TEAM 9

Table of Contents

Contents

Introduction

Congestion, in the context of networks, refers to a network state where a node or link carries so much data that it may deteriorate network service quality, resulting in queuing delay, frame or data packet loss and the blocking of new connections. In a congested network, response time slows with reduced network throughput. Congestion occurs due to the presence of too many hosts in the broadcast domain, broadcast storm or when bandwidth is insufficient and network data traffic exceeds capacity.

There are mainly two types of congestion: RAN (Radio Access Network) and Backhaul. Backhaul is the use of communications systems to get data from an end user to a node in a major network such as the Internet or the proprietary network of a large business, academic institution or government agency. In a RAN, radio sites provide radio access and coordinate management of resources across the radio sites. A device is wirelessly connected to the core network, and the RAN transmits its signal to various wireless endpoints, and the signal travels with another networks' traffic. Congestion across such networks are called Backhaul Congestion and RAN Congestion respectively.

Congestion leads to a negative impact on customer loyalty, especially in price-sensitive markets. Many congestion control and avoidance techniques are used to avoid such collapse but all these techniques are used after the occurrence of such downfall. But what if we could predict congestion in advance and take proactive action to prevent congestion based on the predicted data? From the given dataset, we aim to make a predictive analysis based on machine learning models, regarding the type of congestion (if any) expected in specific cellular networks. The data already provided consists of information regarding the type of congestion in specific cell towers in December 2018 from which an attempt is made to predict the type of network congestion in the future.

Data Preprocessing

Machine learning tools are only as good as the quality of data. Sophisticated algorithms will not make up for poor data. Data needs to go through a few processes before it is ready for further use. For achieving better results from the applied model in Machine Learning projects, the format of the data has to be in a proper manner and format. Also, the data set should be formatted in such a way that more than one algorithm can be executed in one data set, and best out of them is chosen. We undertook the following procedures:

I. **Taking care of missing data**
   The given dataset didn't have any empty cells, i.e., no missing values. Hence, we did not need to bother about taking care of missing data.

II. **Label Encoding**
   The columns, 'ran_vendor' and 'Congestion Type', of the dataset are in text form and are termed as categorical variables. Models understand numbers, so we encode these

categorical variables with numbers. We have used LabelEncoder, which maps text to 0, 1, 2… and encodes them.

III. **Feature Scaling (normalization, standardization)**
It is a method used to standardize the range of independent variables or features of the dataset. It is necessary to scale all our variables, in order to speed up the algorithm. We have used MinMaxScaler() to scale all features in a range of 0 to 1.

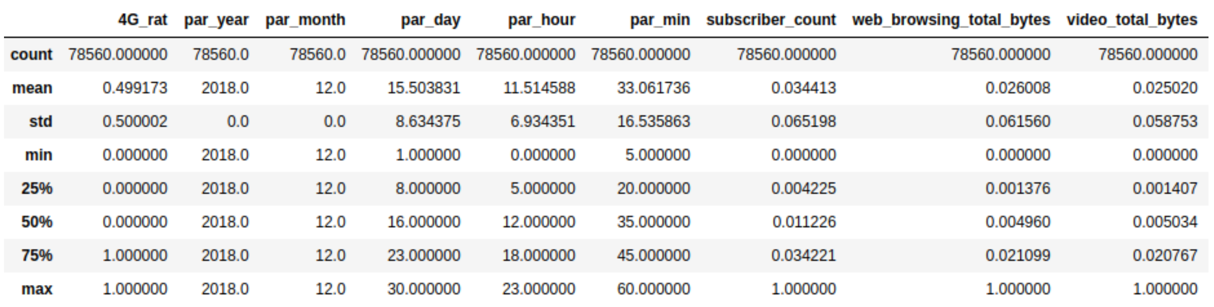
IV. **Splitting into Training Set and Test Set:**
We need to split our dataset into two sets — a Training set and a Test set. We will train our machine learning models on this training set and then test the models on the test set to check how accurately it can predict. We have used 'train_test_split' for this purpose.

Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations.

Summary statistics of the data

We get the count, mean, standard deviation, minimum and maximum values and the quantiles of the data to get a better idea of its distribution.



