THE UNIVERSITY OF CHICAGO

CONCEPT DRIFT CHARACTERIZATION, EXPLANATION, AND MITIGATION IN
CELLULAR NETWORKS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCE
IN CANDIDACY FOR THE DEGREE OF
MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

BY
SHINAN LIU

CHICAGO, ILLINOIS
MAY 26, 2022

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Operational networks commonly rely on machine learning models for many tasks, including detecting anomalies, inferring application performance, and forecasting demand. Yet, unfortunately, model accuracy can degrade due to *concept drift*, whereby the relationship between the features and the target prediction changes due to reasons ranging from software upgrades to seasonality to changes in user behavior. Mitigating concept drift is thus an essential part of operationalizing machine learning models, and yet despite its importance, concept drift has not been extensively explored in the context of networking—or regression models in general. Thus, it is not well-understood how to detect or mitigate it for many common network management tasks that currently rely on machine learning models. Unfortunately, as we show, concept drift cannot always be mitigated by frequently retraining models using newly available data, and doing so can even degrade model accuracy further. In this paper, we characterize concept drift in a large cellular network for a major metropolitan area in the United States. We find that concept drift occurs across many important key performance indicators (KPIs), independently of model, training set size, and time interval—thus necessitating practical approaches to detect, explain, and mitigate it. To do so, we develop Local Error Approximation of Features (LEAF). We introduce LEAF and demonstrate its effectiveness on a variety of KPIs and models. LEAF detects drift; explains features and time intervals that most contribute to drift; and mitigates drift using resampling, augmentation, or ensembling. We evaluate LEAF against industry-standard mitigation approaches (notably, periodic retraining) with more than three years of cellular KPI data from a major cellular provider in the United States. LEAF consistently outperforms periodic retraining while reducing costly retraining operations by as much as an order of magnitude. Due to its effectiveness, a major cellular carrier is now integrating LEAF into its forecasting and provisioning processes.

CHAPTER 1

INTRODUCTION

Network operators rely on machine learning models to perform many tasks, including anomaly detection Shon and Moon [2007], Shon et al. [2005], performance inference Futuriom [accessed August, 2021] and diagnosis, and forecasting Mei et al. [2020], Chinchali et al. [2018]. Deploying and maintaining the models can prove challenging in practice Sommer and Paxson [2010]; a significant operational challenge is *concept drift*, whereby a model that is initially accurate at a particular point in time becomes less accurate over time—either due to a sudden change, periodic shifts, or gradual drift. Previous work in applying machine learning models to network management tasks has generally trained and evaluated models on fixed, offline datasets Sinclair et al. [1999], Dong and Wang [2016], Shon and Moon [2007], Shon et al. [2005], Ayoubi et al. [2018], Mei et al. [2020], Chinchali et al. [2018], demonstrating the ability to predict various network properties or instances at fixed points in time on a static dataset. Yet, a model that performs well offline on a single dataset may not in fact perform well in practice, especially over time as characteristics change.

Models can become less accurate at predicting target variables for many reasons. One cause is a sudden, drastic change to the environment. For example, the installation of new equipment, a software upgrade, or a sudden change in traffic patterns or demands can cause models to suddenly become inaccurate. One notable instance that exhibited such a sudden shift was the COVID-19 pandemic, an exogenous shock that resulted in significant changes to behavior patterns Lutu et al. [2020] (*e.g.*, reduced mobility) and traffic demands Liu et al. [2021] (*e.g.*, as people suddenly began using home WiFi connections more for specific applications and relying less on mobile data). Another possible cause is periodic change, whereby a model that is accurate on a particular time window may become less accurate but will exhibit higher accuracy at fixed, periodic intervals in the future. For example, one clear phenomena that we observe is drift in model accuracy with a seven-day period; diurnal

and periodic patterns in network traffic are, of course, both well-documented and relatively well-understood. A new contribution in this paper, however, is to demonstrate that machine learning models also exhibit similar patterns with respect to model accuracy.

Concept drift is a relatively well-understood phenomenon in machine learning and has been studied in the context of many other prediction problems Schlimmer and Granger [1986], Lu et al. [2018], Gama et al. [2014]. Yet, although concept drift is a relatively well-understood phenomenon in other fields, it has not been explored in the context of computer networking, in spite of the relatively widespread use of machine learning models.

Mitigating concept drift for Key Performance Indicators (KPIs) in a cellular network introduces fundamentally new challenges that make previous approaches from other domains inapplicable. In contrast to tasks such as image or text classification, where the semantics of prediction occurs on a fixed object and characteristics of the features change relatively slowly over time Fdez-Riverola et al. [2007], Žliobaitė [2010a], Gama et al. [2014], predictions of network characteristics occur continuously over time, and occur within the context of a system that changes over time due to both gradual evolution and exogenous shocks. Additionally, networks have unique characteristics, such as dynamic signal interference due to environment changes (*e.g.*, changes due to weather, seasonality, etc.) Zheng et al. [2014], which calls for new approaches. One important implication of our findings is that one of the most common practices for combatting drift in large networks—regularly retraining models Kantchelian et al. [2013], Thomas et al. [2011]—is often ineffective and can in some cases even reduce model accuracy. Due to the nature of certain components of drift, continually retraining using a fixed-size window of recent observations can *degrade* model performance!

Beyond simply detecting and mitigating concept drift, operators often want to know *why* a model has become less accurate. Thus, the ability to explain the behavior of black-box regression models is necessary to help operators use these models in practice. To this end, this paper not only characterizes concept drift in cellular networks, but also develops

(1) approaches to explain how different features contribute to drift; and (2) strategies to mitigate drift through triggered, focused re-retraining, re-sampling, and ensembling approaches that are more efficient than naïve retraining. Towards this goal of *explainable AI (XAI)* models in networking, this paper studies drift in the context of cellular networks and develops new techniques to mitigate concept drift. Although past work has developed methods to help explain concept drift, past work has largely focused on classification problems Jordaney et al. [2017], Yang et al. [2021]; in contrast, prediction problems in the content of cellular networks are often regression problems, which generally lack methods to explain concept drift.

This paper makes three contributions:

First, **we characterize concept drift in the context of a large cellular network**, exploring drift for many cellular key performance indicators (KPIs) using more than three years of KPI data in a major United States city and surrounding metropolitan area, from one of the largest cellular providers in the United States. We demonstrate concept drift in a large cellular network comparing different KPIs, model families, training set sizes, and periods across many regression models and tasks. Concept drift occurs consistently and independently of both the size and period of the training set.

