Quarter 2 Checkpoint Report

Authors: Laura Diao, Ben Sam, and Jenna Yang

Introduction

Network degradation may occur in two forms: packet loss and latency. Packet loss occurs when one or more data packets transmitted across a computer network fail to reach their destination. Latency can be defined as a measure of delay for data to transmit across a network. For internet users, high rates of packet loss and significant latency can manifest in jitter or lag, indicators of overall poor network performance as perceived by the end user. Thus, when these two issues do arise, it would be beneficial for an internet service provider (ISP) such as Viasat to know exactly when the user is experiencing problems in real time. In real world scenarios, situations or environments such as poor port quality, overloaded ports, network congestion and more can impact overall network performance.

Abstract

In order to detect issues in network transmission data, we built a real-time-capable anomaly detection system. This system includes both alerting and monitoring features, which would enable Viasat to properly monitor user network performance and quality in real time. Moreover, detection in real-time would allow Viasat to handle issues more promptly and increase customer satisfaction. The system utilizes simulated network traffic data to train a model that predicts the packet loss rate as well as the latency of an internet connection, and uses these as well as certain features derived from the data to determine whether an adverse event is occurring.

Timeline

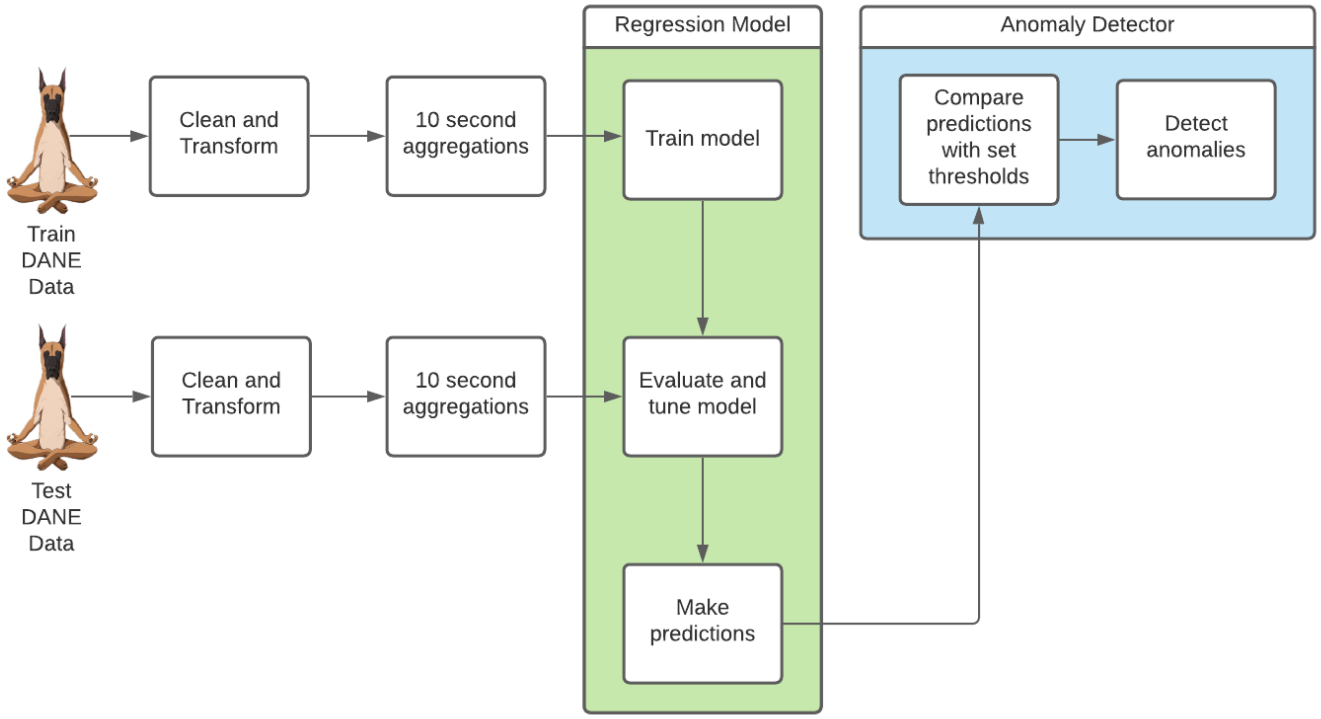
- 1/17 Regression (Q1) model training data generating process established
- 1/19 Adding "Switch Timestamp" feature to DANE, Early Regression model feature EDA
- 1/25 Adding "Switch Time set" DANE feature, Regression pipeline streamlining
- 2/2 Regression model selection, Anomaly classification data generating process established
- **2/9 Classification Model Feature EDA, Model Tuning, Webpage planning and implementation**
- **2/13 Webpage skeleton implemented for checkpoint, Clean up repository**
- **2/16 Oral presentation/Final deliverables planning**
- **2/20 Oral Presentation dry run for checkpoint**
- **2/23 Final deliverable product check in**

- **3/2 Last minute changes**
- **3/9 Project due date**

Limitations

Methods

Data Pipeline



Our project runs on the following data pipeline as seen above. Raw data from DANE (either for train or test purposes) is cleaned and transformed into 10 second aggregations which are then fed into a regression model that makes predictions of loss and latency. With predictions being made on a stream of data in real time, our anomaly detector would ingest these predictions and identify anomalous behavior based on established thresholds of change.

Feature Selection

Features on Seen Data

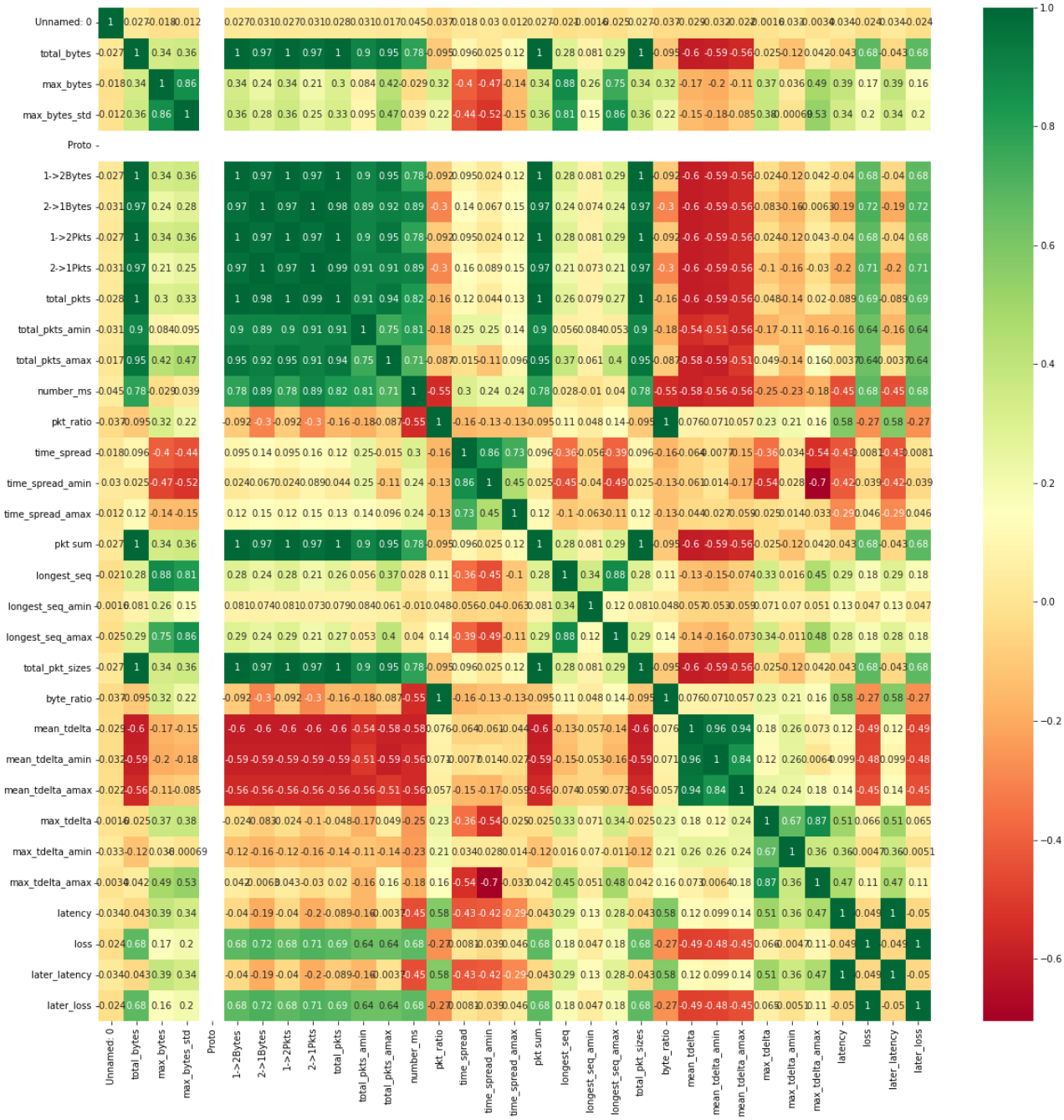

Unnamed: 0
total_bytes
max_bytes
max_bytes_std
Proto
1->2Bytes
2->1Bytes
1->2Pkts
2->1Pkts
total_pkts
total_pkts_amin
total_pkts_amax
number_ms
pkt_ratio
time_spread
time_spread_amin
time_spread_amax
pkt sum
longest_seq
longest_seq_amin
longest_seq_amax
total_pkt_sizes
byte_ratio
mean_tdelta
mean_tdelta_amin
mean_tdelta_amax
max_tdelta
max_tdelta_amin
max_tdelta_amax
latency
loss
later_latency
later_loss
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