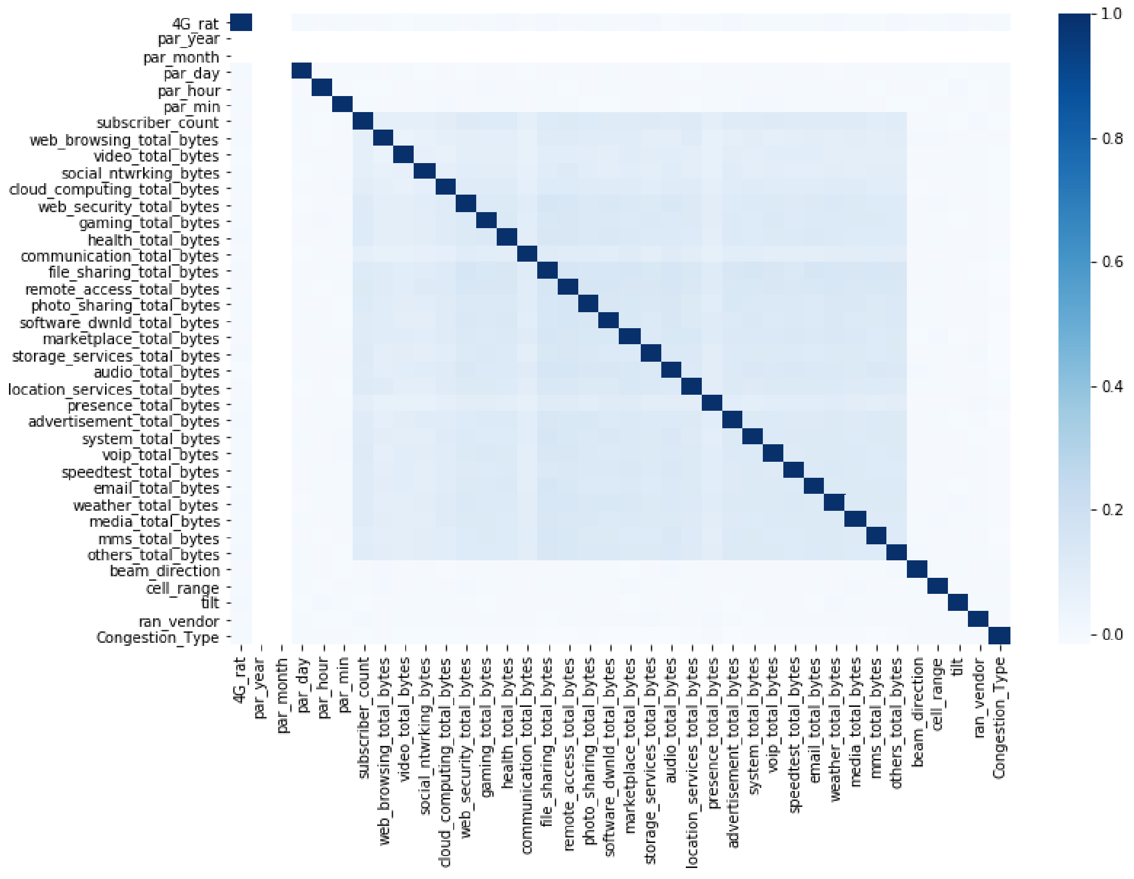
	4G_rat	par_year	par_month	par_day	par_hour	par_min	subscriber_count	web_browsing_total_bytes	video_total_bytes
count	78560.000000	78560.0	78560.0	78560.000000	78560.000000	78560.000000	78560.000000	78560.000000	78560.000000
mean	0.499173	2018.0	12.0	15.503831	11.514588	33.061736	0.034413	0.026008	0.025020
std	0.500002	0.0	0.0	8.634375	6.934351	16.535863	0.065198	0.061560	0.058753
min	0.000000	2018.0	12.0	1.000000	0.000000	5.000000	0.000000	0.000000	0.000000
25%	0.000000	2018.0	12.0	8.000000	5.000000	20.000000	0.004225	0.001376	0.001407
50%	0.000000	2018.0	12.0	16.000000	12.000000	35.000000	0.011226	0.004960	0.005034
75%	1.000000	2018.0	12.0	23.000000	18.000000	45.000000	0.034221	0.021099	0.020767
max	1.000000	2018.0	12.0	30.000000	23.000000	60.000000	1.000000	1.000000	1.000000

There is a notable difference between 75th %tile and max values of features depicting bytes consumed -"usage data". This suggests that there are extreme values, i.e, outliers in our dataset. However, later we will see that these outliers play a major role in determining the type of congestion.