Second, **we introduce *LEAF* (Local Error Approximation of Features) to detect, explain, and mitigate concept drift in cellular networks**. We apply Kolmogorov-Smirnov Windowing (KSWIN) to time-series of estimated errors, which tells us when a model drifts. We complement existing drift explanation methods in regression-based supervised learning tasks. We use LEAplot and LEAgram to inform operators about for *which feature*, *where* and *how much* concept drift occurs, which maps to the under-trained region of a model. Based on these explanations, the framework strategically re-samples, augments the training data, and creates spatio-temporal ensembles to mitigate drift.

Third, **we evaluate these approaches holistically, on more than three years of data from a large cellular carrier, evaluating both the effectiveness and cost**

3

of each approach, across many KPIs and families of machine learning models. LEAF consistently outperforms periodic retraining, while reducing costly retraining operations by as much as an order of magnitude. Although there is no single best mitigation strategy for different types of models, we consistently reduce errors across KPIs. We also find that KPIs with higher coefficient of variance are harder to mitigate. In terms of approaches, resampling tends to be the best mitigation approach for boosting and bagging regression models, augmentation is most effective for distance-based models, and ensembling works best for recurrent neural networks.

A major cellular carrier in the United States is integrating LEAF into its forecasting and provisioning processes. We believe that LEAF can also be applied beyond cellular networks, to other network management problems that use black-box models for regression-based prediction, such as demand forecasting and application performance prediction in ISPs and service provider networks. Such avenues present rich opportunities for future work.

CHAPTER 2

PROBLEM SETUP

We now introduce the problem we tackle in this paper, describing first the nature of the data collected in a cellular network and then the specific modeling problems we address, including the specific features and prediction targets.

2.1   Dataset

We base our analysis in this paper on more than three years of daily eNodeB-level LTE network measurements collected from a major wireless carrier in the United States. Table 2.1 summarizes the characteristics of the dataset. The dataset contains information collected from 898 eNodeBs (evolved NodeBs, or the "base station" in the LTE architecture) in a large city and surrounding metropolitan area (rural, suburban, and urban included) in the United States. The dataset spans more than three years—from January 1, 2018 to May 3, 2021—and contains 737,577 logs.

Collection period	Jan. 1st 2018 – May 3rd 2021
Identifiers	eNodeB ID & Time stamp
Number of KPIs	224
Groups of KPIs	Network performance Resource utiliazation User experience
KPIs of interest	Downlink volume (DVol) RRC est. success (REst.) Downlink Throughput (DTP) S1-U call drop rate (CDR) Peak active UEs (PU)
Number of eNBs	898
Number of logs	737,577

Table 2.1: *Summary of dataset.*

Each log contains 224 Key Performance Indicators (KPIs) collected for a base station on

a particular date. KPIs are statistics collected and used by the operator of the network to monitor and assess network performance. The 224 KPIs fall into three categories: (1) access network performance (*e.g.*, throughput, access congestion, packet loss, establishment success), (2) resource utilization (*e.g.*, data volume, peak active users, active session time, cell availability rate), and (3) user experience features (*e.g.*, call drop rate, abnormal UE releases, VoLTE user satisfaction rate). Further, some of the KPIs have separate directional measurements, such as the downlink where the network is transmitting the data down to the UEs and the uplink where the network is receiving data from the UEs. The data also contains features like unique identifiers of dates, eNodeBs, and their characteristics (*e.g.*, the density of their deployment location).

KPIs are typically calculated using eNodeB counters, measurements, and events generated by user equipment (UE) and radio access network (RAN) equipment. Counters are incremented based on network events that are generated or received by the eNodeB. Some metrics are established to measure the network based on the LTE standard released by 3GPP 3GPP [accessed July, 2021], the body that determines cellular standards. These "raw" measurements can be collected from the eNodeB using external tools that store data for longer periods of time or build formulas (which resemble KPIs but are generated from one or many underlying raw KPIs). One example of a formula to create a KPI is drop rate, which is calculated by dividing the number of abnormal releases by the number of successful established connections. KPIs are stored as statistical representations (*e.g.*, min/max, average, percentiles).

2.2   Modeling Goal

We focus our study of concept drift in the context of network capacity forecasting. Capacity forecasting is an important problem for operators as it guides infrastructure configuration, management, and augmentation. We focus on per-eNodeB level KPI forecasting to provide suggestions for capacity adjustment, deployment, maintenance, and operation in large cellular networks. We aim to predict the following KPIs that are most relevant for use in capacity

Characteristic	DVol	REst	DTP	CDR	PU
Std/Mean	0.81	0.85	0.59	2.48	1.76
Periodic	✓	✓	✓	✓	✓
Bursty				✓	✓
Data Lost					✓
Balanced			✓		✓

Table 2.2: *Characteristics of target KPIs. DVol: Downlink volume; REst: RRC establishment success; DTP: Downlink Throughput; CDR: S1-U call drop rate; PU: Peak active UEs.*

planning: measurements of network performance (downlink throughput, RRC / Radio Resource Control establishment success), resource utilization (downlink data volume, peak number of active UEs / User Equipment), and user experience (S1-U / S1 User plane external interface call drop rate). Table 2.2 summarizes the main characteristics of these KPIs.

These KPIs exhibit a variety of statistical patterns and characteristics which, as we will see in Section 5 can ultimately affect how models drift over time, as well as the best strategies for mitigating drift for a particular KPI. All KPIs exhibit 7-day periodicity, although some KPIs exhibit far more variance than others. *Downlink volume (DVol)* and *RRC establishment success (REst)* present similar ranges in their distributions. In contrast, *S1-U call drop rate (CDR)* and *peak active UEs (PU)* show a more bursty behavior over time. While the data distributions of *downlink throughput (DTP)* and PU have balanced distributions that do not present a long tail, DVol, REst, and CDR present more skewed distributions. Some data for PU was lost between July 2019 and January 2020. We use historical data—*i.e.*, all available KPIs and dates (as features) up to a given day—to forecast one or more target KPIs of interest 180 days in the future. We use a 180-day forecast window because the model outputs we focus on are used *in practice* for network infrastructure provisioning and augmentation over long timescales.