Figure 1.1

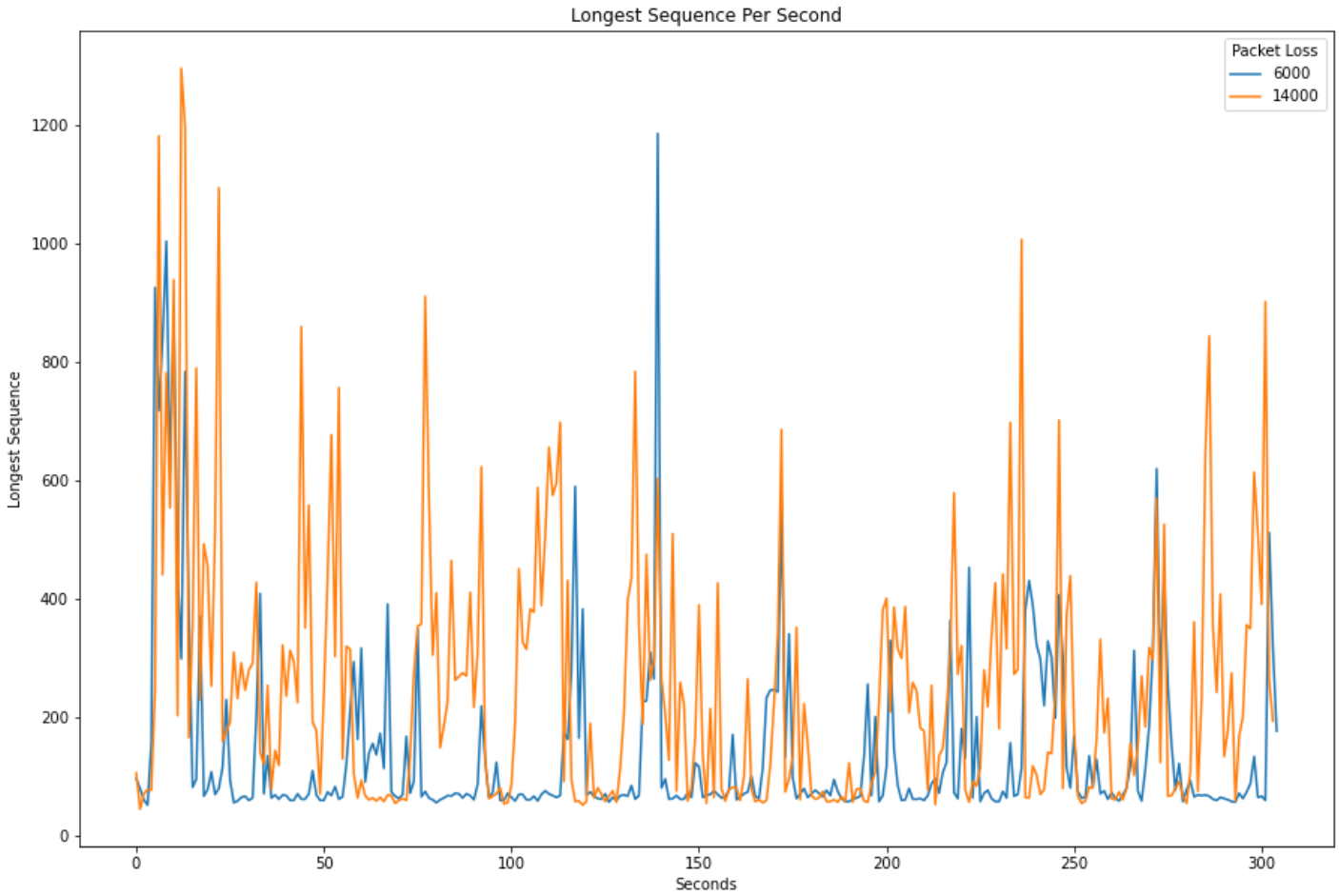

Longest Sequence Per Second
Packet Loss
6000
14000
Longest Sequence
Seconds
0
200
400
600
800
1000
1200
0
50
100
150
200
250
300

Figure 1.2

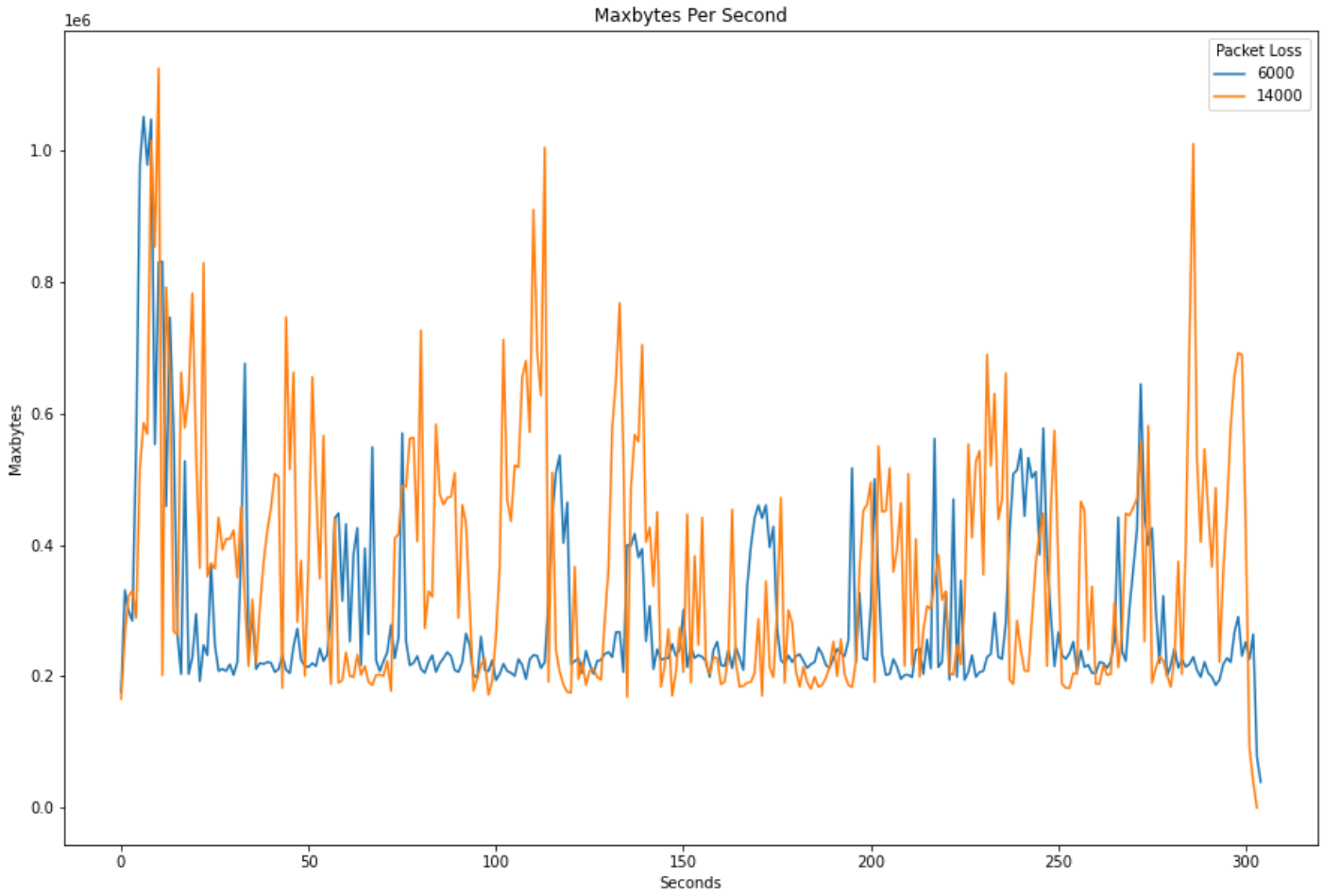

Maxbytes Per Second
1e6
Packet Loss
6000
14000
Maxbytes
Seconds
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250
300

Figure 1.3

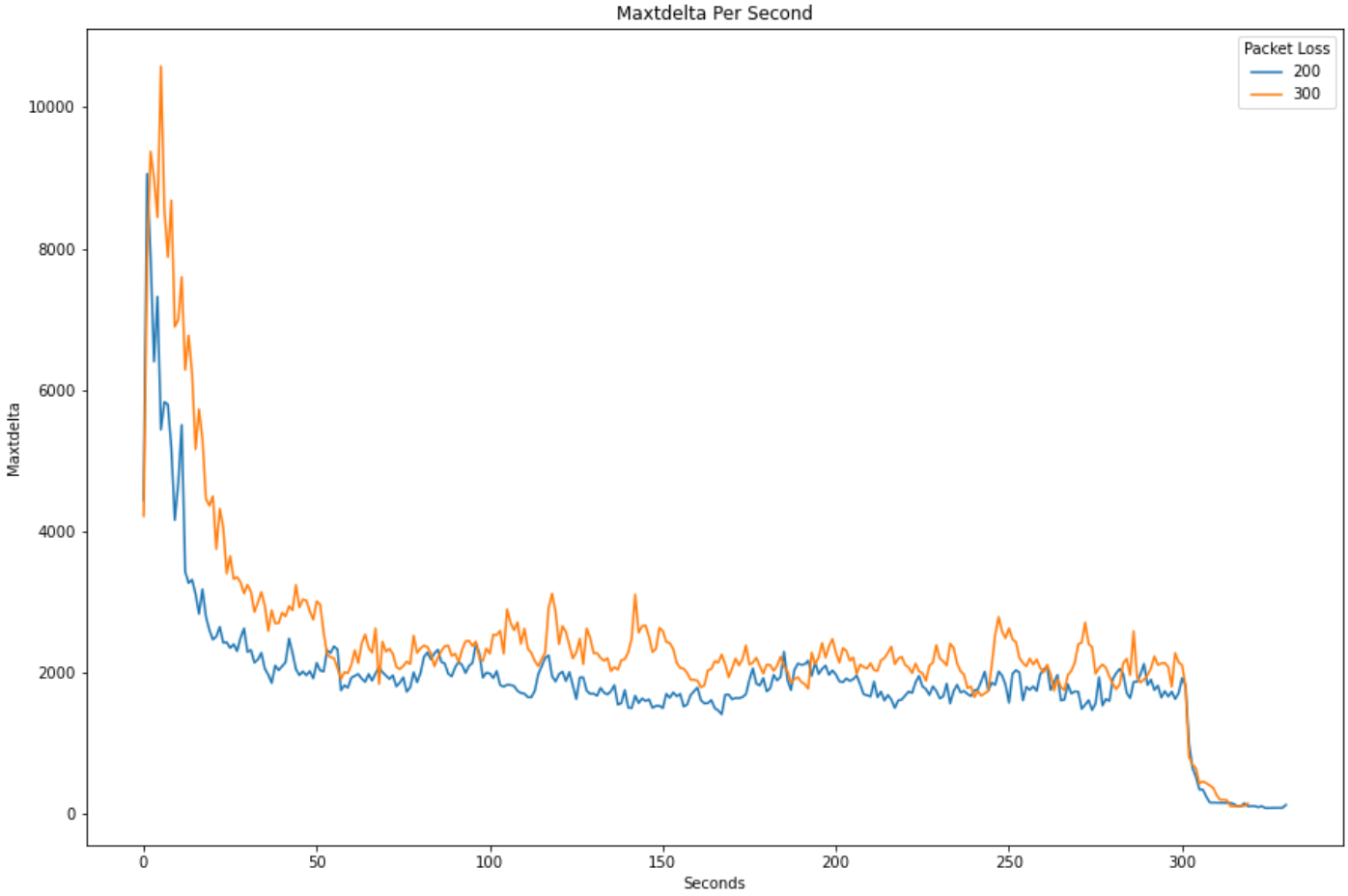

Maxtdelta Per Second
Packet Loss
200
300
10000
8000
6000
4000
2000
0
Maxtdelta
0
50
100
150
200
250
300
Seconds