Data Visualization

I. Correlation plot

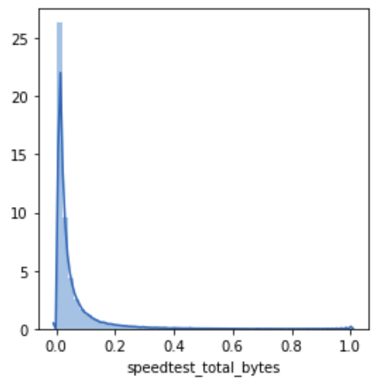
We have plotted the correlation graph and checked its results. From the graph, it is clear that the correlation among the features is not very strong indicating that none of the quantities need to be removed during feature selection.

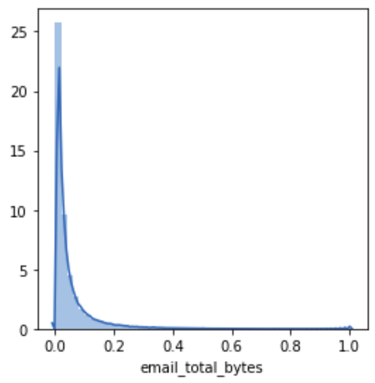


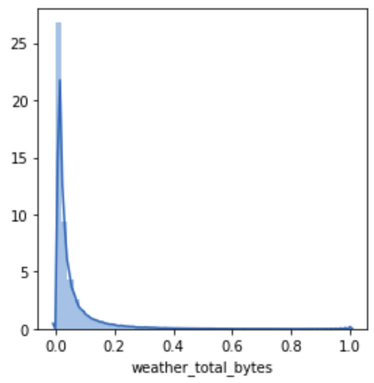
Correlation plot

II. Skewness of features

Now to check the linearity of the variables it is a good practice to plot its distribution graph and look for skewness.
We see that the subscriber count and ‘usage data’ features are positively skewed. Hence, we have applied a ‘logarithmic transformation’ on these features to bring them closer to a normal curve.





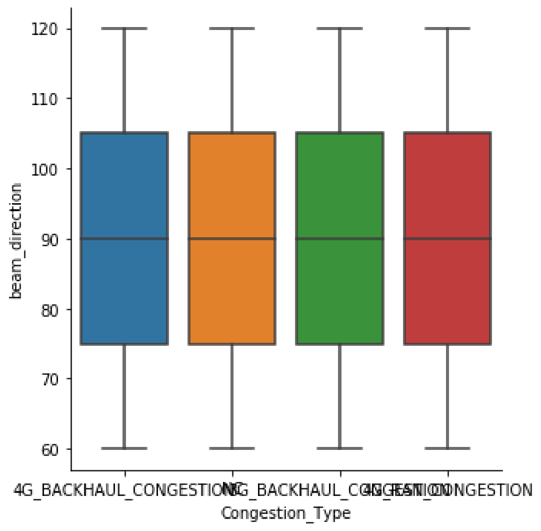


III. Box plot grouped by congestion type

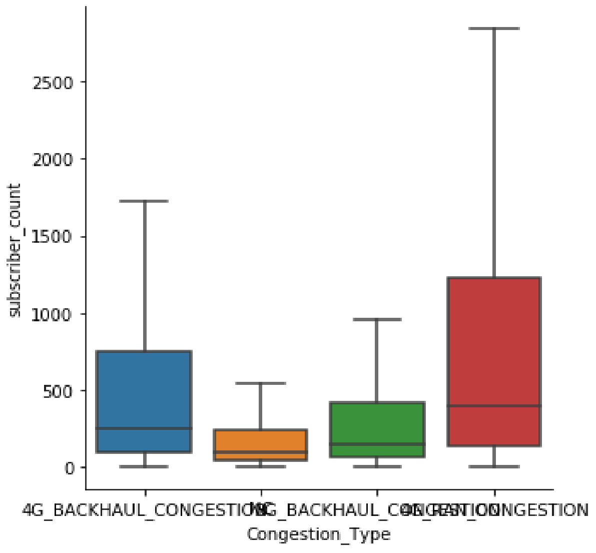
We have plotted boxplots for three features, namely, subscriber_count, total_bytes (adding all 'usage data' features) and beam_direction. We can clearly see the differences in their nature and conclude the following:

1. In the beam_direction boxplot, all 4 types of congestion are equally and symmetrically distributed. This means that the beam direction will have no effect on congestion type.
2. The subscriber_count and the total_bytes are unequally distributed which means that these features will have more effect on congestion type.