2.3   Experiment Setup

We aim to evaluate the prediction performance of different models for a number of KPIs 180 days in the future. The nature of this problem is regression with time-series information. Regression models are a better fit than classification because (1) we aim to forecast target KPIs that are all characterized wide ranges of possible numerical values and (2) fine grained forecasting better enables operators to understand the KPIs and take action on their network.

To explore the performance of different widely-adopted regression techniques, we use for all experiments in this paper the AutoGluon AI [accessed July, 2021] AutoML pipeline and TensorFlow. AutoGluon is used to quickly prototype deep learning and machine learning algorithms on various existing frameworks: LightGBM, XGBoost, CatBoost, Scikit-learn, MXNet, and FastAI. For all selected models, it enables automatic data processing, architecture search, hyperparameter tuning, and model selection and ensembling. We emphasize that the goal of our study is not to create and tune models that maximize performance, but rather to demonstrate and better understand concept drift in a real-world network.

We select five different families of models: (1) gradient boosting algorithms like LightGBM, LightGBMLarge, LightGBMXT, CatBoost, and XGBoost; (2) bagging algorithms such as Random Forest and Extra Trees; (3) distance-based algorithms like KNeighbors; (4) recurrent neural networks like LSTM. All models either incorporate temporal features (*e.g.*, time stamps), or are time-series models (LSTM). Although it is viable to fine tune each model's hyperparameters by hand, we rely on the default auto-selection pipeline with the goal of maintaining a fair comparison and to make training scalable and efficient. For example, the LSTM network is implemented in TensorFlow, which has 100 units for the LSTM, followed by a dropout layer and a dense layer.

For all experiments, we develop models for each selected target KPI (listed in Table 2.1). As the input of the models, we use a portion of the history of all categorical and numerical

KPIs from all eNodeBs present in our dataset up to the date when the model is generated. While we target the forecasting of the target KPIs on an eNodeB-by-eNodeB basis, we generate a single model for the entire network, *i.e.*, we create a single model capable of forecasting values for each individual base station. We discard string-based features (*e.g.*, eNodeB ID) because they might generate undesired bias that could affect accuracy when predicting eNodeBs not present in the training set, *i.e.*, text representation might be encoded to representations which are not meaningful for base stations that are not present at all times in the dataset.

2.4   Metrics for Evaluating Drift

The characterization of model performance over time is essential to evaluate concept drift. Given that we model network capacity forecasting as a regression problem, we test model effectiveness by computing distances between prediction and ground truth by date, specifically using Root Mean Squared Error (RMSE). To better understand drift across different KPIs whose natural operating value ranges are drastically different (*e.g.*, call drop rates are scalars mostly less than 1, while downlink volume scalars are often greater than 300,000), we normalize the RMSE and derive the Normalized Root Mean Squared Error (NRMSE):

$$NRMSE(M, t) = \frac{\sqrt{\frac{1}{N_t} \Sigma_{j=1}^{N_t} (M(k_{tj}) - \widehat{M(k_{tj})})^2}}{\max(k_{tj}) - \min(k_{tj})} \tag{2.1}$$

Here we denote M as the model we aim to evaluate, t denotes a specific date, and k refers to the target KPI being forecasted. k_t is the subset on date t, N_t is the number of data points on this subset, $M(k_{tj})$ denotes the jth measurement, while $\widehat{M(k_{tj})}$ identifies the jth prediction of M.

In addition to offering an equitable comparison across multiple KPIs, the NRMSE is well-suited to understand concept drift because (1) it penalizes large errors that can be

costly for ISPs operations (*e.g.*, over-provision of resources); and (2) it captures the relative impact of errors over extended periods of time. In practice, NRMSE scores under 0.1 and R^2 over 90% indicate that the regression model has very good prediction power Nau [2014]. To avoid possible error in interpretation, we also test the performance of regression by the other metrics including coefficient of determination (*i.e.*, R^2), mean absolute percentage error, mean absolute error, explained variance score, Pearson correlation, mean squared error, and median absolute error. We omit these metrics for brevity. We observe that all phenomena we describe in the paper using NRMSE also hold for these metrics.

Average NRMSE distance from a static model. To show the longe-term effectiveness of different mitigation schemes, we study the relative evolution of errors over extended periods of time. For each day in the dataset, we compute the average NRMSE across all eNodeBs and compare it to the error generated by a model that is never retrained, *i.e.*, a static model. We define $\Delta\overline{NRMSE}$ as the average distance between the error for a mitigated model M_1 against the static model M_0 in the form of a percentage distance. We define:

$$\Delta\overline{NRMSE}(M_1, M_0) = \frac{\overline{NRMSE}(M_1) - \overline{NRMSE}(M_0)}{\overline{NRMSE}(M_0)} \times 100\% \qquad (2.2)$$

where $\overline{NRMSE}$ is the average over time.

CHAPTER 3

DRIFT CHARACTERIZATION

We explore how trained forecasting models can be affected by concept drift by training a number of models from different categories of regression-based predictors, targeting different KPIs. We then study how different training set sizes and training periods can affect model performance. After identifying concept drift behaviors in cellular network, we evaluate the effectiveness of periodic retraining with recent data, which is the state of the art for mitigating concept drift. We demonstrate that it is hard to identify a single retraining strategy across models and KPIs, making naïve retraining a difficult strategy to implement in practice.

3.1 Drift Behavior for Different KPIs

In this section, we explore whether we can observe drift for the KPI forecasting task and additionally whether the drift patterns from different KPIs are similar. We experiment with different KPIs. We train all the models mentioned in Section 2.3 for each target KPI. To keep all other variables the same, the inputs of models are completely unchanged, but we alter the target KPI to predict. We use a 90-day window of historical data from all eNodeBs with an end date of July 1, 2018 for our training data, *i.e.*, forcasting KPIs starting from December 28th, 2018 (we plot from Mid-March 2019 because of data losses between January and March 2019). We then test these models on data subsets split by date. NRMSEs between the predictions and ground truth values are calculated daily using 4 different models: CatBoost, ExtraTreesMSE, KNeighborsDist, and LSTM.