Figure 1.4

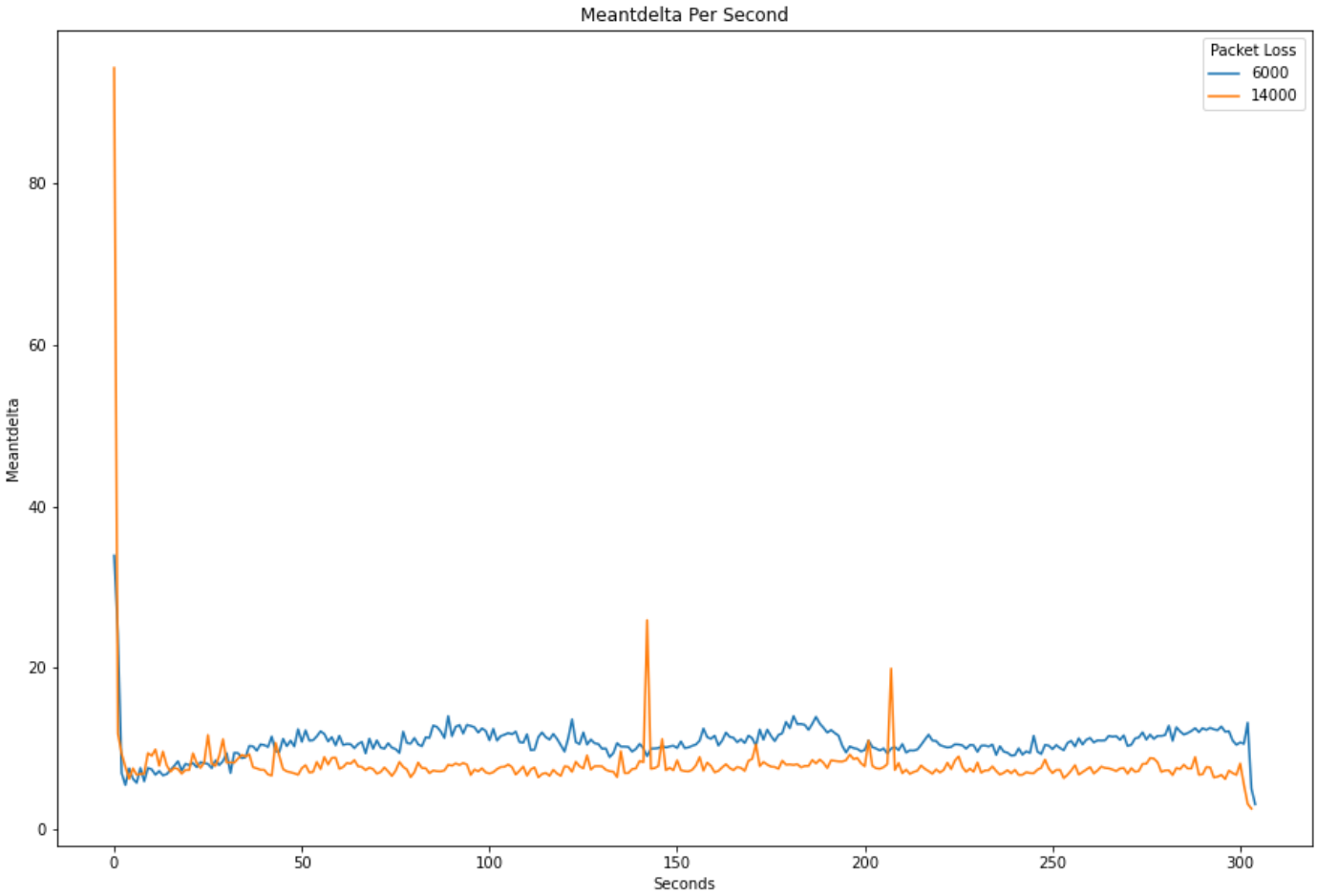

Meantdelta Per Second
Packet Loss
6000
14000
Meantdelta
Seconds
0
20
40
60
80
0
50
100
150
200
250
300

Figure 1.5

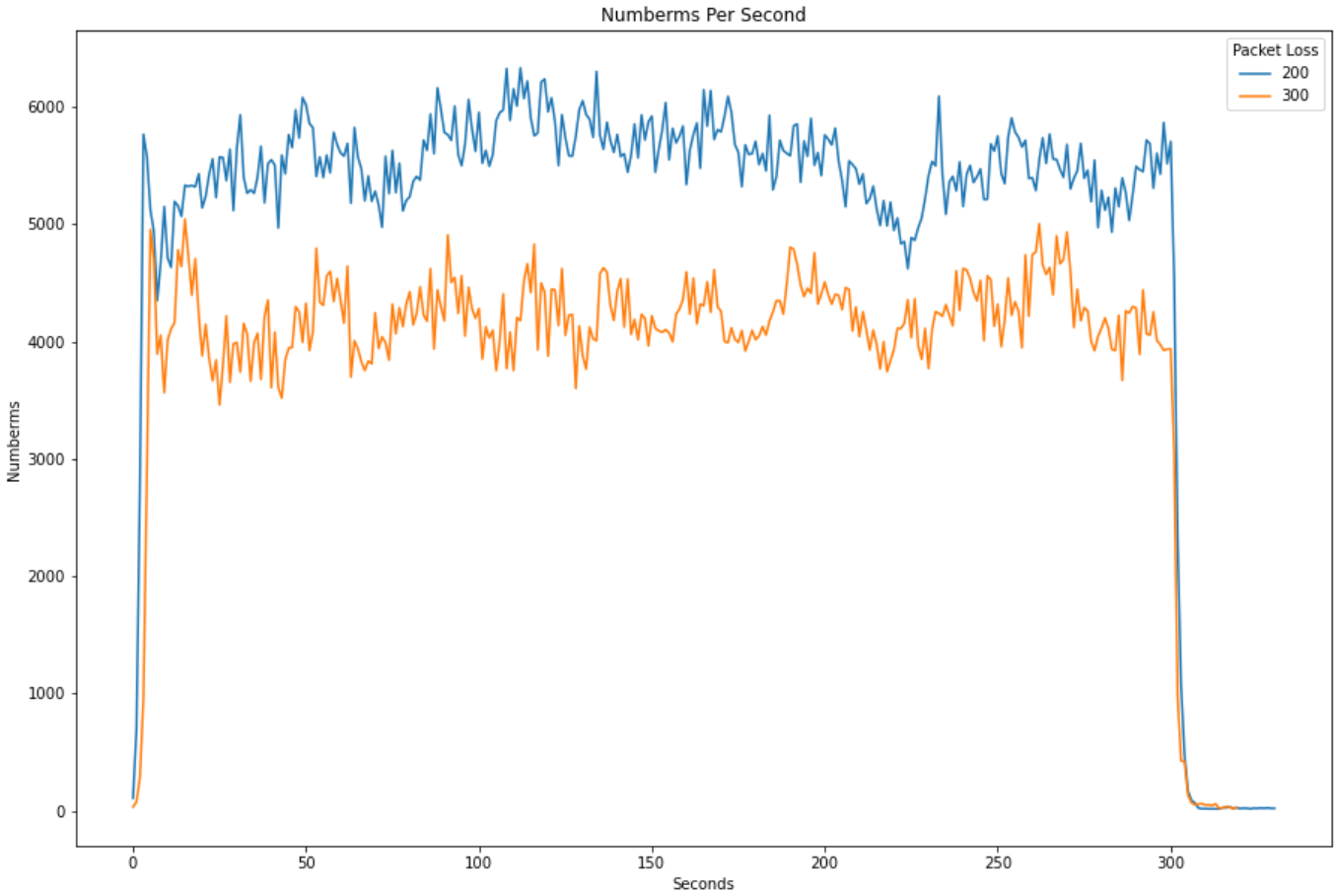

Numberms Per Second
Packet Loss
200
300
Numberms
Seconds
0
1000
2000
3000
4000
5000
6000
0
50
100
150
200
250
300

Figure 1.6

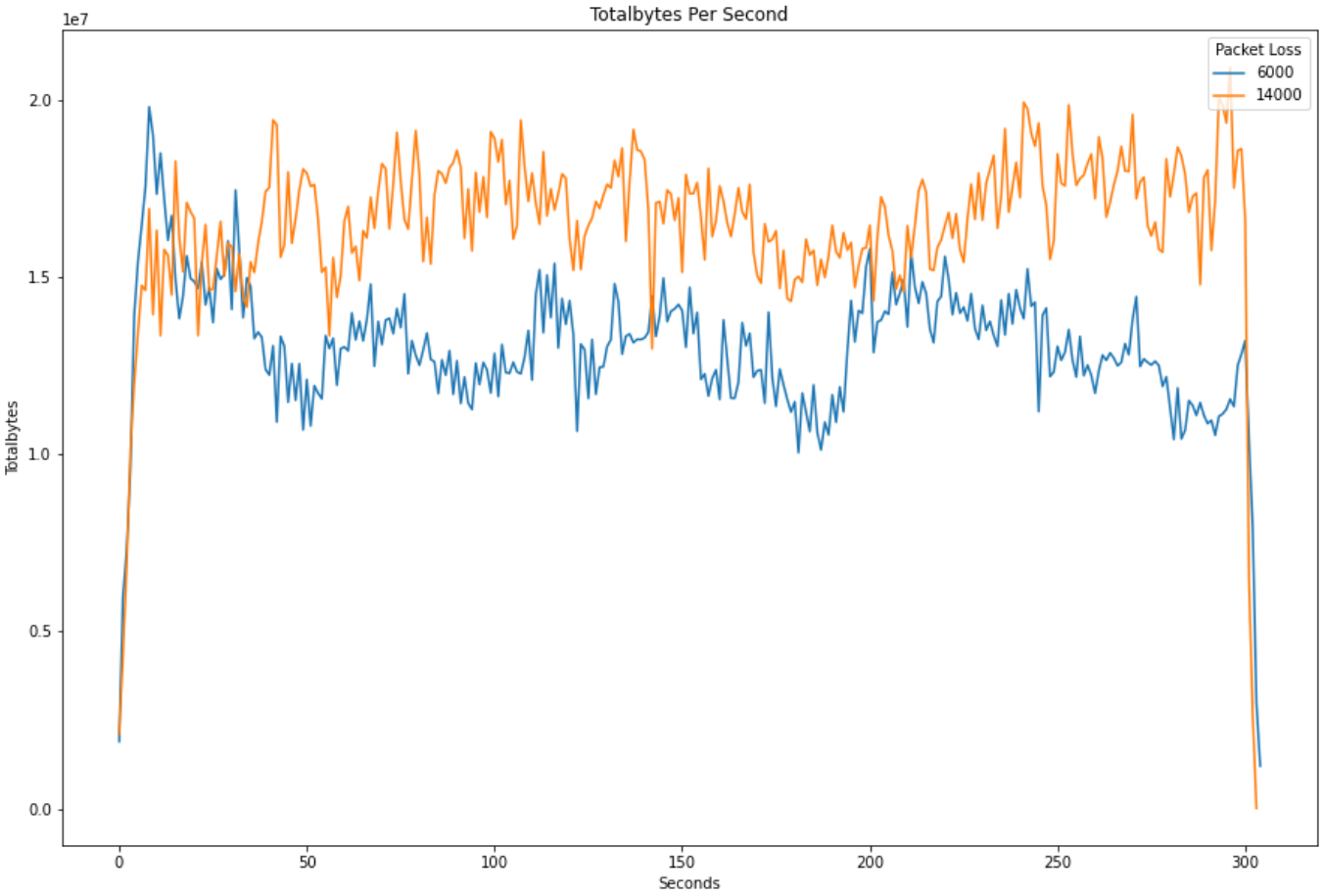

Totalbytes Per Second
1e7
Packet Loss
6000
14000
2.0
1.5
1.0
0.5
0.0
Totalbytes
0
50
100
150
200
250
300
Seconds

