Predicting Network Anomalies

Luca Dovichi, Kasey Corra, Clarice Kim

I. Intro

Our group decided to build on NetMicroscope's work from last summer. Their system automatically identified network degradation, but it could not warn network operators of impending issues. Therefore, we examined whether an ML pipeline could flag high-risk regions of a network before degradation occurs. We operationalized this goal by evaluating a range of machine learning models on a basic predictive task: given the current network state, will the next network measurement be anomalous?

The project was split into two classification tasks: first, we determined our labels for when devices have degraded performance. We would ideally do this manually, but NetMicroscope's historical dataset spans thousands of records. Instead, we utilized unsupervised models to identify anomalies automatically. Then, we built predictive models that, given a current network state, predict the likelihood of degradation during the next collection period.

II. Unsupervised Labeling

Given the size of the NetMicroscope dataset, we performed unsupervised learning to determine which points are anomalies. For a baseline comparison, we repeated the implementation from this summer: we updated an exponential moving average and flag points that are more than λ standard deviations above that mean ("basic_ema_anomaly"). We were also interested in whether our predictive models could learn more complex relationships. Therefore, we implemented another live model, (D)SPOT[1], which has a more sophisticated methodology that changes thresholds as new values are recorded ("dspot_anomaly").

Finally, NetMicroscope's current anomaly detection methodology uses a global λ to determine which points are anomalies. If any device measures a latency more than λ over its exponential moving average, it will be flagged. Of course, some devices have more variability in their traffic, and corporate clients may desire different anomaly thresholds. Therefore, we created a third labeling technique that uses a λ_i value hand-tuned for each device ("tuned_dspot_anomaly"). We believe that this better represents production deployments, where anomaly methods will differ given the needs of their clients. We evaluated degradation prediction on each labeling technique separately.

III. Supervised Predictions

After completing our unsupervised labeling, we used the dataset to train models to predict upcoming degradation. At a high level, we repeatedly (1) chose a model, (2) trained it on a subset of the labeled data, and (3) evaluated whether, given the current network state, it could classify whether the next measurement would be an anomaly. Given the time-dependence of our data, we had to be careful to train on data collected before test points. Otherwise, we could leak data about the test set and fail to reproduce a true production deployment.

III.A Feature Engineering

In order to train models to predict network degradation, we used a lookback window to represent historical network health. Each lookback window considers data from the last 30 hours (normally around 10 data points) before the prediction point and creates aggregate features from that window. The

[1] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, Christine Largouët. Anomaly Detection in Streams with Extreme Value Theory. KDD 2017 - Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug 2017, Halifax, Canada. ff10.1145/3097983.3098144ff.ffhal-01640325

aggregate features include anomaly counts and network metrics such as latency distributions. We also accounted for missing measurements by encoding the "completeness" of the data window.

Our consideration of 3 distinct labeling methods, 2 model designs, and many different model architectures resulted in a significant evaluation space. Our original proposal had included discussion of multiple prediction windows (predicting anomalies in the next day vs. week); however, we found that these additional evaluations were computationally challenging. Rather, we present our findings to determine the merits of a more comprehensive study.

III.B Train-Test Split

The original dataset was collected over three periods, each separated by over 11 days. We filtered to only include devices that were active across all three periods, allowing us to train models on the first two periods and test on the third. This allows us to prevent data leakage, since (unlike the traditional 80-20 split that samples evenly) our lookback windows (used to train the models) could never include test points. This also mimics production deployments where training data must precede the testing data.

III.C Per-Device Modeling

The distribution of each device's latencies will of course vary due to its configuration and placement in a network. The variance of these measurements also differs, meaning that devices may have different thresholds for what can be considered anomalous. As a result, we suspected that models would perform better if they were trained on each device independently, learning the unique patterns of each. To prepare the device dataset, we dropped devices that lacked anomalies in either the training or testing periods under any labeling method.