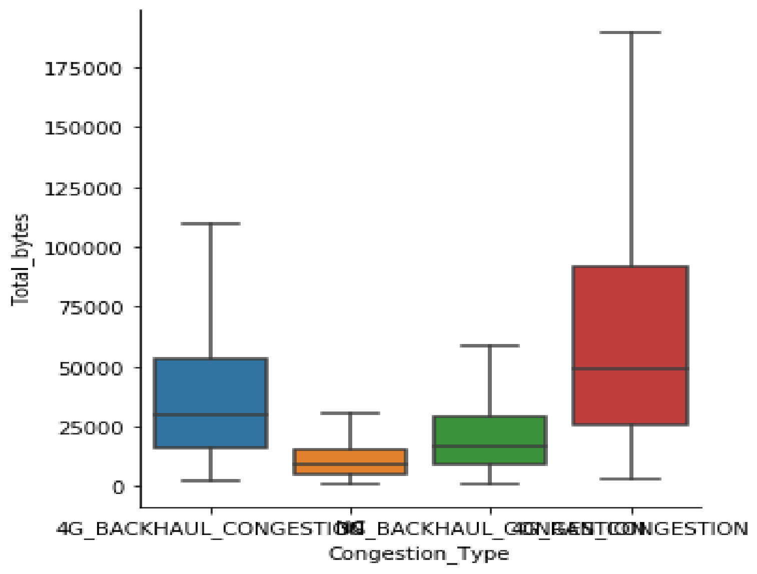
Hence, at this point, it seems that the subscriber_count and 'usage data' features will have the greatest effect in determining the type of congestion.



Beam direction boxplot

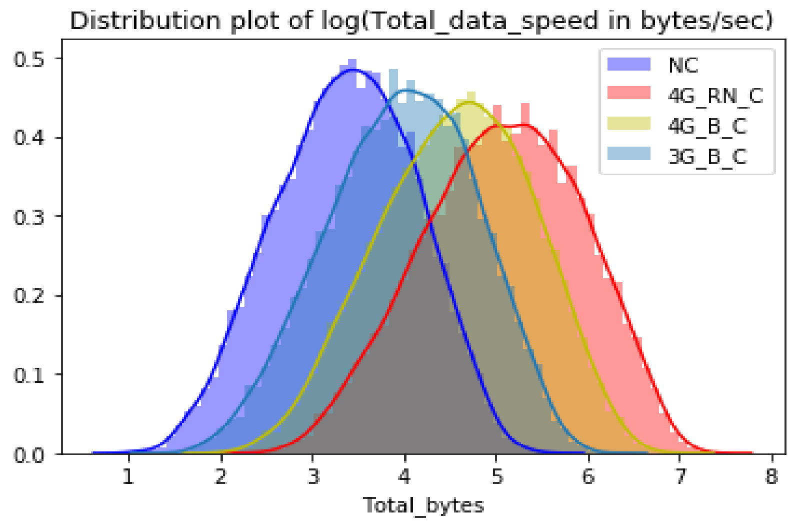


Subscriber count boxplot



Total bytes boxplot

IV. Distribution plot



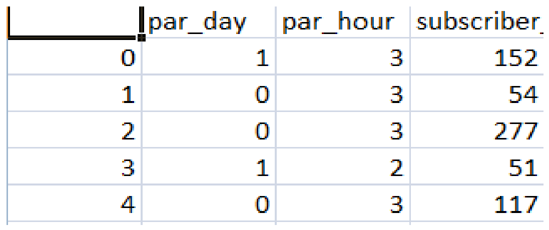
Distribution plot of total bytes

We can interpret the distribution plot as follows: Suppose that we draw a vertical line at a particular value on the x-axis (data speed). This line will cut the distribution curves of the congestion type at different values. These values will suggest the chances (which is directly proportional to frequency) of a specific type of congestion at that data speed.

Feature Engineering

A feature is an attribute or property shared by all of the independent units on which analysis or prediction is to be done. The features in data are important to the predictive models used and will influence the results obtained. The quality and quantity of the features have a great influence on the performance of the model.
Some features regarding cell towers were already present in the dataset, like cell name, day and time of usage, number of subscribers and the bytes consumed in different types of online activities.

