Project Name: Real Network Modeling

Authors: Jaye Cho, Shasank Bonthala, and Michael Pattik

Introduction

The internet as we know it was introduced all the way back in 1983, but it wasn't until 1994 that a company known as NetMarket first started selling internet services. Since then, the satellite service providing industry has increased exponentially. Every company aims to provide the best network conditions to customers in an industry where quality is so important, and that is specifically where improvements can always be made. Being able to detect network issues and monitor network performance for issues is so important in the real world for positive customer experience and that is the problem we aim to tackle with our project. At the end of the day, the customer makes the business, and increasing customer satisfaction is always a priority in a successful business.

Before we go into specifics about the approach to tackling this problem, it is important to understand a few key technical terms. The two biggest network conditions that we are targeting with this project are packet loss and latency. Packet loss refers to the loss in packets of data when these packets of data don't reach the destination. This will be represented as a ratio in our project. The packet loss ratio is the number of packets lost divided by the total number of packets since the last lost one (1/2000 refers to 1 packet lost out of 2000 packets). Latency on the other hand, refers to delays in the network that are observed from a cause and effect of some physical change in the system. This will be represented in our data as an integer. Being able to successfully predict these could be instrumental for Viasat as this will allow service providers to better monitor network performance and identify issues.

The models (we experimented with linear regression, bayesian ridge regression, support vector regression, and KNN regression) that we built this quarter to predict latency and packet loss rely heavily on a tool called DANE, which is a dataset generation tool that collects deterministic network traffic data under configurable network conditions. Latencies and packet loss ratios are able to be customized prior to running this tool to generate datasets that can be used in analysis and eventually leads to models.

Prior to this quarter's work DANE was a tool that only produced deterministic data. This means it would drop the *n*th packet every time, where *n* is a variable input by the user. The issue with this is that the data that was collected wasn't very representative of the real-world, which causes the model to be unrepresentative as well. However, DANE was updated this quarter and included a randomness component which allowed for a more real-world representative dataset. It was this updated version of DANE that was used in this project. This has been a key component to enhance the credibility of our models.

Methods

The first step in our data preparation was to add the latency and loss values to the dataset. These values were chosen by our team when changing the configurations to run DANE. We simply created two columns in each of the datasets outputted from DANE that indicated the datasets' respective latency and loss ratios. Next, we went through each dataset and eliminated the first 25 seconds of data to reduce bias.

In total, the models include ten features. The user can configure the models by deciding which of these ten features are included. The first feature is the maximum packet size sent in each interaction. As shown in Figure 1.1, the maximum packet size is either at the bandwidth cap, zero, or just above zero. The interactions with a maximum packet size just above zero are likely explained by the "check" packets sent through the network to ensure proper transmission of information. We see these among the lower packet loss ratios but not the higher packet loss ratios. This is not surprising, as the higher packet loss ratio interactions are more likely to fail to deliver these and/or not send them in the first place. The second feature is