Different KPIs exhibit different drift patterns. Figure 3.1 presents the concept drift along with time for four classes of KPIs. Overall, the drift patterns are quite unique for each class, and they vary in two aspects. First, deviations in NRMSE occur at different periods of time. In Figure 3.1(a), NRMSE of downlink volume experiences a sudden shift in April 2020 (corresponding to quarantine period due to COVID-19), and gradually drifts back to normal

values in later months of 2020. Conversely, the throughput prediction model in Figure 3.1(b) witnesses a rather stable NRMSE series. Figure 3.1(c) exhibits higher values and larger fluctuations before COVID-19, but the model stabilizes and shows improved NRMSE values after April 2020. Moreover, short-lived, abrupt increases in error are much more frequent than for any other KPIs, due to the burstiness of CDR. For the prediction of peak users, Figure 3.1(d) demonstrates that July 2019 to November 2019 are harder to predict, because of lost data.

Second, the high-frequency components have different patterns for different KPIs. By using signal processing techniques like STFT, no obvious weekly pattern is found on NRMSE of CDR. But if we look at the 3-week insets of Figures 3.1(a), 3.1(b), and 3.1(d), three repetitions of similar signal patterns appear, which indicates a weekly pattern.

Different KPIs are most effectively predicted using different models. As shown in Figure 3.1, we can find a model that performs relatively well for each target KPI. For all the target KPIs, there is at least one model with NRMSE less than 0.1 for at least half of a year. For example, even for call drop rate, the KPI that is most challenging to accurately predict (due to high coefficient of variances), the average NRMSE from CatBoost is 0.069. And ExtraTrees in volume has an average NRMSE of 0.098.

3.2 Drift Behavior for Individual KPIs

In this section, we analyze how changes in the model used, and how it is trained, impact concept drift. We use a single KPI (downlink volume) as an illustrative example.

Individual KPI predictions drift consistently across different models. As mentioned earlier, we use four different models to check for concept drift over time, while predicting KPIs 180 days in the future. We find (Figure 3.1(a)) that all models exhibit the same pattern. For example, between April and October 2020, all models exhibit a drastic rise in NRMSE due to the COVID-19 pandemic. We find similar model drift across other metrics such as R^2, mean absolute error, etc. Given our observations that CatBoost performs well when instances

of drift (*e.g.*, COVID-19 jump in Volume and data loss for Peak UEs) happen across KPIs, we use it for the rest of this paper to simplify presentation.

Consistent drift appears when trained using different training set sizes. To investigate how parameters of a model could impact concept drift, we evaluate the impact of varying the training set size. We vary the size of historical data from eNodeBs with an end date of July 1st, 2018 and retrain models for each size training window. As Figure 3.2(a) illustrates, all NRMSE values of different training set window sizes drift similarly over time. While training set windows of one month of data lead to a higher NRMSE, and three months and one year witness a glitch around June 2020, the signal pattern remains the same: no matter what training set size, NRMSE experiences a sudden increase after COVID-19 lockdown and gradually recovers after October 2020.

In Figure 3.2(a), we see that the model effectiveness of two weeks performs very similarly to that of one year, while model training time on two weeks is *18x* more efficient than one year. Given these results, we use two weeks for the training set window for the rest of the paper if not otherwise specified. This optimizes the tradeoff between performance over time (*i.e.*, robustness) and efficiency.

Consistent drift appears when trained on different periods. Since we are testing the long-term effectiveness of our model, another concern is that whether the choice of the training period would affect the drift. In this experiment, we train on a number of different 14 day windows of historical data from all eNodeBs. In Figure 3.2(b), the legend shows which 14 days are selected for training: 14 days of data before the date. For each training set, the model is tested daily in the future using a 180 day forecast.

Figure 3.2(b) also suggests that, no matter which training period, all NRMSE values drift simultaneously at similar timestamps. Our experiment also suggests that models trained on more recent time periods do not necessarily result in better model performance. For example, during training, the green line uses all the data from June 17th to July 1st, 2019 as features

and the downlink volume from December 18th to January 1st, 2020 as the forecasting target. This means that the model is trained on a rather abnormal period—winter holiday before the beginning of the COVID-19 lockdowns. Doing so results in less accurate model predictions on the following test dates, especially when drastic changes brought by COVID-19 quarantine begin. On the contrary, model trained on older data (blue line in Figure 3.2(b)) preserves the performance better during COVID-19 lock downs. This observation opens new possibilities for future research in exploring the best strategies for adapting to drift.

3.3   Naïve Retraining: Does It Work?

In operational networks, a common approach to counteract potential concept drift is to retrain models regularly. Retraining using the most recent data is often considered an effective way to deal with concept drift. Many existing solutions Kantchelian et al. [2013], Thomas et al. [2011] adopt this approach. To understand whether this approach is indeed effective and further analysis on concept drift is necessary, we retrain a number of different models regularly using different retraining approaches, such as altering the frequency of retraining. Note that we do not fine-tune the models because fine-tuning is very costly for retraining and thus impractical, due to the need of manual adjustments from human experts. For this experiment, we use a training set of 14 days to forecast traffic volume 180 days in the future, using CatBoost. Given a retrain frequency N, a model is trained using using 14 days of data and evaluated using the NRMSE for the N following days. The model is then retrained iteratively every N days. We choose the model, period, and training set size based on the observations presented later in Section 3.

To our surprise—and counter to conventional practice—we find that naïvely retraining regularly is not an optimal approach to adapt to concept drift. Compared to a static model, Naïve retraining is either ineffective, or effective but very costly, requiring a large amount of retrains. In Table 3.1, we show the average NRMSE changes compared to the static model. For most models, *i.e.*, DVol, REst, and DTP, the more frequently a model retrains, the

14

Retraining frequency	$\Delta\overline{NRMSE}$ of Target KPIs					#Retrains
	DVol	REst	DTP	CDR	PU	
Static	–	–	–	–	–	0
7 days	-28.46%	-27.94%	-24.03%	32.59%	-18.41%	149
30 days	-17.21%	-18.11%	-19.82%	1.16%	-10.72%	37
90 days	-0.45%	0.08%	-14.99%	-3.88%	-1.09%	12
180 days	0.84%	0.04%	-10.52%	-1.14%	1.80%	6
365 days	-5.02%	0.78%	-6.20%	0.85%	-0.33%	3