Figure 1.7

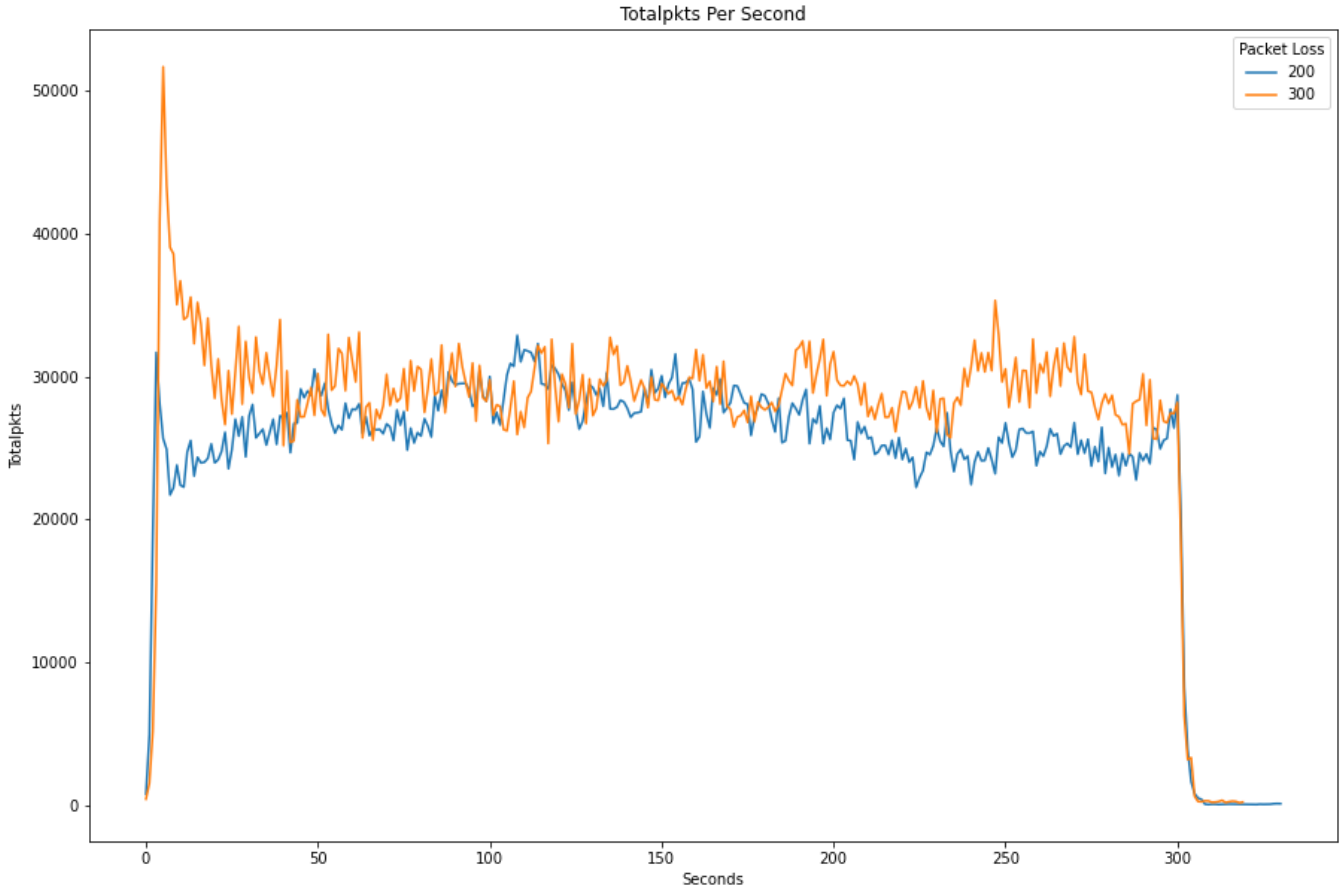


Figure 1.8

Features on Unseen Test Data

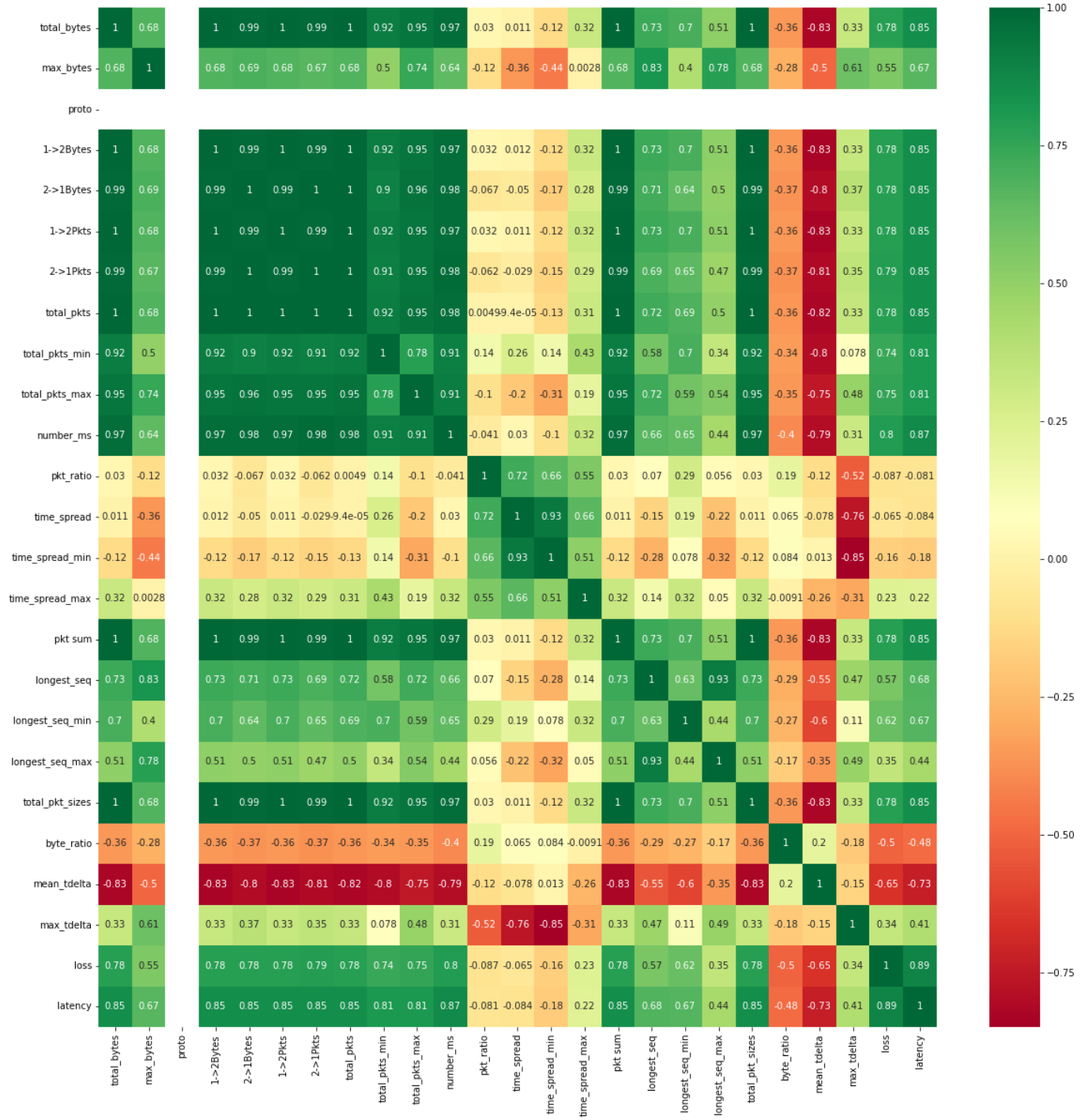

total_bytes 1 0.68 1 0.99 1 0.99 1 0.92 0.95 0.97 0.03 0.011 -0.12 0.32 1 0.73 0.7 0.51 1 -0.36 -0.83 0.33 0.78 0.85
max_bytes 0.68 1 0.68 0.69 0.68 0.67 0.68 0.5 0.74 0.64 -0.12 -0.36 -0.44 0.0028 0.68 0.83 0.4 0.78 0.68 -0.28 -0.5 0.61 0.55 0.67
proto
1->2Bytes 1 0.68 1 0.99 1 0.99 1 0.92 0.95 0.97 0.032 0.012 -0.12 0.32 1 0.73 0.7 0.51 1 -0.36 -0.83 0.33 0.78 0.85
2->1Bytes 0.99 0.69 0.99 1 0.99 1 1 0.9 0.96 0.98 -0.067 -0.05 -0.17 0.28 0.99 0.71 0.64 0.5 0.99 -0.37 -0.8 0.37 0.78 0.85
1->2Pkts 1 0.68 1 0.99 1 0.99 1 0.92 0.95 0.97 0.032 0.011 -0.12 0.32 1 0.73 0.7 0.51 1 -0.36 -0.83 0.33 0.78 0.85
2->1Pkts 0.99 0.67 0.99 1 0.99 1 1 0.91 0.95 0.98 -0.062 -0.029 -0.15 0.29 0.99 0.69 0.65 0.47 0.99 -0.37 -0.81 0.35 0.79 0.85
total_pkts 1 0.68 1 1 1 1 1 0.92 0.95 0.98 0.00499 4e-05 -0.13 0.31 1 0.72 0.69 0.5 1 -0.36 -0.82 0.33 0.78 0.85
total_pkts_min 0.92 0.5 0.92 0.9 0.92 0.91 0.92 1 0.78 0.91 0.14 0.26 0.14 0.43 0.92 0.58 0.7 0.34 0.92 -0.34 -0.8 0.078 0.74 0.81
total_pkts_max 0.95 0.74 0.95 0.96 0.95 0.95 0.95 0.78 1 0.91 -0.1 -0.2 -0.31 0.19 0.95 0.72 0.59 0.54 0.95 -0.35 -0.75 0.48 0.75 0.81
number_ms 0.97 0.64 0.97 0.98 0.97 0.98 0.98 0.91 0.91 1 -0.041 0.03 -0.1 0.32 0.97 0.66 0.65 0.44 0.97 -0.4 -0.79 0.31 0.8 0.87
pkt_ratio 0.03 -0.12 0.032 -0.067 0.032 -0.062 0.0049 0.14 -0.1 -0.041 1 0.72 0.66 0.55 0.03 0.07 0.29 0.056 0.03 0.19 -0.12 -0.52 -0.087 -0.081
time_spread 0.011 -0.36 0.012 -0.05 0.011 -0.029 -9.4e-05 0.26 -0.2 0.03 0.72 1 0.93 0.66 0.011 -0.15 0.19 -0.22 0.011 0.065 -0.078 -0.76 -0.065 -0.084
time_spread_min -0.12 -0.44 -0.12 -0.17 -0.12 -0.15 -0.13 0.14 -0.31 -0.1 0.66 0.93 1 0.51 -0.12 -0.28 0.078 -0.32 -0.12 0.084 0.013 -0.85 -0.16 -0.18
time_spread_max 0.32 0.0028 0.32 0.28 0.32 0.29 0.31 0.43 0.19 0.32 0.55 0.66 0.51 1 0.32 0.14 0.32 0.05 0.32 -0.0091 -0.26 -0.31 0.23 0.22