III.D Multi-Device Modeling

Of course, multi-device modeling must also account for the varied distribution of latencies across devices. We kept the same training and validation process for these models, but we added a normalization step. We changed each host's latencies to be represented by a z-score relative to the host mean. We also adjusted the normalized values to be positive by subtracting the minimum normalized value. We chose to have positive, normalized values since we planned to mark the absence of a data collection point with -1 for latency. We were careful to normalize the points according to the *training data distributions* before fitting the test points to those previous distributions. This ensures no data leakage between the training and test sets.

III.E Training

We trained and tested the following classifiers before selecting the top-performing models for both per-device and multi-device modeling: Logistic Regression, Nearest Neighbors, Linear SVM, RBF SVM, Decision Tree, Random Forest, AdaBoost, Naive Bayes.

III.F Model Evaluation Metrics

Next, we considered the metrics to evaluate our models on. Accuracy is a poor proxy for performance in this context, since there are relatively few anomalies: models can perform very well by always predicting no degradation. Instead, we evaluated the models' tradeoffs for recall and false positives. We intended to represent two users:

1. Mission-critical clients want to ensure minimal disruptions. They would be interested, therefore, in the rate of false positives required to correctly predict 90% of network anomalies.
2. Low-budget clients want to ensure minimal engineering hours. They don't want to task engineers with diagnosing non-existent issues, so they'd be interested in the number of anomalies they could detect while only having a 10% false positive rate.

III.G Selecting Best-Performing Models

To evaluate the per-device models and multi-device models, we found the best-performing model for each anomaly detection method and evaluation metric:

Best Per-Device Model for Each Evaluation Metric

Metric / Method	**basic_ema_anomaly**	**dspot_anomaly**	**tuned_dspot_anomaly**
Accuracy	Random Forest (0.8681)	Random Forest (0.9174)	Random Forest (0.9525)
FPR at 90% Recall (best to be low)	Logistic Regression (0.7983)	Linear SVM (0.8492)	AdaBoost (0.7568)
Best Recall @ 10% FNR (best to be high)	Naive Bayes (0.2127)	Linear SVM (0.1304)	Naive Bayes (0.1794)

Best Multi-Device Model for Each Evaluation Metric

Metric / Method	**basic_ema_anomaly**	**dspot_anomaly**	**tuned_dspot_anomaly**
Accuracy	AdaBoost (0.8679)	AdaBoost (0.9177)	AdaBoost (0.9653)
FPR at 90% Recall (best to be low)	Linear SVM (0.6891)	Naive Bayes (0.6189)	Naive Bayes (0.7926)
Best Recall @ 10% FNR (best to be high)	Logistic Regression (0.2928)	Logistic Regression (0.4062)	Naive Bayes (0.4783)

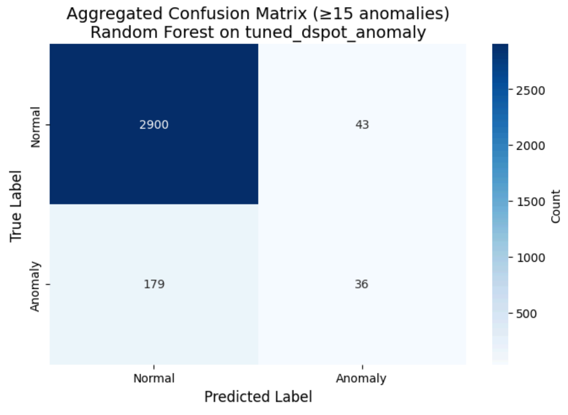
The "best" model varies depending on client goals and the deployed labeling technique. We argue that Random Forest is a good, generalist per-device model, as it balances a relatively low FPR at 90% recall and high best recall @ 10% FNR. We similarly decided that Naive Bayes was a solid multi-device model since it similarly balanced a low FPR at 90% recall and a high best recall at 10% FNR.

