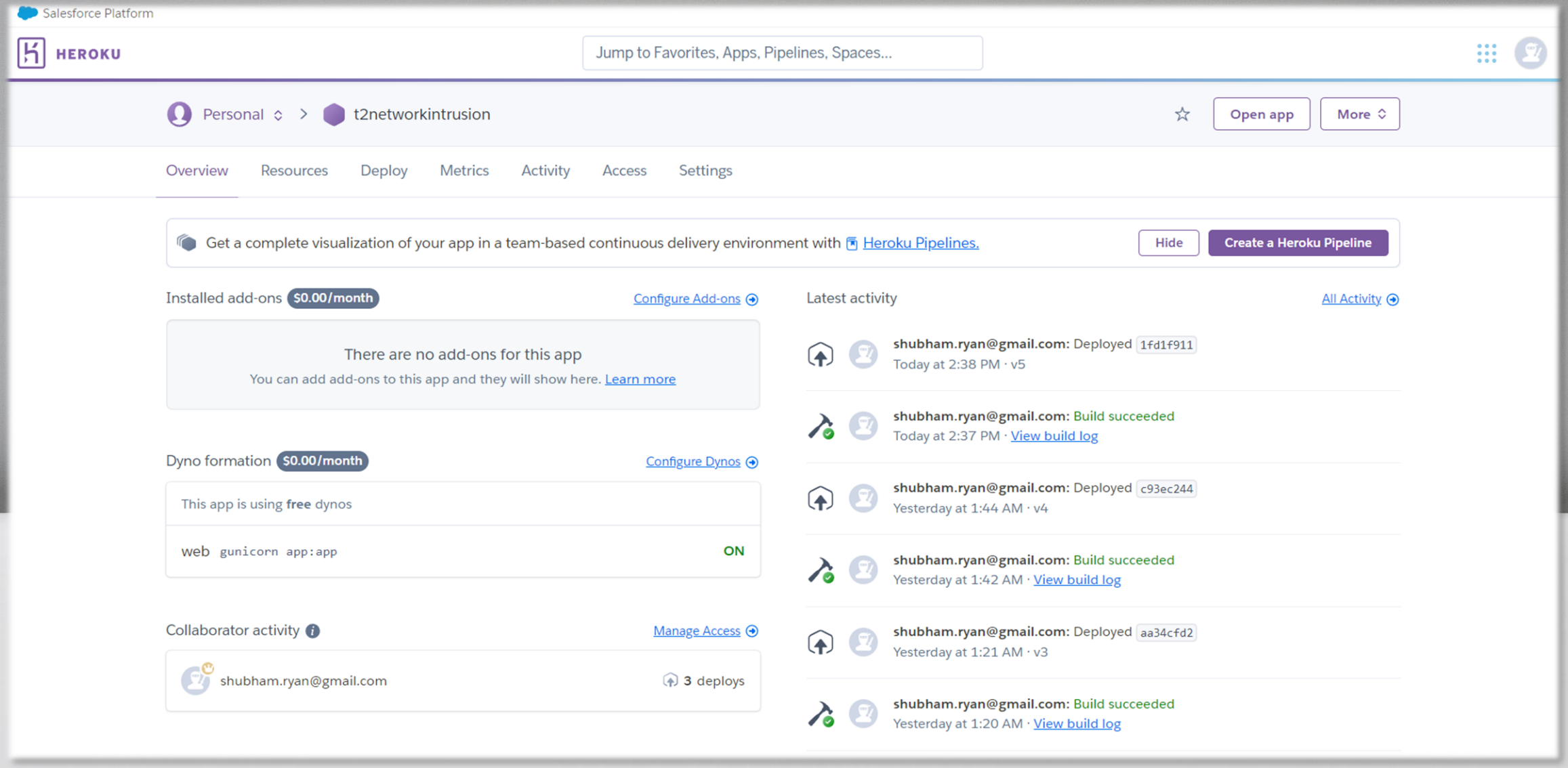
200508118

Project Manager

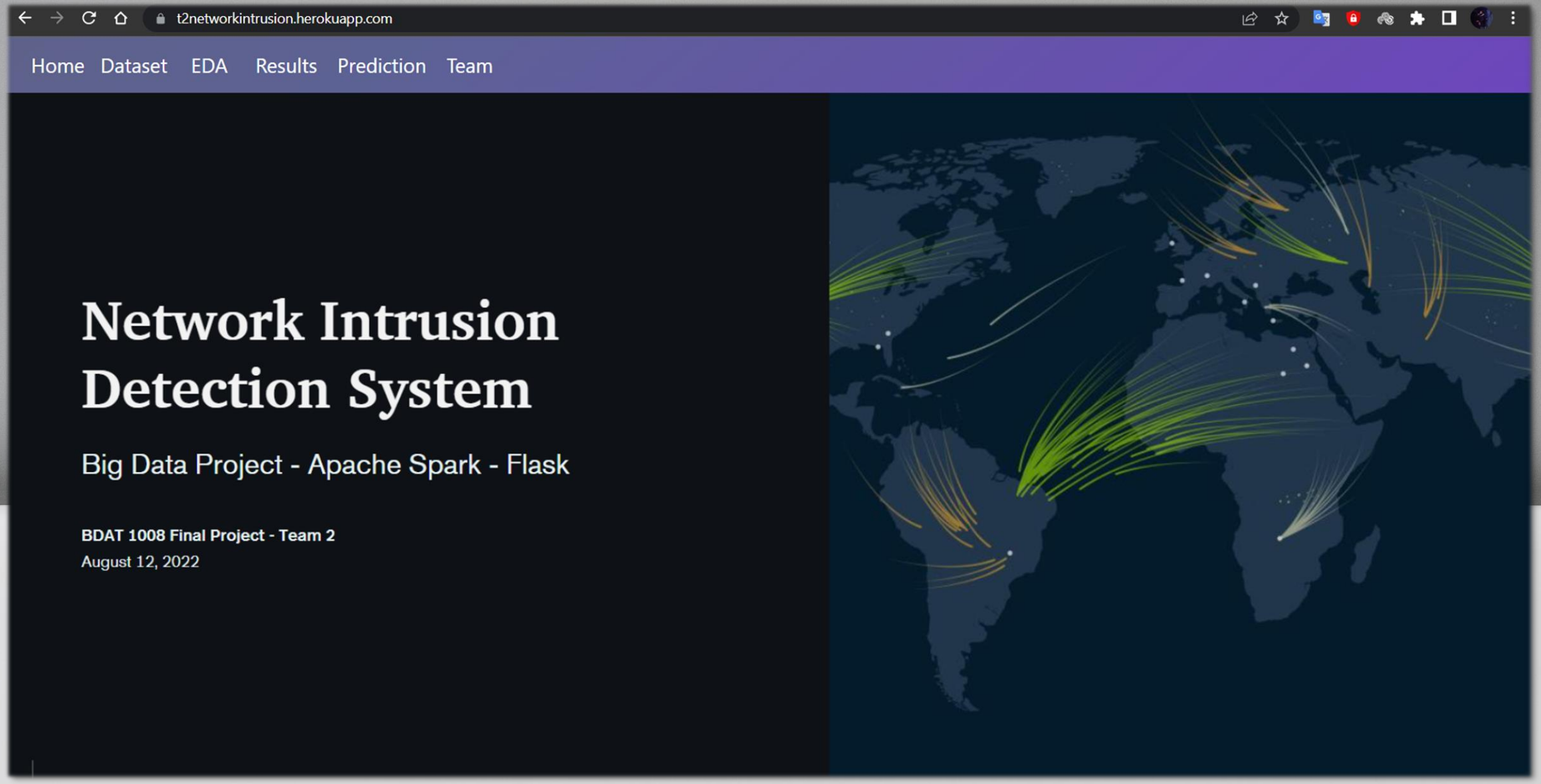
chintan.vajani@mygeorgian.ca

LinkedIn

Website Deployment – Heroku – t2networkintrusion



Website Deployment – Heroku – t2networkintrusion



Fit – Gap Analysis (1)

- ✓ **More Data = Wider Purview**
  Adding more data to analysis can also help gain a broader and more complete perspective on a business problem.

- ✓ **More Visualizations** could be added by writing code in Python

- ✓ **Javascript chart libraries could be integrated for having much more variety and customizations in the visualizations.** For e.g., plotly.js, highcharts etc.

- ✓ **Adding interactive Filters for the User to analyze network attacks relationships among different variables according to their needs.**

References

https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.StringIndexer.html

https://medium.com/udemy-engineering/pyspark-under-the-hood-randomsplit-and-sample-inconsistencies-examined-7c6ec62644bc

https://suziepyspark.blogspot.com/2021/03/using-pickle-to-save-model-in-incorta.html

https://sparkbyexamples.com/pyspark/different-ways-to-create-dataframe-in-pyspark/

https://github.com/biagiom/spark-network-traffic-classifier

https://unsplash.com/

https://www.barracuda.com/glossary/intrusion-detection-system

https://www.ecb.torontomu.ca/~bagheri/papers/cisda.pdf

Thank You