pkt sum 1 0.68 1 0.99 1 0.99 1 0.92 0.95 0.97 0.03 0.011 -0.12 0.32 1 0.73 0.7 0.51 1 -0.36 -0.83 0.33 0.78 0.85
longest_seq 0.73 0.83 0.73 0.71 0.73 0.69 0.72 0.58 0.72 0.66 0.07 -0.15 -0.28 0.14 0.73 1 0.63 0.93 0.73 -0.29 -0.55 0.47 0.57 0.68
longest_seq_min 0.7 0.4 0.7 0.64 0.7 0.65 0.69 0.7 0.59 0.65 0.29 0.19 0.078 0.32 0.7 0.63 1 0.44 0.7 -0.27 -0.6 0.11 0.62 0.67
longest_seq_max 0.51 0.78 0.51 0.5 0.51 0.47 0.5 0.34 0.54 0.44 0.056 -0.22 -0.32 0.05 0.51 0.93 0.44 1 0.51 -0.17 -0.35 0.49 0.35 0.44
total_pkt_sizes 1 0.68 1 0.99 1 0.99 1 0.92 0.95 0.97 0.03 0.011 -0.12 0.32 1 0.73 0.7 0.51 1 -0.36 -0.83 0.33 0.78 0.85
byte_ratio -0.36 -0.28 -0.36 -0.37 -0.36 -0.37 -0.36 -0.34 -0.35 -0.4 0.19 0.065 0.084 -0.0091 -0.36 -0.29 -0.27 -0.17 -0.36 1 0.2 -0.18 -0.5 -0.48
mean_tdelta -0.83 -0.5 -0.83 -0.8 -0.83 -0.81 -0.82 -0.8 -0.75 -0.79 -0.12 -0.078 0.013 -0.26 -0.83 -0.55 -0.6 -0.35 -0.83 0.2 1 -0.15 -0.65 -0.73
max_tdelta 0.33 0.61 0.33 0.37 0.33 0.35 0.33 0.078 0.48 0.31 -0.52 -0.76 -0.85 -0.31 0.33 0.47 0.11 0.49 0.33 -0.18 -0.15 1 0.34 0.41
loss 0.78 0.55 0.78 0.78 0.78 0.79 0.78 0.74 0.75 0.8 -0.087 -0.065 -0.16 0.23 0.78 0.57 0.62 0.35 0.78 -0.5 -0.65 0.34 1 0.89
latency 0.85 0.67 0.85 0.85 0.85 0.85 0.85 0.81 0.81 0.87 -0.081 -0.084 -0.18 0.22 0.85 0.68 0.67 0.44 0.85 -0.48 -0.73 0.41 0.89 1
total_bytes max_bytes proto 1->2Bytes 2->1Bytes 1->2Pkts 2->1Pkts total_pkts total_pkts_min total_pkts_max number_ms pkt_ratio time_spread time_spread_min time_spread_max pkt sum longest_seq longest_seq_min longest_seq_max total_pkt_sizes byte_ratio mean_tdelta max_tdelta loss latency
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Figure 2.1

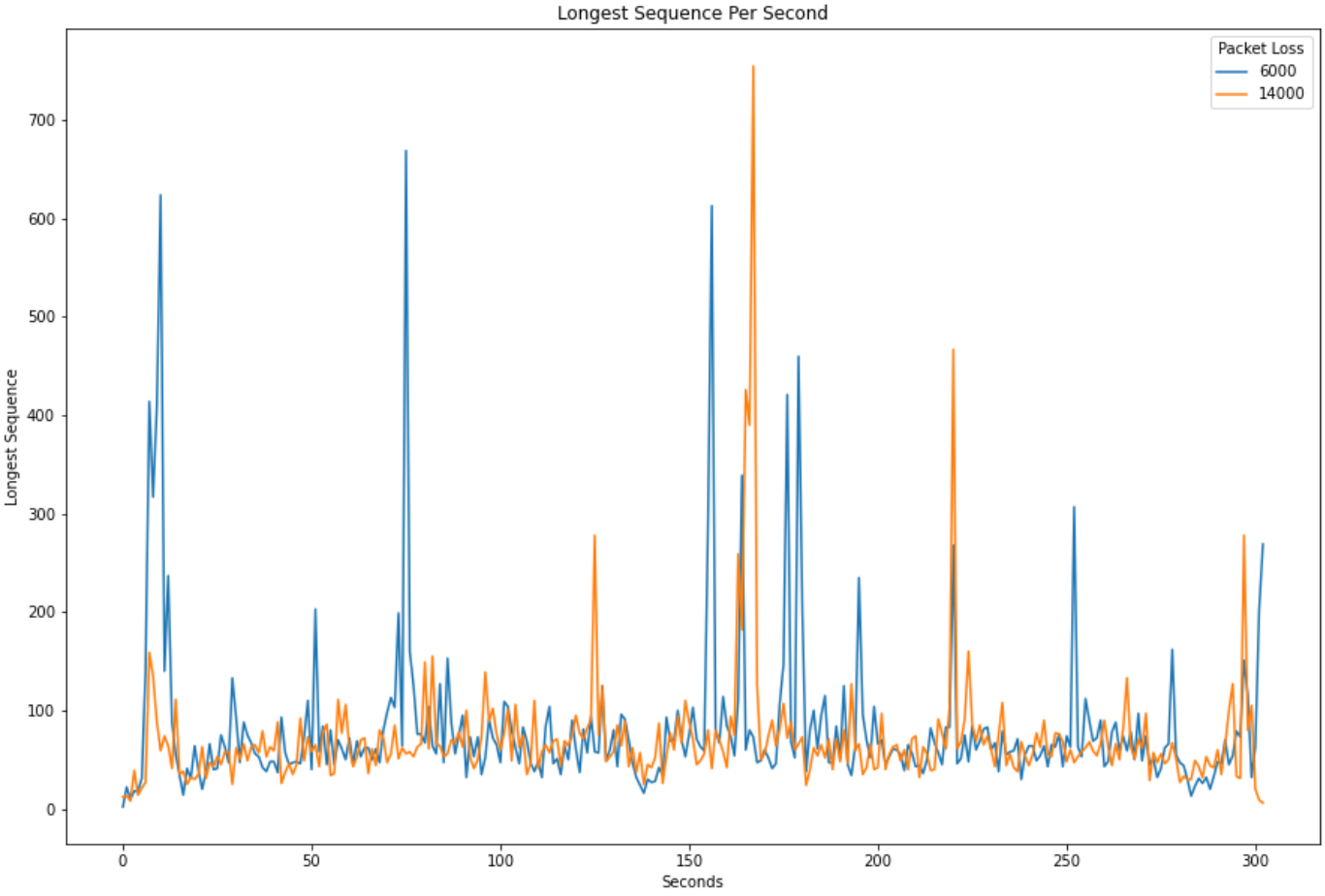

Longest Sequence Per Second
Packet Loss
6000
14000
Longest Sequence
Seconds
0
100
200
300
400
500
600
700
0
50
100
150
200
250
300

Figure 2.2

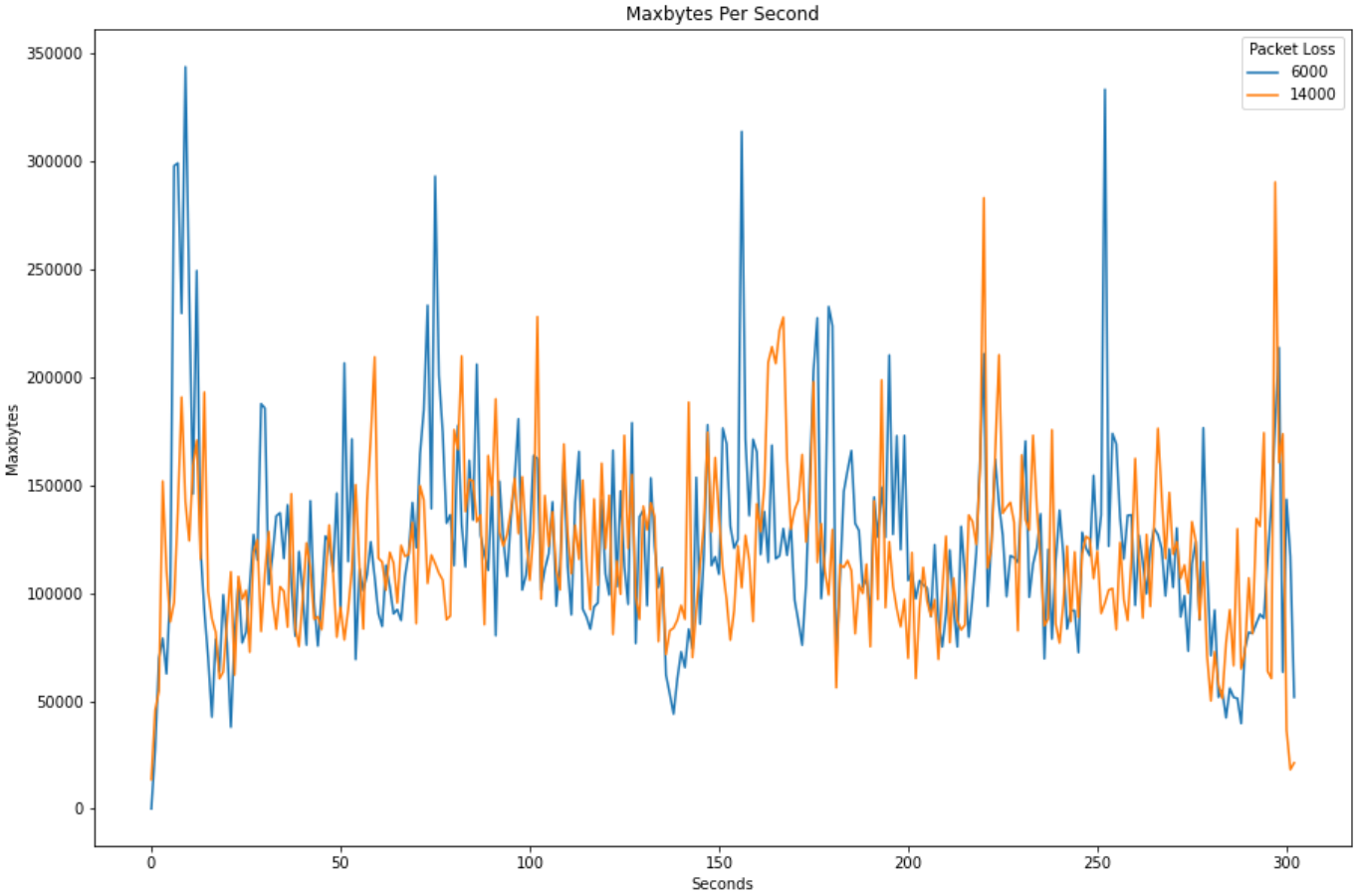

Maxbytes Per Second
Packet Loss
6000
14000
Maxbytes
Seconds
350000
300000
250000
200000
150000
100000
50000
0
0
50
100
150
200
250
300