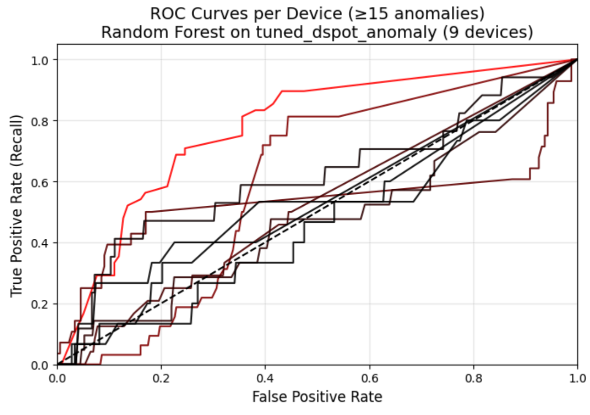
III.H Comparing Per-Device and Multi-Device Modeling

In order to compare "per-device" and "multi-device modeling," we examined the performance metrics of each model type for each anomaly detection method. Among other models, we investigated the best-performing "per-device" and "multi-device" models, Random Forest and Naive Bayes. We notice that while RF consistently has higher accuracy than NB, NB outperforms RF in FPR at 90% Recall and Recall at 10% FNR. This may indicate that multi-device models outperform per-device models; however, it could also be a result of increased training data. As discussed in the next section, devices with more anomalies generally perform better.

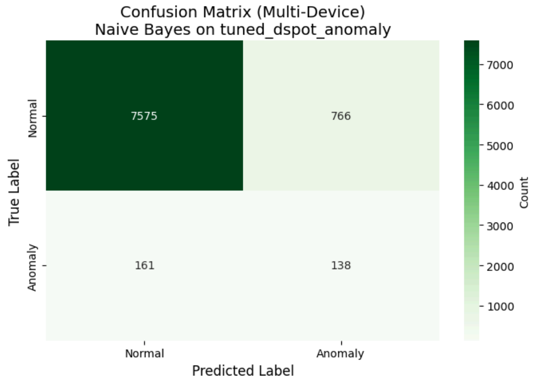
III.I Visualizing Best-Performing Models

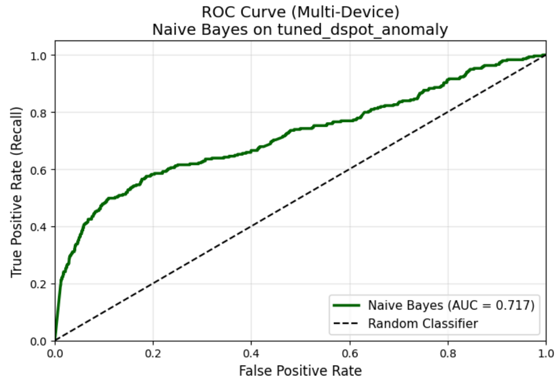
The ROC curve provides the ideal visualization for the model's performance with respect to both false positives and false negatives. Our visualization of the per-device models in particular proves promising for the application of Random Forest to this classification problem, particularly when it is trained on data with a sufficient number of anomalies.





In the ROC curve pictured above, each device's curve is colored according to the number of anomalies in its training data (brighter red indicates more anomalies present, darker red/black represents fewer). There is thus a clear improvement in the model's ability to distinguish between benign and anomalous classes when there are more anomalous samples present in the training data. This presents a promising indication that future iterations of this same process on larger test sets would show better predictive performance.





The multi-device results indicate that the Naive Bayes model correctly identifies most normal samples, but produces a significant amount of false positives, missing over half of the anomalies at the threshold. However, the early curve reveals that the model performs quite well when operating under strict false-positive limits – it identifies anomalies (relatively) well when restricted to a small number of false alarms.

IV. Conclusions

1. **There is predictive power in detecting anomalies.**
   Model performance indicates that models may not be able to predict every anomaly without many false positives. Models can, however, identify anomalies more often than random chance.

2. **Multi-device models slightly outperform per-device models in predicting degradation.**
   Further research is warranted to understand whether such a conclusion is a result of (1) the amount of training data for per-device versus multi-device models or (2) patterns of degradation generalizing across devices.

3. **A customer's needs determine which models they choose to make their predictions.**
   Because different models perform best on overall accuracy, Recall @ 10% FPR, and FNR @ 90% recall, model selection will depend on a customer's tolerance for false alarms or occasional failure for predicting upcoming anomalies.

4. **Repeating with more anomalies could improve model performance.**
   The ROC curve for the per-device RF model indicates that devices trained on larger numbers of anomalies have better performance.