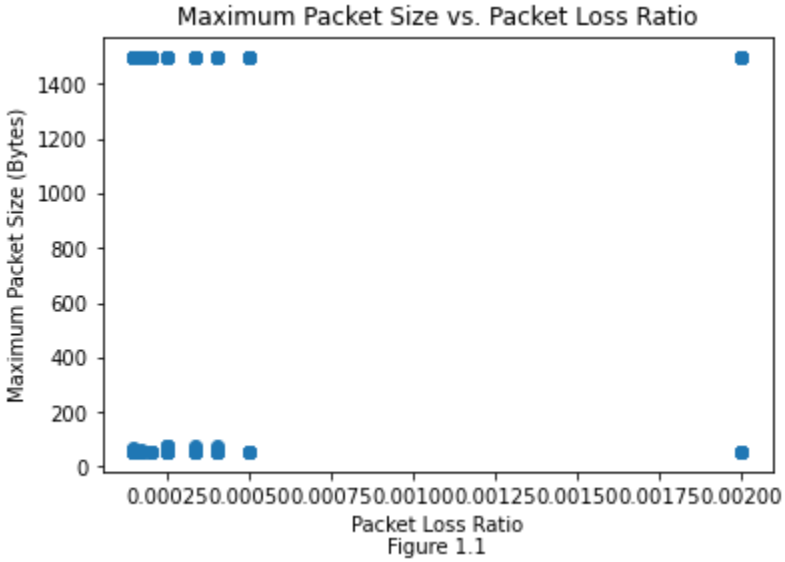


Figure 1.1

the range of packet sizes sent in each interaction. This is determined by calculating the difference between the largest and smallest packet sent in a given interaction. This feature is quite similar to the first, with the main difference being the sensitivity to anomalies. The third feature is the mean packet size sent in each interaction. The association between mean packet size and packet loss ratio/latency is unclear, but it appears to be oscillating in some manner. However, it is clear that there are different distributions among different packet loss ratios/latencies, and according to the feature importance ranks below, the models were able to effectively discern these patterns. The fourth feature is the mean packet duration for each interaction. This value is determined by calculating the difference between the time that each consecutive packet was sent, which represents the time required for each packet to be sent, then using the mean of these differences.

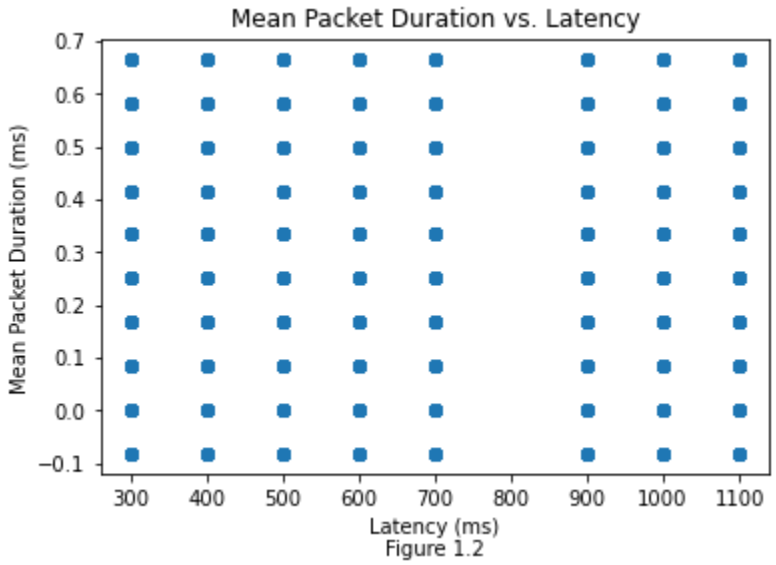


Figure 1.2

As shown in Figure 1.2, this value only occurs at very specific intervals. We believe this is explained by DANE's underlying data capture technique. However, since these values occur at different frequencies based on the latency or packet loss ratio, this feature is still able to provide value to the models. The fifth feature is the total packet direction. This value represents the overall flow of data. If more packets are being sent from Port 1 to Port 2 the value will be more positive, and if more packets are being sent from Port 2 to Port 1, the value will be more negative. The direction tends to be more one-sided and outliers are more common in interactions on a connection with a higher packet loss ratio. In terms of latency the opposite is true; lower latency connections show a wider range of directions, including more outliers. The sixth and seventh features are the total packets and bytes sent respectively. The values and trends of these features are similar to those of the third feature, mean packet size. The eighth feature is the length of each interaction, which is determined by calculating the difference between the time of the first packet in the interaction being sent and the time of the last packet being received.