Table 3.1: *Changes of average NRMSE and number of retrains, over time, for different periodic retraining strategies. Periodic retraining is often ineffective, especially for longer timescales.*

better results we obtain. Unfortunately, while a 7 days retraining period can mitigate a large portion of errors, frequent retrains can become very costly and require 12x more retrains than a 90 days retraining period. Further, we observe that this trend breaks for bursty KPIs such as CDR (32.59% of error increased). In this case, frequent retraining can even adversely affect model performances. Overall, we conclude that naïve retraining is impractical because it normally works all at high retraining frequencies and requires specific tests across different KPIs to at best fine-tune its performance. Both are challenging when run at scale in operational networks. The above results demonstrate that naïvely retraining models is not an optimal solution to combat concept drift, motivating the rest of this paper.
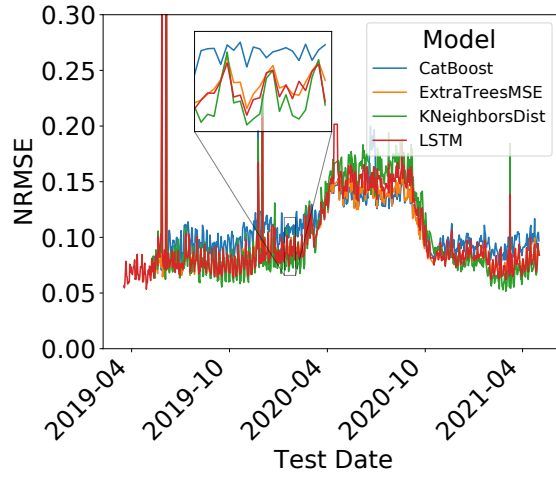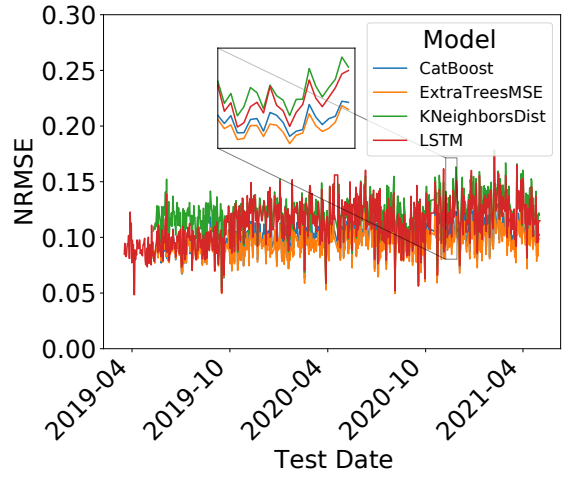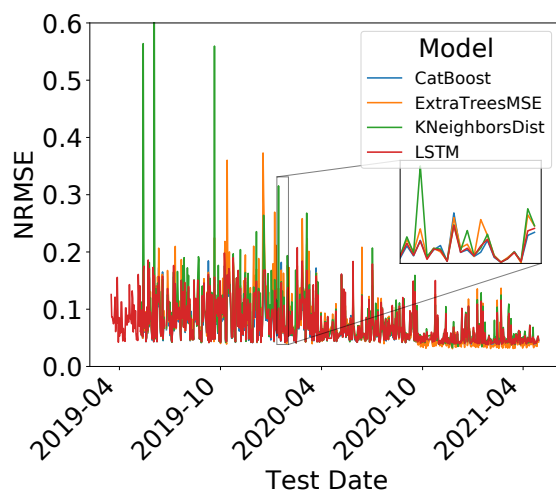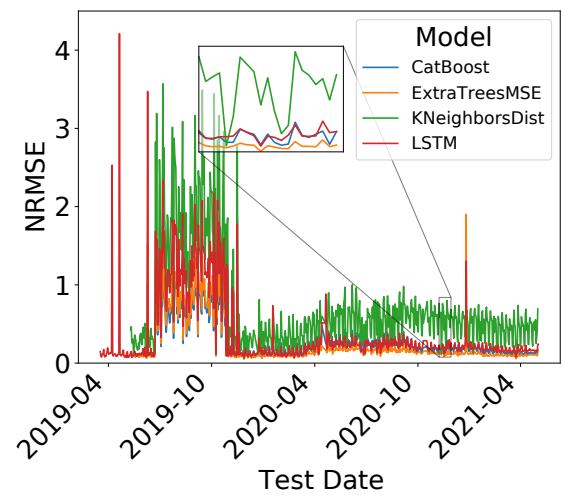
Figure 3.1: *Drift of different models for KPIs of interest. Inset figures exhibit a 3-week view (all starting from Sunday) of NRMSE for the box-selected period. We omit REst because similarity with volume. Some data is lost between July, 2019 and January, 2020 for Peak active UEs.*

16
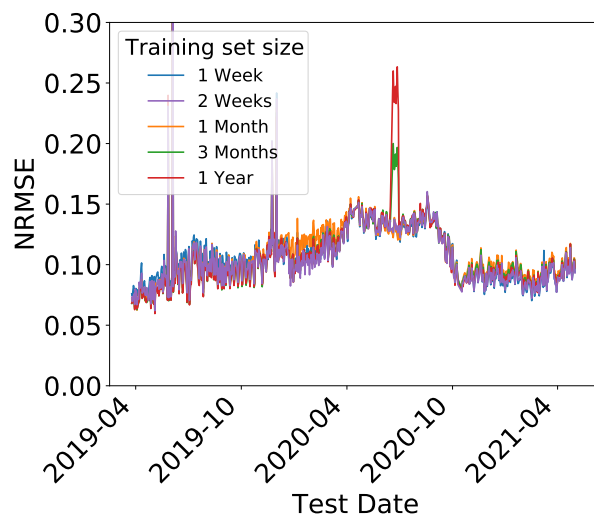
((a)) *Training set size.*
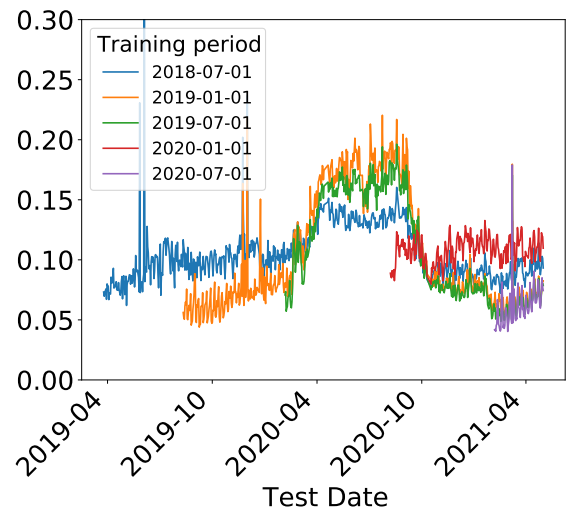
((b)) *Training set period (training set size = 2 weeks).*