Figure 2.3

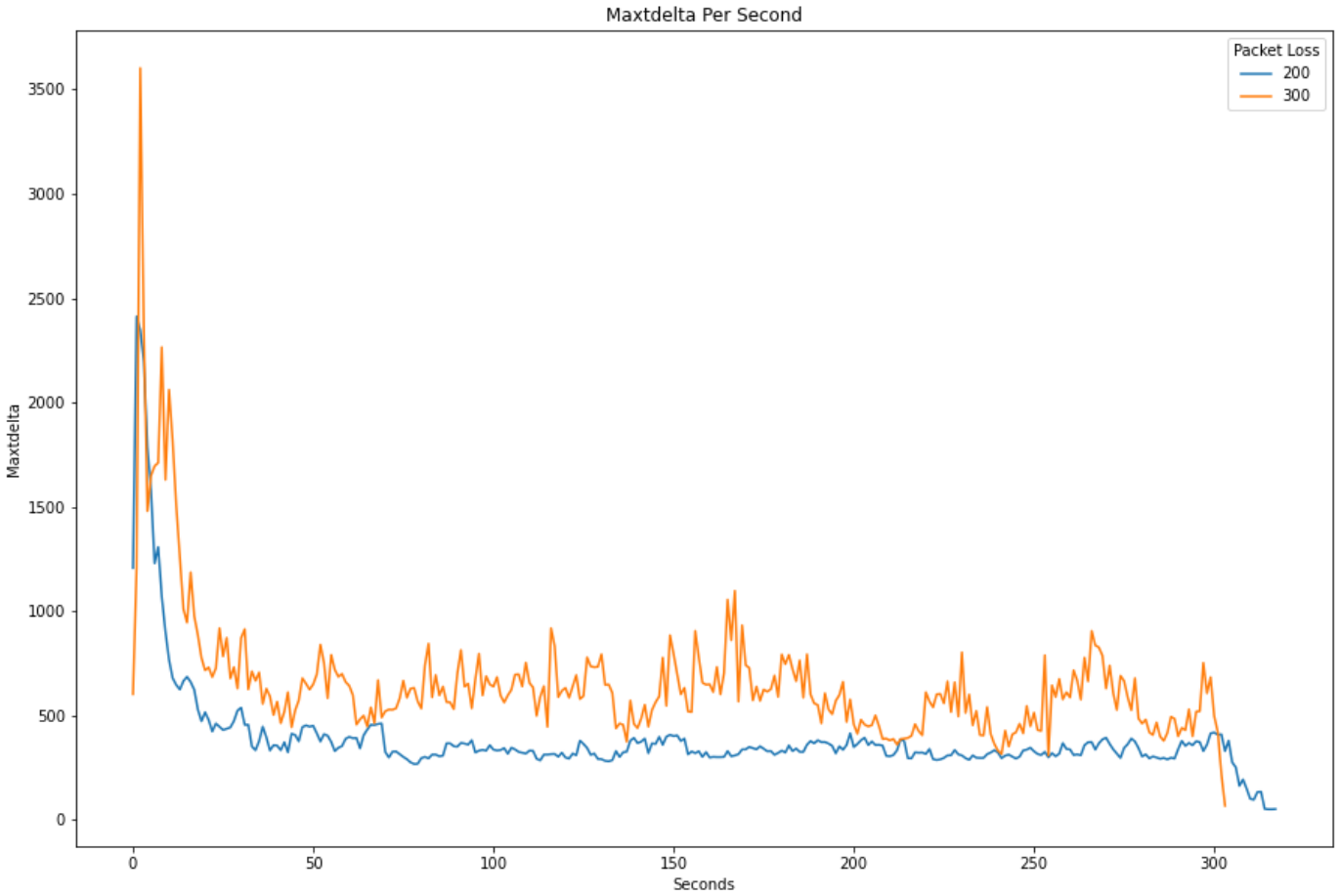

Maxtdelta Per Second
Packet Loss
200
300
Maxtdelta
Seconds
0
500
1000
1500
2000
2500
3000
3500
0
50
100
150
200
250
300

Figure 2.4

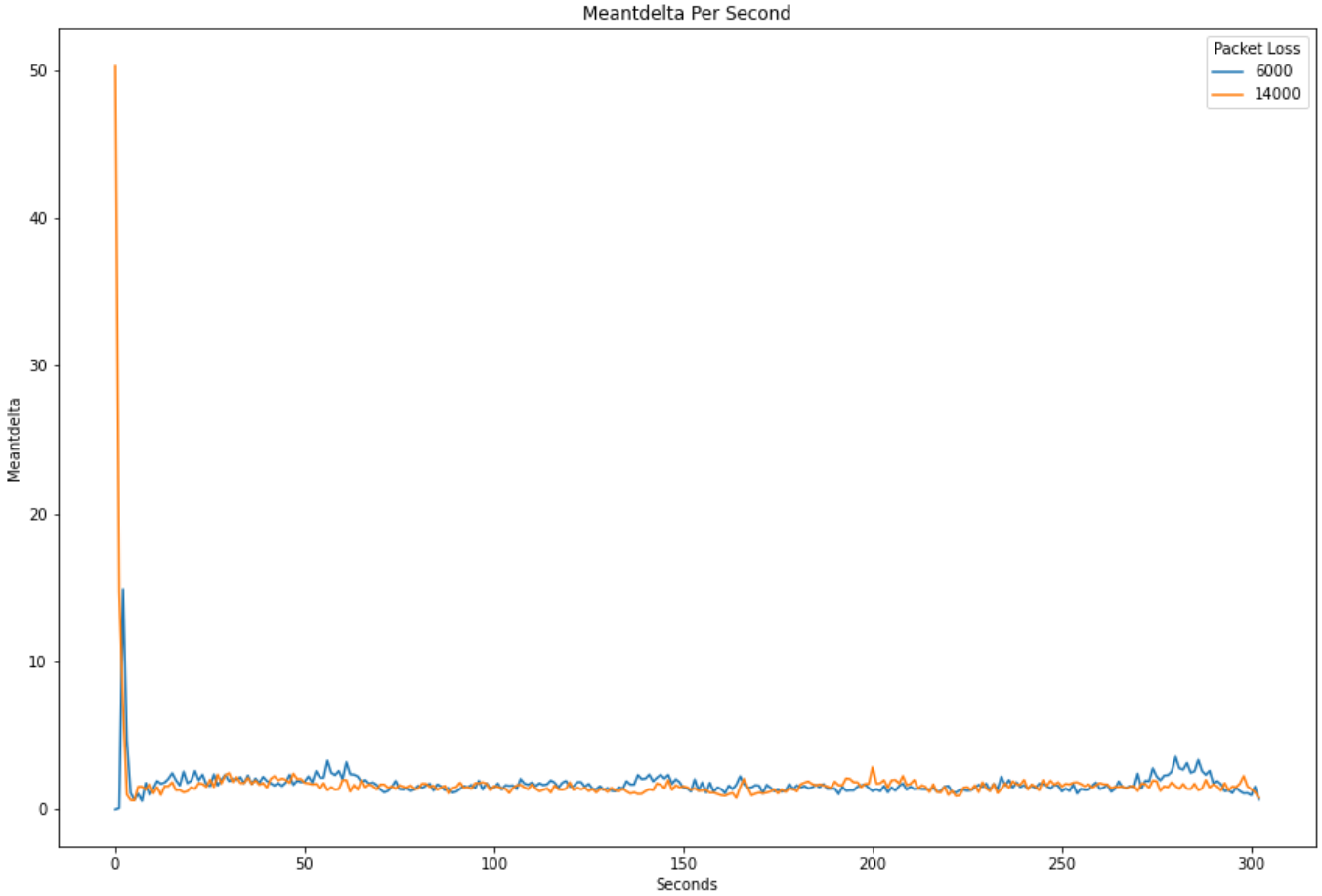

Meantdelta Per Second
Packet Loss
6000
14000
Meantdelta
Seconds
0
10
20
30
40
50
0
50
100
150
200
250
300