1. The **par_day** values were classified into two **groups -Weekends**(Saturday and Sunday) **and Weekdays**, which were encoded with the values 0 and 1 respectively.
2. The **par_hour** values were classified into bins of 0-3, (2am-8am; 8am-6pm; 6pm-10pm and 10pm-2am) **based on the expected range of web-traffic in time intervals of the day** .



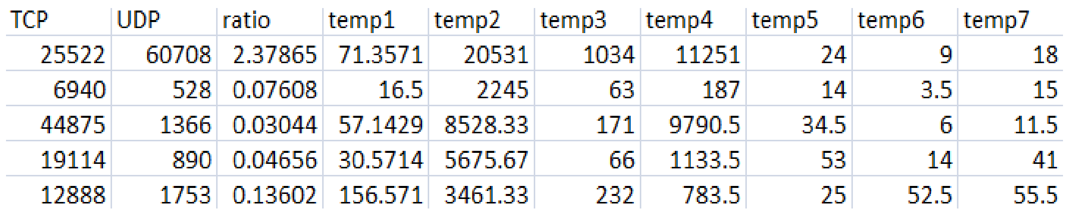
	par_day	par_hour	subscriber_
0	1	3	152
1	0	3	54
2	0	3	277
3	1	2	51
4	0	3	117

3. TCP (**Transmission Control Protocol**) is connection oriented, i.e. it tracks all data sent, requiring acknowledgment. Besides being responsible for error checking and correcting, is also responsible for controlling the speed at which this data is sent. TCP is capable of detecting congestion in the network and will back off transmission speed when congestion occurs.
TCP = web_browsing_total_bytes + social_ntwrking_bytes + health_total_bytes + communication_total_bytes + file_sharing_total_bytes + remote_access_total_bytes + location_services_total_bytes + presence_total_bytes + advertisement_total_bytes + speedtest_total_bytes + email_total_bytes + weather_total_bytes + mms_total_bytes + others_total_bytes

4. UDP (**User Datagram Protocol**) is connection-less, i.e. it does not use acknowledgments at all. It does not provide congestion control but allows the fastest and most simple way of transmitting data to the receiver because error recovery is not attempted.
UDP = video_total_bytes + cloud_computing_total_bytes + web_security_total_bytes + gaming_total_bytes + photo_sharing_total_bytes + software_dwnld_total_bytes + marketplace_total_bytes + storage_services_total_bytes + audio_total_bytes+ system_total_bytes + media_total_bytes + voip_total_bytes.

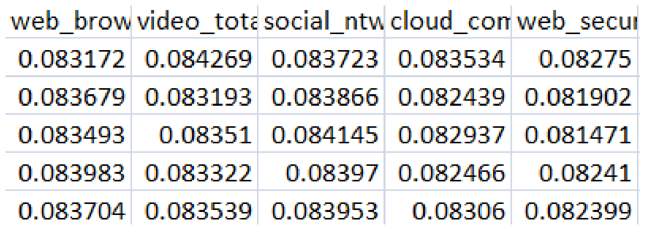
5. Further, we added another feature, calculating the **ratio** of the two newly created features **(UDP ÷ TCP).**
6. Measures like **mean, standard deviation, variance were examined** for each of the categories of data usage.Data bytes variables which have similar distribution into single variable and then averaged that variable down so as to keep the scale same in all the new variables with the old variables.
    a. **temp1** = web_security_total_bytes + gaming_total_bytes + storage_services_total_bytes + system_total_bytes + media_total_bytes
    b. **temp2** = health_total_bytes + 'location_services_total_bytes + 'advertisement_total_bytes + voip_total_bytes + 'speedtest_total_bytes + 'email_total_bytes + weather_total_bytes + mms_total_bytes + others_total_bytes
    c. **temp3** = web_browsing_total_bytes + social_ntwrking_bytes
    d. **temp4** = presence_total_bytes + communication_total_bytes
    e. **temp5** = photo_sharing_total_bytes + software_dwnld_total_bytes
    f. **temp6** = file_sharing_total_bytes + remote_access_total_bytes
    g. **temp7** = marketplace_total_bytes + audio_total_bytes

Further, to have a mean value, the temp values were replaced by the value obtained by dividing each temp value by the number of items falling in each category.
($Temp_i$ = $Temp_i \div n_i$)