Figure 3.2: *Effects of training set size and training set period on concept drift for the downlink volume KPI*

CHAPTER 4

LOCAL ERROR APPROXIMATION OF FEATURES (LEAF)

In this section, we introduce LEAF, a framework for drift detection, explanation, and mitigation. A common challenge in evaluating the performance of machine learning-based solutions is that only a subset of algorithms (*e.g.*, linear or logistic regression, KNeighbors, decision trees, decision rules) are interpretable Molnar [2020]. Unfortunately, in practice, a wide variety of uninterpretable models are used (*i.e.*, outsourced models such as OpenAI GPT-3 Brown et al. [2020], or black-box models). LEAF aims to work with any regression-based supervised model, and provide detection, explanation, and mitigation capacities independently of the model used. In this section, we describe the core intuition behind LEAF, followed by the technical details for each component. We use a concrete example (applying CatBoost to forecast downlink volume) to illustrate the framework in an operational setting.

4.1   LEAF Framework Overview

LEAF compensates for concept drift in machine learning models by implementing a pipeline of three sequential components: 1) a drift *detector*; 2) a set of tools to *explain* drift for a feature; and 3) a drift *mitigator*. Note that LEAF does not require the use of any specific model nor internal access to the model. Instead, it solely requires access to the previously used training set data, new data as it arrives, and the generated model for execution.

Figure 4.1 shows the three steps in LEAF's pipeline: First, the detector ingests the outputs of the in-use model in the form of NRMSE time-series to determine whether drift is occurring. The detector applies the well-known Kolmogorov-Smirnov Windowing (KSWIN) method Raab et al. [2020] on the time-series to identify a change in distribution of the output error, providing a signal on whether drift is occurring. If drift is detected, the explainer is triggered, indicating the time instance at which drift occurred. The explainer uses LEAplot and LEAgram to characterize the drift. These tools provide insights on the time intervals
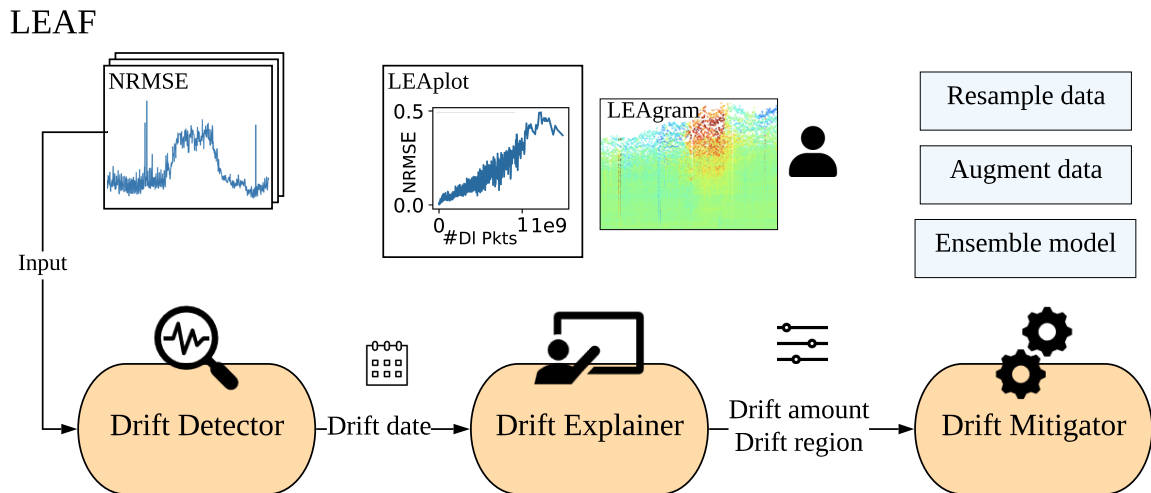
Figure 4.1: *The LEAF framework that detects (§ 4.2), explains (§ 4.3), and mitigates (§ 4.4) concept drift.*

where drift is most severe, and offer guidance for model compensation. Based on the error distribution, the mitigator automatically resamples or augments using previous data, or creates ensembles for targeted regions.

Operators can use any or all of LEAF's components to gain insight into their own inference or prediction pipelines. Operators can be notified of the detection of drift. They can take the output of the drift explainer to further understand drifting features, their severity, and the regions in the dataset experiencing the most drift. Finally, the mitigator is automatically triggered to provide effectiveness metrics for several candidate mitigated models. After a monitoring phase, operators can either let LEAF automatically replace the original model, or decide on an appropriate manual course of action based on the metrics that the model provides.

4.2   Detection

The drift detection module monitors the NRMSE time-series and notifies operators when drift occurs. Although drift detection is critical, it is not the main focus of this work, as there is significant prior work on drift detection. Our main research contributions are in

19

the areas of drift explanation and mitigation. Therefore, we apply the most effective drift detection techniques in this well-explored area Lu et al. [2018], Gonçalves Jr et al. [2014], Barros and Santos [2018], Bifet and Gavalda [2007]. We feed NRMSE time-series into Kolmogorov-Smirnov Windowing (KSWIN) Raab et al. [2020], Togbe et al. [2021][1]. Note that we do not have the ground truths of drifts in our dataset, thus we cross-check detection results with clear signals (*e.g.*, missing data, network changes due to COVID-19).

KSWIN is a recent concept drift detection method Raab et al. [2020] based on KS test, which is a non-parametric statistical test that makes no assumption of underlying data distributions Togbe et al. [2021]. It monitors the performance distributions as data arrives. We take the CatBoost NRMSE time-series to forecast downlink volume as an example; the model is trained on a total volume of 14 days of data before July 1, 2018 (illustrated in Figure 3.2(a)). By applying KSWIN, we observe that instances of drift are detected when the data exhibits major anomalies around June 2019, December 2019, and April 2021. The beginning and end of the COVID-19 quarantine period are also effectively detected. We tested KSWIN across the NRMSE time-series of the five KPIs of interest, using different model types and different training set sizes and periods. KSWIN performs well across all tasks.