Figure 2.5

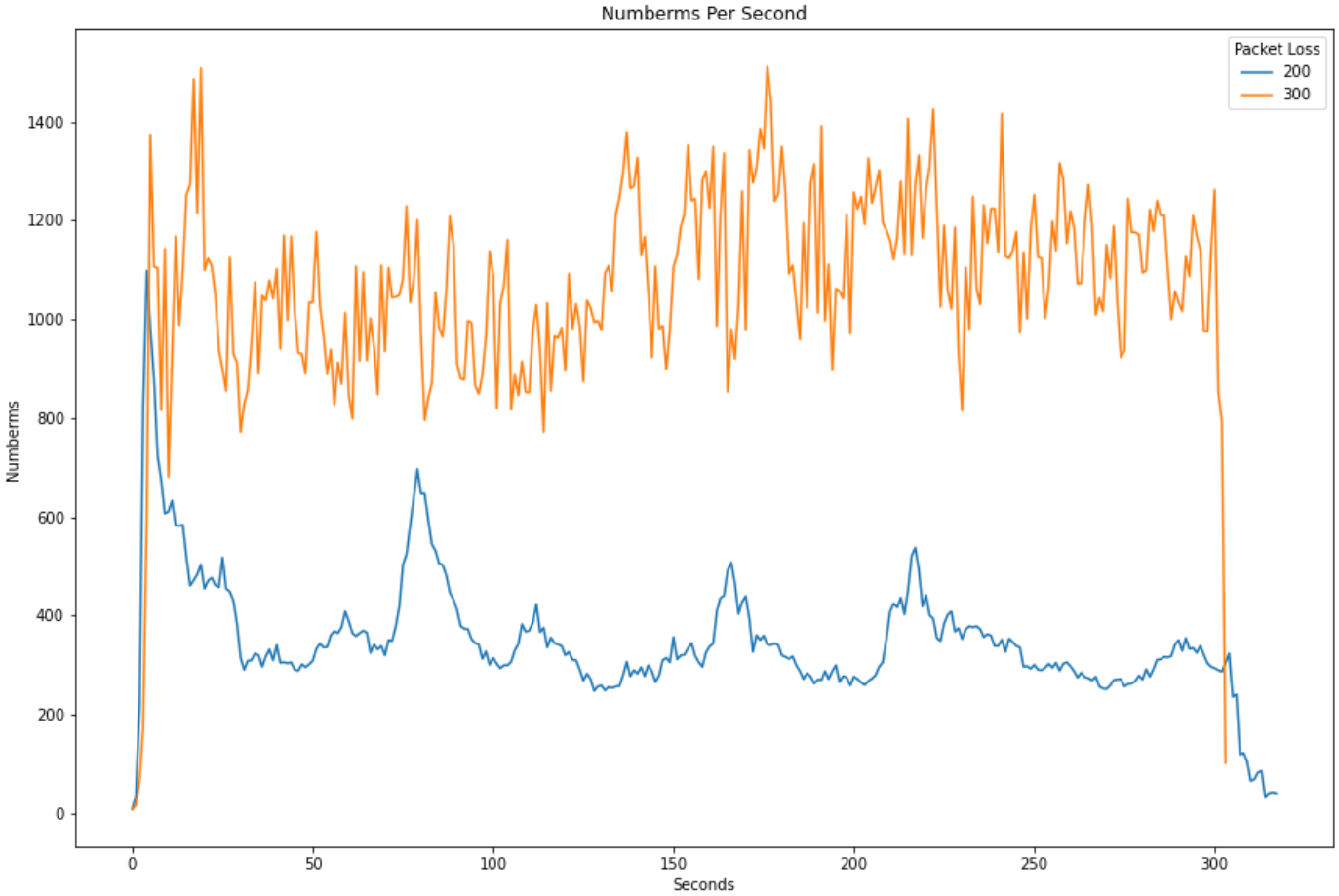

Numberms Per Second
Packet Loss
200
300
Numberms
Seconds
0
200
400
600
800
1000
1200
1400
0
50
100
150
200
250
300

Figure 2.6

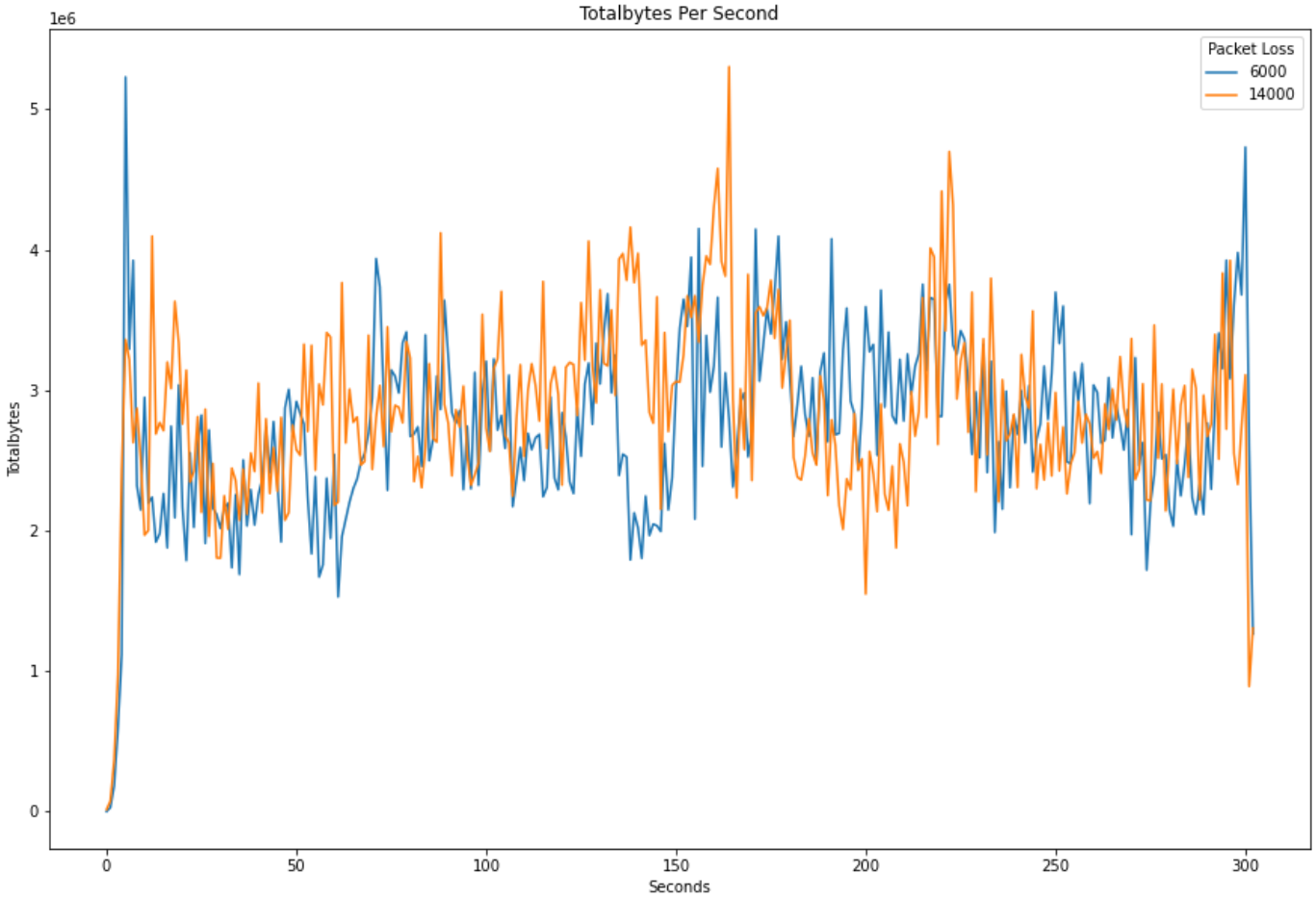

Totalbytes Per Second
1e6
Packet Loss
6000
14000
Totalbytes
Seconds
0
1
2
3
4
5
0
50
100
150
200
250
300

Figure 2.7

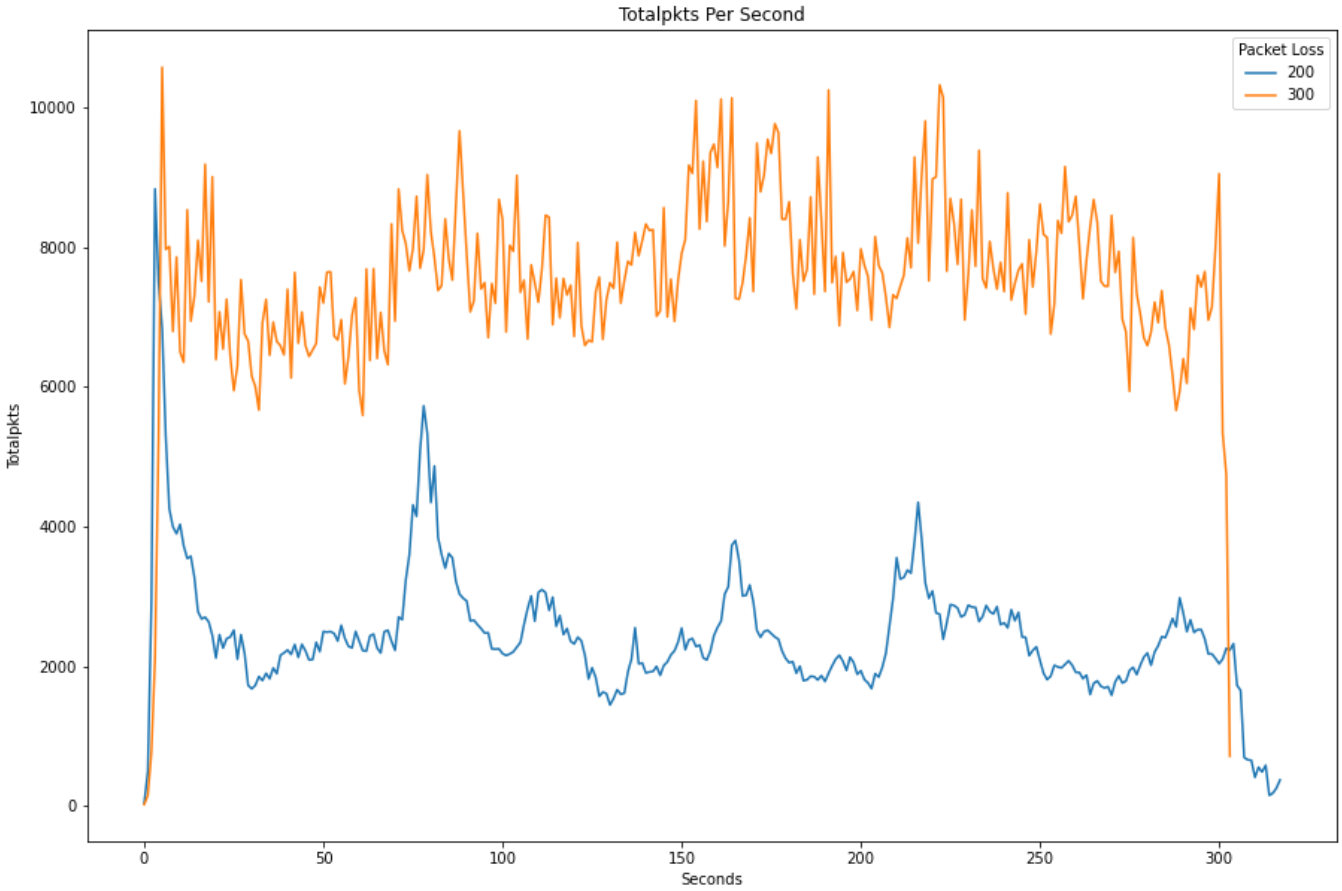


Figure 2.8

After training our models on all the features, we calculated the feature importances and found the following features to be the most effective for our models:

Loss

- total_pkts
- total_pkts_sizes
- 2→1Bytes
- number_ms
- mean_tdelta
- max_tdelta

Latency

- byte_ratio
- pkt_ratio
- time_spread
- total_bytes
- 2→1Pkts

The model for loss used data features that generally encoded information about total network throughput, such as the number and size of packets and directional bitrate. It also used features indicating the rate at which packets came in, such as the number of packets per second as well as the time difference, or time delta, between packet arrival times.