TCP	UDP	ratio	temp1	temp2	temp3	temp4	temp5	temp6	temp7
25522	60708	2.37865	71.3571	20531	1034	11251	24	9	18
6940	528	0.07608	16.5	2245	63	187	14	3.5	15
44875	1366	0.03044	57.1429	8528.33	171	9790.5	34.5	6	11.5
19114	890	0.04656	30.5714	5675.67	66	1133.5	53	14	41
12888	1753	0.13602	156.571	3461.33	232	783.5	25	52.5	55.5

7. **Log Transformation** was also applied to all the categories of Data Usage, in order to scale them down to a comparable range. (and also remove the skewness that would have otherwise appeared in the plot of the unscaled data).
$y_{new} = \log(y_{old} + 1)$



web_brow	video_tota	social_ntw	cloud_con	web_secu
0.083172	0.084269	0.083723	0.083534	0.08275
0.083679	0.083193	0.083866	0.082439	0.081902
0.083493	0.08351	0.084145	0.082937	0.081471
0.083983	0.083322	0.08397	0.082466	0.08241
0.083704	0.083539	0.083953	0.08306	0.082399

8. These transformed values of Data Usage were all summed up to create a new feature named **Total_Bytes**

Feature Selection

Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested. Having irrelevant features in your data can decrease the accuracy of many models, especially linear algorithms like linear and logistic regression.

Three benefits of performing feature selection before modeling your data are: Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise. Improves Accuracy: Less misleading data means modeling accuracy improves. Reduces Training Time: Less data means that algorithms train faster.

Univariate Selection

Statistical tests can be used to select those features that have the strongest relationships with the output variable. The scikit-learn library provides the SelectKBest class, which we have used with a suite of different statistical tests to select a specific number of features.

The 4 most important features, with their scores, which this class gives are:

1) TCP - 2.376e+04 **2) UDP - 1.602e+04**
3) temp6 - 1.233e+04 **4) total_bytes - 1.338e+03**

Recursive Feature Elimination

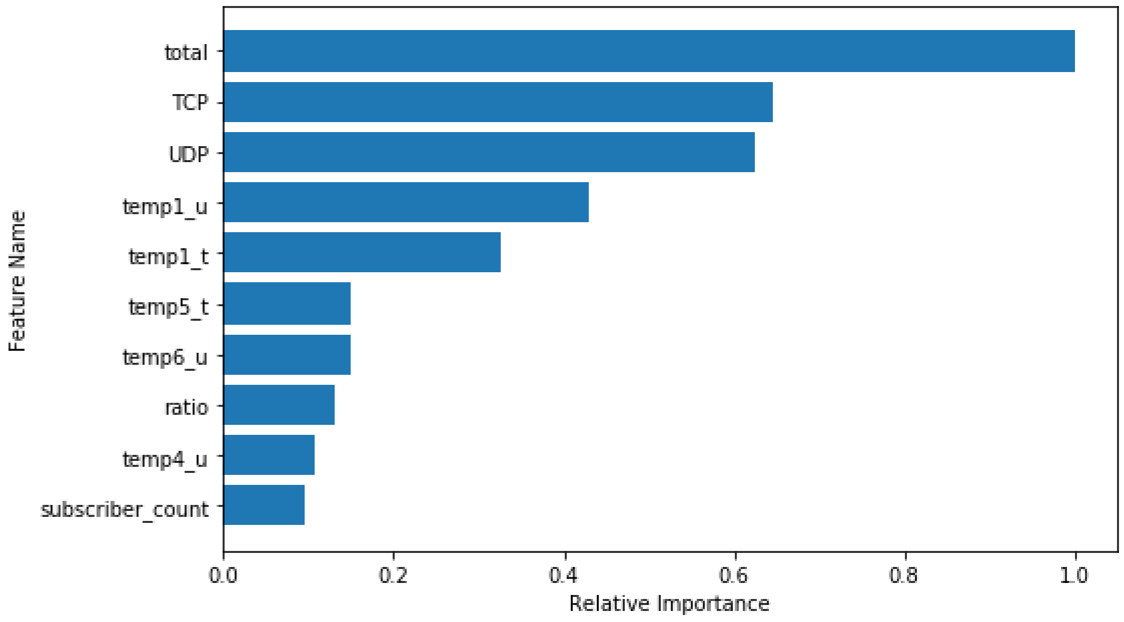
RFE works by recursively removing attributes and building a model on attributes that remain. It uses model accuracy to identify which attributes (and combinations of attributes) contribute the most to predicting the target attribute.