4.3   Explanation

To assist mitigation and help operators understand drift, we define our goals of the drift explainer as follows: (1) to understand the extent of drift in a given feature range, (2) to map errors back to operational costs and decompose them in a spatial-temporal manner. The LEAF explanation module is based on the idea of local error decomposition: In any regression-based black-box model, local errors may occur in a specific range of a given feature or correlated feature set. Upon drift is discovered in the NRMSE time-series (*e.g.*, COVID-19

---

1. We also tested ADWIN, DDM, HDDM, EDDM, PageHinkley, but KSWIN was the most effective on our NRMSE series.
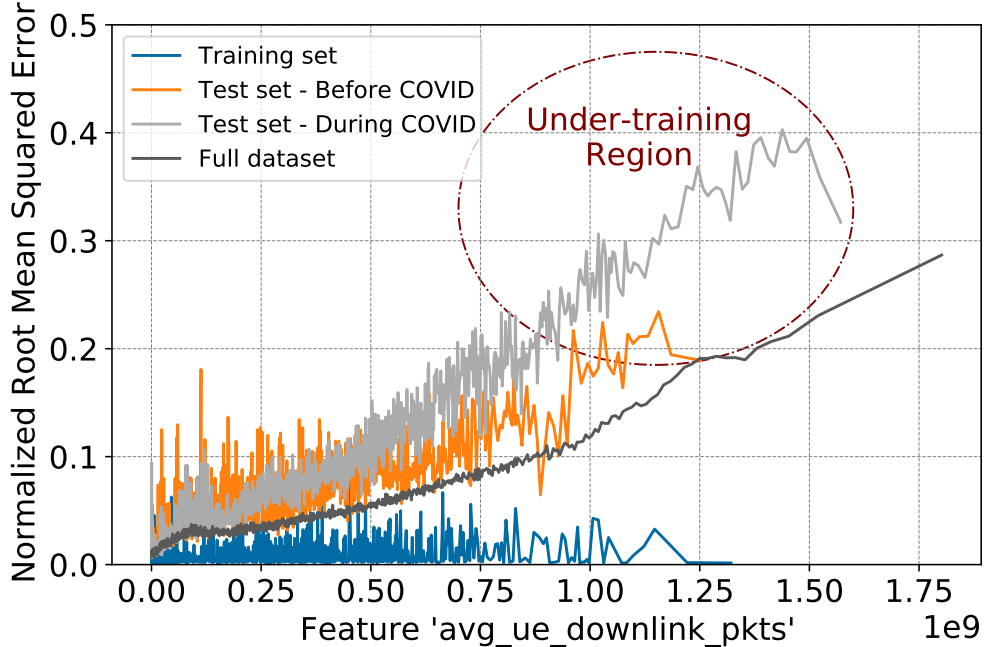
Figure 4.2: *The LEAplot (1,000 bins) that decomposes CatBoost NRMSE time-series in Figure 3.2(a). The distribution of drift is shown along with the values of the most important feature* `avg_ue_downlink_pkts` *(number of average user equipment downlink packets).*

quarantine period of CatBoost in blue line of Figure 3.2(a)), LEAF aims to determine the extent of error present in specific regions of the feature space.

LEAplot. We have developed LEAplot to help visualize the local error components and compare them across different data subsets. We start by identifying the most sensitive feature through permutation Breiman [2001] using Autogluon AI [accessed July, 2021]. For each data subset, we then group samples based on the value of the important feature into N bins. Next, NRMSE is computed for samples within each bin. This technique yields an approximation of local errors over the range of the most sensitive feature(s) for a certain model and corresponding dataset.

LEAplot also characterizes the extent of drift and provides the foundation of the LEAF framework. Figure 4.2 illustrates local error decomposition using an LEAplot for the CatBoost model. `avg_ue_downlink_pkts` (number of average user equipment downlink packets) is the most important feature for this model. When its value is below 0.75e9, the errors in "During

((a)) *LEAgram (before mitigation).*   ((b)) *LEAgram (after TRAO mitigation).*
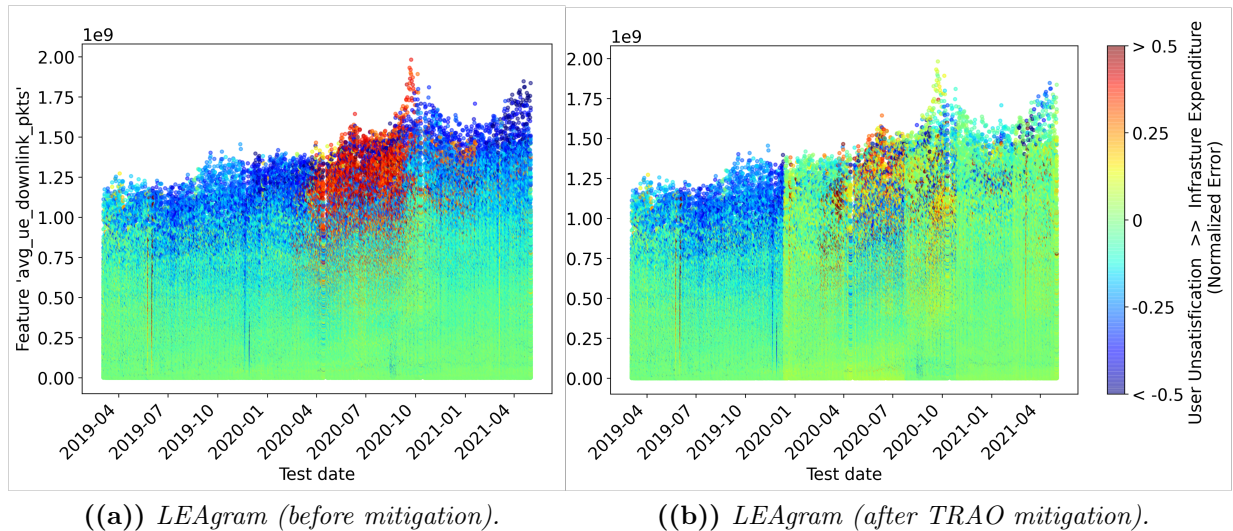
Figure 4.3: *The LEAgrams that decompose NRMSE time-series, where over- and under-estimation of models are different concerns of operators and they map back to extra infrastructure expenditure and user unsatisfaction. (a) illustrates the decompostion of CatBoost NRMSE shown in Figure 3.2(a). (b) shows errors of mitigated CatBoost model if TRAO (see Table 4.1) is in use.*

COVID" test set are more than 10x those of training set and nearly 1.5x of those in the "Before COVID" test set between 0.5e9 and 0.75e9. Drift is worse above 0.75e9 for "During COVID" test set.