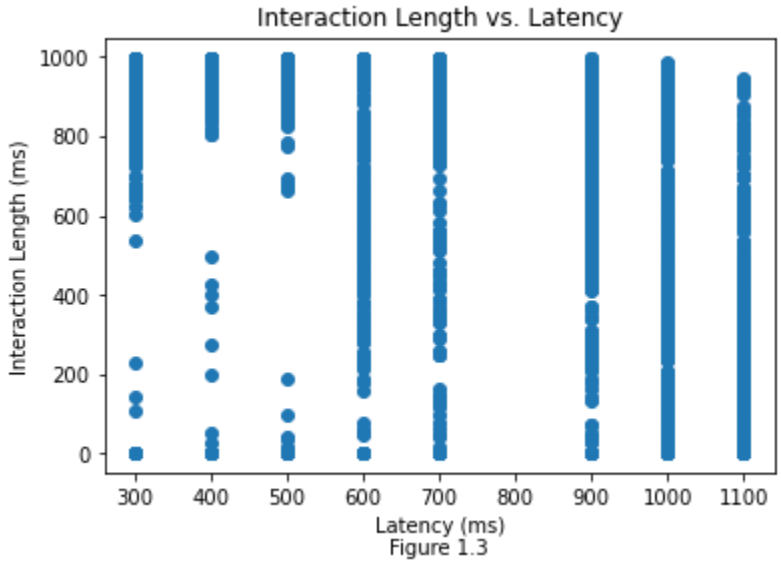


Figure 1.3

As shown in Figure 1.3, interactions on a network with a lower latency tend to have longer durations, typically above 600 milliseconds in length. Meanwhile, the interactions on networks with a higher latency have more uniformly distributed durations. This indicates that if a large number of interactions with a length of 600 milliseconds or less are observed in a particular conversation, the network is more likely to have a higher latency. The ninth and tenth features are represented by the number of packets sent over time and number of bytes sent over time respectively. These values are determined by dividing the total number of packets or bytes sent by the length of the interaction. As shown in Figure 1.4, the number of packets sent over time in individual interactions is never above 10 packets per millisecond when using a network with a high packet loss ratio. This trend is also seen when examining the total bytes sent over time in individual interactions. This is a logical and expected observation, as one should expect more information to be able to be sent per millisecond on a network that loses fewer packets.

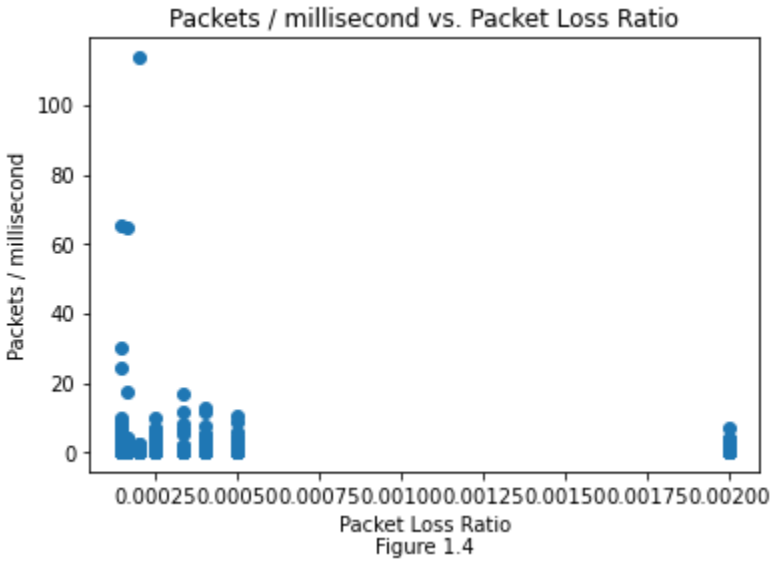


Figure 1.4

One of the biggest challenges was to evaluate the performance of features quantitatively. Scikit-learn's feature importance algorithm was used to numerically rank features in order of

importance to a model. The algorithm is a commonly used model inspection technique that allows for the numerical evaluation of features in a model. A high level overview of the algorithm involves iterating through each feature and individually computing a reference score following the formula below:

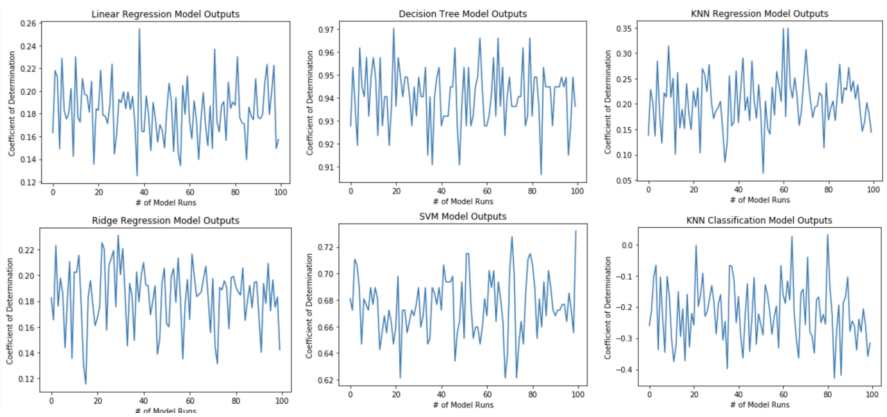
- Inputs: fitted predictive model m, tabular dataset (training or validation) D.
- Compute the reference score s of the model m on data D (for instance the accuracy for a classifier or the R^2 for a regressor).
- For each feature j (column of D):
    - For each repetition k in $1, \ldots, K$:
        - Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\tilde{D}_{k,j}$.
        - Compute the score $s_{k,j}$ of model m on corrupted data $\tilde{D}_{k,j}$.
    - Compute importance i_j for feature f_j defined as:

$$i_j = s - \frac{1}{K} \sum_{k=1}^{K} s_{k,j}$$

Results

We built 6 machine learning models to predict the latency and loss ratios: linear regression, ridge regression, decision tree, SVM, KNN regression, and KNN classification. We started with a simple linear regression model then turned to ridge regression to reduce the potential issue of overfitting. We also used the KNN regression and classifier models because they proved to work well in our previous quarter's DANE data. We also explored other classificatio models such as decision tree and SVM. The decision tree classifier predicts values by learning simple decision rules inferred from the features, while the SVM classifier finds an optimal boundary between the outputs. The model outputs are shown in the tables below:

For Packet Loss Predictions



The graphs to the left show the coefficient of determination outputs for 100 runs of each model. In other words, we recorded the r-squared values of the models after

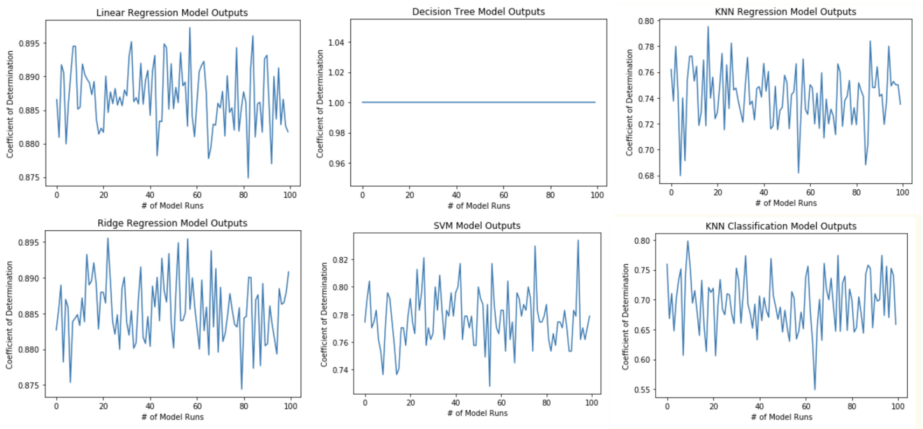
running each model 100 times. The x-axis represents each run of the models, and the y-axis represents the correlation coefficient. In the first graph, where we plot the outputs of the linear regression model, we can see that the first run of the model resulted in a r-squared value of about 0.16, while the 47th run of the model resulted in a r-squared value of about 0.25. Just by a quick glance of these graphs, we can see that on average, the Decision Tree model performed the best. Here is a summary of the models:

Model	Mean Squared Error	Coefficient of Determination
Linear Regression	3.01 * 10^-7	0.181
Ridge Regression	3.05 * 10^-7	0.183
Decision Trees	1.92 * 10^-7	0.940
Support Vector Machine	3.52 * 10^-7	0.675
KNN Regression	2.89 * 10^-7	0.204
KNN Classification	4.58 * 10^-7	-0.219

The table above represents the results of our 6 models on predicting packet loss ratios. Hyperparameters were tuned for all models to ensure the best performance and the mean squared errors were compared between them. As seen in the table, the Decision Trees model performed best in terms of lowest mean squared error and highest coefficient of determination. Although the mean square error is fairly similar and significantly small across all models, the high r-squared value makes it clear that Decision Trees is the best model to use when predicting packet loss.

Latency Predictions

The graphs on the right show the coefficient of determinations of the models predicting latency. The interpretation of these models is the same as



the graphs shown previously for loss ratio predictions. In contrast to the models' performances on predicting loss ratios, our models surprisingly performed significantly better in predicting latency. Across all six models, we see that the r-squared value was above 0.7 on average. We also see that the Decision Tree classifier might also be the best choice model for predicting latency. The table below shows a summary of the models' results in predicting latency.

Model	Mean Squared Error	Coefficient of Determination
Linear Regression	8.07 * 10^4	0.886
Ridge Regression	8.14 * 10^4	0.885
Decision Trees	0.0	1.0
Support Vector Machine	5.66 * 10^4	0.777
KNN Regression	1.83 * 10^5	0.740
KNN Classification	2.18 * 10^4	0.693

Similarly, the results from the latency predictions show that the Decision Trees model was also the best performing model in terms of mean squared error and correlation. A mean square error of 0 and correlation coefficient of 1 seems too good to be true. Although we did not get the chance to further investigate this outcome, we will make sure to validate this outcome in the future. Until then, we want to offer the KNN Classification model as the best predictor for latency as it has the smallest mean squared error compared to the other models.