The 4 most important features which this model gives with Logistic Regression Model are:
1) subscriber_count **2) TCP**
3) UDP **4) total_bytes**

Feature importance using decision tree

When we train a classifier such as a decision tree, we evaluate each attribute to create splits; we can use this measure as a feature selector. We have used Random forests, which are among the most popular machine learning methods thanks to their relatively good accuracy, robustness, and ease of use. They also provide two straightforward methods for feature selection—mean decrease impurity and mean decrease accuracy.

RandomForest gives the following relative feature importance:



Modelling

When you look at machine learning algorithms, there is no one solution or one approach that fits all. Our problem statement involved supervised learning and was a multi-classification type problem. Thus, by trial and error, we tried to find the model that gave us the best accuracy.

Baseline Models

1. Application and Interpretation-

- **Logistic Regression -** Starting with the fundamentals we applied logistic regression on our training and validation dataset. In its basic form, it uses a logistic (sigmoid) function to model a binary dependent variable. As the problem at hand was a multi-classification type, we used the one versus rest approach in our solution, but being a basic algorithm, we couldn't get the optimum accuracy and we moved on to other models.

MCC score after k-fold cross-validation	**0.7206**

- **Random Forest -** Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Random decision forests correct for decision trees' habit of overfitting to their training set, thus we expected better accuracy. Although it did give better results as compared to decision trees, further models outdid it in terms of accuracy and Matthews score.

MCC score after k-fold cross-validation	**0.6995**

- **XGBoost -** XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way. It proved to be the best algorithm and gave us maximum accuracy.

MCC score after k-fold cross-validation	**0.7396**

2. Tuning –

We tuned each model using GridSearchCV to get the parameters corresponding to optimum accuracy from them.
GridSearchCV - Grid Search Cross validation was applied on XGBoost and Multi-Layer Perceptron models and the best estimating parameters that increased prediction accuracy on the validation dataset were obtained.

After parameter tuning -

Model	MCC before tuning	MCC after tuning
XGBoost	**0.7396**	**0.7393**

3. Domain Adaptation to Advanced Models-

After fitting some basic baseline models we shifted our domain to advanced models like SVM, Multi-Level Perceptrons (MLPs) and Artificial Neural Networks (ANNs) to enable feature based learning that made our predictions more robust and increased our accuracy.

- **SVM -** A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. We used SVM with “rbf” kernel to learn nonlinear relationships between the features.

MCC score after k-fold cross-validation	**0.7120**

- **MLP -** A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of, at least, three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

MCC score after k-fold cross-validation	**0.7203**

Model Selection

After applying several basic as well as advanced models on our dataset we achieved highest accuracy as well as MCC via XGBoost. After hyperparameter tuning using GridSearchCV, we trained it again using stratified K-Fold Cross Validation to achieve optimum bias-variance trade-off.

Reducing Overfitting

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the model ability to generalize.

We have followed the following strategies to overcome overfitting:

1. **Dropping features-** Keeping irrelevant attributes in dataset can result in overfitting. It is important to remove redundant and irrelevant attributes from dataset before evaluating algorithms. We have dropped many unimportant features like 4G_rat, par_year, par_month, beam_direction, cell_range, tilt, and ran_vendor.
2. **K-fold cross-validation-** Cross-validation is a powerful preventative measure against overfitting. The idea is clever: Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model. In standard k-fold cross-validation, we partition the data into k subsets, called folds. Then, we iteratively train the algorithm on

k-1 folds while using the remaining fold as the test set (called the “holdout fold”). In our code, we have used stratified k-fold cross-validation keeping the value of k=10.

3. **Regularization-** Simply put, it introduces a cost term for bringing in more features with the objective function. Hence, it tries to push the coefficients for many variables to zero and reduce cost term.

   XGBoost has an option to penalize complex models through both **L1 and L2 regularization**. We have used **reg_alpha parameter** for L1 regularization and **reg_lambda parameter** for L2 regularization.

Conclusion

From the above analysis, we can infer that a heavy feature engineering was required to predict the congestion type. We started by analyzing and figuring out the important features and dropping the insignificant ones. We did feature engineering on the selected features and got new features like TCP, UDP, UDP/TCP and temp (grouped by similarity) variables. We then showed that these features are important by running three feature selection algorithms.

Finally, we modelled our features on various Supervised Classification models and found out that XGBoost gives us the best MCC score of 0.739. We have ensured that the model does not overfit the data by using k-fold cross-validation and regularization parameters, alpha and lambda, on the XGBoost model.