The model for latency utilized features that captured information about the cross-traffic behavior of the connection such as the ratio between directional traffic. It also used features that represent the frequency of packets such as the range of their ime stamps (time_spread) and directional packet rate

Modeling and Performance

With the features selected from above, we trained the following models for predicting both loss and latency: Decision Tree (dt), Random Forest (rt), Extra Trees (et), and Gradient Boost (gbc). The performance of the model on both seen and unseen data can be observed in the tables below.

R^2 On Seen Data (Loss)

feat	dt	rf	et	gbc
['mean_tdelta', 'total_pkt_sizes', 'pkt sum', ...	0.720	0.759	0.747	0.769
['mean_tdelta', 'total_pkts', 'total_pkt_sizes...	0.700	0.757	0.744	0.758
['mean_tdelta', 'total_pkt_sizes', 'number_ms'...	0.664	0.752	0.739	0.754
['total_pkts', 'number_ms']	0.661	0.724	0.691	0.668
['total_pkts_max', 'mean_tdelta', 'total_pkt_s...	0.659	0.791	0.797	0.774

Figure 3.1

R^2 On Seen Data (Latency)

	feat2	dt2	rf2	et2	gbc2
0	['byte_ratio']	1.000	1.000	1.000	1.000
7	['byte_ratio', 'pkt_ratio']	1.000	1.000	0.999	1.000
1	['byte_ratio', 'total_bytes']	0.983	0.982	0.978	0.978
3	['2->1Pkts', 'byte_ratio']	0.983	0.981	0.985	0.983
4	['2->1Pkts', 'byte_ratio', 'total_bytes']	0.983	0.981	0.957	0.978

Figure 3.2

R^2 On Unseen Data (Loss)

feat	dt	rf	et	gbc
['total_pkts_max', 'mean_tdelta', 'total_pkt_s...	0.245	0.536	0.519	0.429
['mean_tdelta', 'total_pkts_max', 'pkt sum', '...	0.237	0.520	0.515	0.428
['total_pkts_max', 'mean_tdelta', 'total_pkt_s...	0.236	0.549	0.531	0.450
['mean_tdelta', 'total_pkts_max', '2->1Bytes',...	0.202	0.554	0.555	0.514
['total_pkts_max', '2->1Bytes', 'number_ms', '...	0.199	0.556	0.552	0.506

Figure 3.3

R^2 On Unseen Data (Latency)

feat2	dt2	rf2	et2	gbc2
['time_spread_min']	-2.275	-1.819	-2.132	-1.707
['total_bytes', '2->1Pkts', 'pkt_ratio', 'time...	-4.169	-1.837	-1.546	-2.880
['pkt_ratio', 'total_bytes', 'time_spread_min'...	-4.384	-1.814	-1.416	-2.639
['2->1Pkts', 'pkt_ratio', 'time_spread_min', '...	-4.384	-1.844	-1.462	-2.552
['pkt_ratio', 'time_spread_min', 'max_tdelta']	-4.898	-1.232	-1.308	-1.750

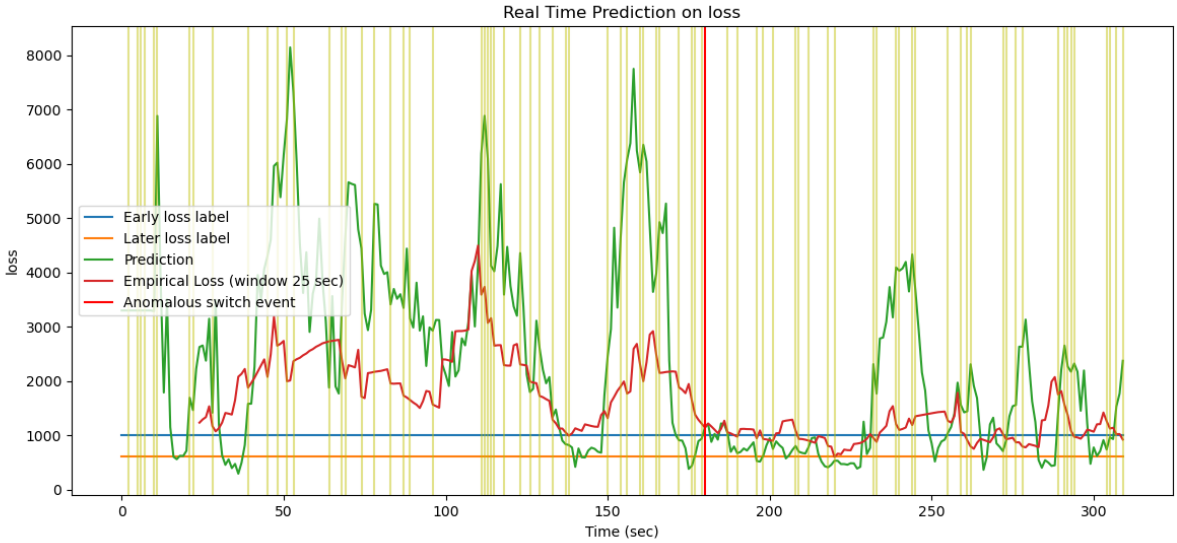
Figure 3.3

We see that for seen data, the models perform relatively well with about a 0.7 R^2 metric across all the models. However, in predicting latency, the models achieve almost perfect R^2 scores, indicating that overfitting is likely occuring (see limitations section).

For unseen data, we observe that, expectedly, the models performed worse than with the seen data, with moderate lower R^2 metrics (from 0.2 to 0.5) for loss and extremely poor metrics for latency. While the performance was not exactly favorable, we hope that grid search and performance tuning will allow us to improve our baseline models.

As for this regression model, we are seeing that it captures significant change well enough to determine anomalous events. Applying the trained loss model gives us a decent prediction in terms of the empirical observed loss within the data:

Loss Model Predictions on Unseen Test Data (loss ratio of 1:600 to 1:1000)



Latency Model Predictions on Unseen Test Data (latency of 200 (no switch))

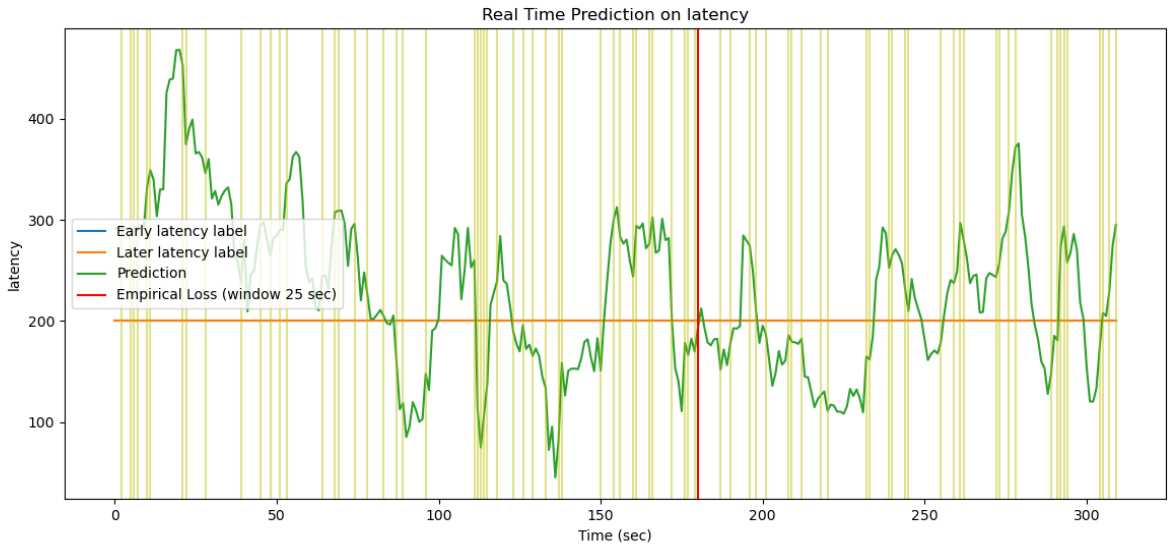


Figure 2.8

The latency model's predictive power is less important, so a wider margin of error will be allowed for its results. (We plan to expand more on model performance here later.)

Throughout our project, we ran into some limitations. Due to the size of the dataset we processed and trained on, this led to long run times and made our workflow less efficient. To handle this, we worked on reducing overcorrelated features by referencing the correlation matrix.

When training and testing various models, we ended up taking out the PCA model due to issues with running different feature combinations on the model testing pipeline. For instance, our combinations of different features had cases of extracting one or more features eg. ['total_bytes'] or the combination ['total_bytes', 'pkt_ratio'] from the transformed data. If our PCA model was constructed to take more than one paramter to test the second feature combination, it would lead to an error in input size when there

one parameter to test the second feature combination, it would lead to an error in input size when there were not two features to train on in the case of testing a feature combination with only one component. To overcome this issue and attempt to utilize pca, we attempted to implement a multi parameter/model grid search, testing the model(s) on different features in another form might resolve the aforementioned issue with testing pca on varying sizes of feature combinations.

Another limitation we ran into involved the previous training metric, MSE. MSE was a good metric to measure the total amount of error from the loss predictions, however it was misleading and hard to compare between models because of the larger loss numbers we worked with eg. 10,000, 2000. Using R2, we can focus on whether the model is performing well on a scale from -1 to 1.

Additionally, due to a major bug in the data processing component of the pipeline, our model was overfitting. the helper function, agg10 which aggregated transformed data features over 10 seconds was intaking the first row and appending that data for every 10 second aggregation. Fixing this error, we saw a significant change in the model performance.

Overall, we learned another lesson to keep code and files organized and reproducible. Going back and documenting how the files are connected as well as how each function interacts with eachother was crucial for our team to understand how the pipeline functioned. This enabled us to detect bugs and components that could be optimized.

Future Work

With a rudimentary pipeline established for ingesting raw DANE data and producing predictions via a regression model, a key task for the upcoming weeks would be to optimize this pipeline by adjusting the ETL code for increased scalability and reproducibility. The goal would be to have a polished pipeline that is robust to file error bugs and different sizes of data so that the entire anomaly detection system may work as efficiently as possible.

Moreover, we plan to build on our baseline models by tuning the performance further and performing grid search to find optimal hyperparameters for our models. The grid search would be integrated into the overall pipeline to output more accurate predictions.

On the data generation side of the project, since our goal is to predict anomalies on a data stream in real-time, we plan to implement or modify DANE such that it would be able to emulate such a process. Currently, DANE is able to generate runs of data that are accessed after completion of the runs. However, to better emulate the desired anomaly detection configurations, we hope to be able to make modifications to DANE that will allow us to access the data real-time in a stream instead of after the fact.

Finally, we plan to implement and integrate the anomaly detection mechanism for the system, which would ingest the predictions from our regression model and identify anomalies in 10 second windows. This would require more fleshing out of the definition and threshold of an anomaly, as well as being able to run our models on a stream of data. Once all the parts are consolidated into one system of anomaly detection, we aim to conduct testing for the entire pipeline and system to ensure it is working properly before publishing.