Feature Importance Testing

Feature	Feature Importance Score (Packet Loss Model)	Feature Importance Score (Latency Model)
max_packet_size	0.523	0.353
range_packet_size	0.009	0.013
avg_packet_size	0.368	0.469
avg_packet_dur	0.663	0.327
total_packet_dir	0.551	0.604

total_packets	0.414	0.392
total_bytes	0.469	0.289
interaction_length	0.136	0.771
packets_time_ratio	0.201	0.254
bytes_time_ratio	0.121	0.203

Based on Scikit-learn's feature importance algorithm, the most important features were different in the packet loss prediction model and the latency model. This isn't entirely surprising because of the nature of latency and packet loss in affecting network performance, but it is important to note the most important features for each model. For packet loss the most important features were avg_packet_dur and total_packet_dir. This makes sense logically because we are predicting packet loss so it would make sense for packet features to be the most important. The least important features for the packet loss model are range_packet_size and bytes_time_ratio. For the latency model that we chose, the most important features were by far interaction_length and then total_packet_dir. The least important feature was also by far range_packet_size (it appears range_packet_size isn't an effective feature in this situation).

Conclusion

We believe that the future of the network providing industry can greatly benefit from the work we have done with our project. Being able to predict when network conditions occur at any given time for users can be instrumental in the improvement of the quality of service. And as was emphasized throughout the project, a customer's experience in an industry like this cannot be overstated. At the end of the day, the customer is the most important factor in any business decision making process and improving their satisfaction is exactly what we hoped to achieve with this project.

In terms of applying our project in the future, we have a couple ideas. One of our biggest ideas to progress this project and further develop it is to create a monitoring API dashboard that utilizes the model to make predictions on a given user's network. This can either be a customer-facing or internal tool, but the idea is that when these predictions are made (based on prior data from that

user's network) and there are network conditions detected, there could either be a flag notification from a user's perspective or there can even be some sort of change in traffic. Rerouting the traffic to a replica server or the use of a load balancer to manage traffic between multiple replicas of the server are brainstormed ideas that we had. Obviously, these ideas aren't fleshed out, but the future of this industry is very bright and we are hoping our research can contribute positively to that.

Team Contributions

In terms of the report, Shasank was in charge of the introduction, background and conclusion sections. Shasank helped with feature engineering, model selection testing, as well as hyperparameter tuning. Shasank was also responsible for the feature importance testing. Overall, the code was done together so that part was evenly distributed. Shasank was also involved with discussing all 10 features as well as helping format the report. The website was done evenly amongst all students.

In terms of the report, Michael was in charge of the methods and conclusions sections as well as helping format the report. The code was done together so Michael also helped with feature engineering, model selection testing, and hyperparameter tuning. Michael was also involved with the exploratory data analysis. Michael created 5 of the features but was also involved in discussing all 10.

Jaye contributed to the methods and results sections of the report. She was mainly responsible for building the models. She hypertuned the parameters and did some data preparations before running the models preparations. She also participated in the feature engineering discussion, deciding on which features to include in our models. Overall, the project was evenly distributed amongst all students, and all students were heavily involved in the decision making process.

References

1. A brief history of the internet. (n.d.). Retrieved December 7, 2021, from https://www.usg.edu/galileo/skills/unit07/internet07_02.phtml#:~:text=January%201%2C%201983%20is%20considered,Protocol%20(TCP%2FIP).
2. Andrews, E. (2013, December 18). *Who invented the internet?* History.com. Retrieved March 9, 2022, from https://www.history.com/news/who-invented-the-internet
3. https://www.networkcomputing.com/networking/packet-loss-vs-latency-analyzing-impact
4. Austin, D. (2018, April 26). A complete guide to network traffic analysis training. Esellweb Blog. Retrieved December 7, 2021, from https://www.esellweb.com/a-complete-guide-to-network-traffic-analysis-training/.
5. Fortunato, T., Zahid, N., Gadaj, M., Salamone, S., MacVittie, L., Bocetta, S., Korolov, M., & Edwards, J. (2019, January 10). *Packet loss vs. latency: Analyzing the impact*. Network Computing. Retrieved March 9, 2022, from https://www.networkcomputing.com/networking/packet-loss-vs-latency-analyzing-impact