LEAgram. LEAgram augments LEAplot by incorporating temporal information. It divides the test set by time interval (*e.g.*, date) and assigns samples from those divided datasets into N bins, based on the value of the most important feature with regard to drift. NRMSE is computed within each bin. When N is greater than or equal to the number of samples on each time interval, NRMSE is equivalent to Normalized Mean Absolute Error (NMAE). LEAgram shows the error for individual samples in the entire test set, arranged temporally.

However, in operational cellular networks, overestimation leads to different outcomes compared with underestimation when modeling for capacity planning purposes. For example, overestimation could result in unnecessary infrastructure expenditure, while underestimation can lead to user dissatisfaction as infrastructure is not augmented when it should be. Given these practical issues, we improve the NRMSE-based approach to create LEAgrams with two

modifications: first, we expand the choice of bin N to be always greater than the number of samples to see individual sample effects; second, we change the metric to Normalized Error (NE) to preserve the sign. Figure 4.3(a) shows the LEAgram which explains the performance of CatBoost in Figure 3.2(a). From March 15, 2020 to November 1, 2020, when the value of feature `avg_ue_downlink_pkts` is above 0.75e9, large positive errors occur, implying overestimation. If the operator were to base decisions on the output of this model, they may unnecessarily add to infrastructure. The model also has negative prediction errors above a value of 1e9 for feature `avg_ue_downlink_pkts`, which could lead to user dissatisfaction.

4.4   Mitigation

The mitigation module is based on the idea of informed adaptation. Given the information from LEAF's drift explainer, we can understand the feature that most contributes to drift, the amount of drift at each range of values, and the over/underestimation status of each sample. We develop 3 families of mitigation strategies using this information.

Resample data. When drift is detected, the latest samples are provided to the explanation module to derive the distribution of errors along with the values of the most important feature. This error distribution shows the area of the dataset that is inaccurately trained or under-training. Thus, we sample from the existing collected dataset (including the latest drifting samples) based on weights provided by this error distribution. In order to compensate for the errors in the original model. We explicitly use non-linear (cubic) resampling Lahiri and Lahiri [2003] to determine the weights.

Augment data. While resampling is an effective approach, there could be no sample in the existing dataset that is within the range that we seek to mitigate. Therefore, we must augment Cubuk et al. [2019], Zhong et al. [2020] the previous dataset. We observe the ground truth of models, then characterize their distribution of outputs O when drift occurs. From the error distribution E derived from LEAF, we linearly resample instances from the past.

23

Then new data outputs of those samples are created by adding perturbation based on O and random gaussian noise.

Ensemble model. The intuition behind the ensemble Dietterich et al. [2002], Krawczyk et al. [2017] method is the locality of drifts we observed. As shown in Figure 4.3(a), when the value of feature `avg_ue_downlink_pkts` is above 0.75e9, large positive errors occur during the COVID-19 quarantine period. Thus, a model targeting the specific temporal region can be more effective. By ensembling multiple targeted retraining models, we can preserve performance in regions of the dataset where existing models are performing well, while improving in other regions. We split the training and testing set into quartiles based on values of the most important feature. Then apply resampling or augmentation on each quartile.

4.5   Putting it Together: LEAF End-to-end Mitigation Schemes

In this section, we put together the LEAF components organically and define end-to-end schemes. When drift detection is combined with explanation and individual mitigation methods, the mitigator is often triggered more than once. In this case, another design choice remains: how do we manage the data samples that are subsequently resampled or augmented by LEAF? We denote the original model as M_0 and the subsequent drift triggered models as M_i. For instance, if we use one of the mitigation methods and derive a resampled or augmented data subset S_i for the ith mitigation. Because S_i is derived from the error distribution of the previous dataset, we must either append the latest S_i only to the original training set (AO), or continuously append sample S_i to S_{i-1} (CA). The baseline Naïve retraining, on the other hand, is always retrained on the latest dataset (AL). M_i substitutes M_{i-1} and drift is detected upon the new NRMSE series after switching the model.

Table 4.1 summarizes the abbreviations of end-to-end mitigation schemes. They are named after the variations from each part of LEAF. Naïve retraining (no LEAF information is used) and TLAL (Triggered Learning Always using Latest samples, only information from the detector is used) are two baselines we compare LEAF schemes with. We evaluate these

24

Scheme	Detector	Explainer	Mitigator
Naïve	No - Naïve	No	Latest data
TLAL			
TRAO	Yes - Triggered	Yes	Resample data
TRCA			
TAAO			Augment data
TACA			
TEAO			Ensemble model
TECA			

Table 4.1: *Abbreviations of LEAF end-to-end mitigation approaches. Naïve and TLAL are two baselines in Section 5. AL: Always use Latest data; AO: Always append data to Original training set; CA: Continuously Append data to the previous training set.*

approaches comprehensively in Section 5.

As an example of the end-to-end schemes, Figure 4.3(b) shows the mitigated effects of TRAO on CatBoost. Once the mitigation is triggered successfully at the beginning of December 2019, and also after October 2020, the under-estimation above 1e9 is eased. Between March 15th, 2020 and October 1st, 2020, the extreme over-predictions above 0.75e9 are largely reduced. Overall, Figure 4.3(b) shows a 18.16% of reduction on $\Delta\overline{NRMSE}$ compared to Figure 4.3(a), which demonstrates the effectiveness of these end-to-end mitigation strategies.

CHAPTER 5

EVALUATION

In this chapter, we evaluate how the LEAF framework drift mitigation compares to baseline techniques, *i.e.*, no retraining, naïve retraining and TLAL. We present how different mitigation schemes improve end-to-end model performance over the entirety of the dataset first described in Section 2.1. Our results demonstrate that LEAF consistently outperforms periodic retraining while requiring orders of magnitude fewer retraining operations.

5.1 End-to-End Mitigation Results

We implement the LEAF detection, explanation, and mitigation pipeline and evaluate the retraining schemes described in Section 4.5. We evaluate how pipelines perform across four models (one per model family) and the five target KPIs (defined in Table 2.2). We compare the average error over the duration of the dataset against the naïve retraining scheme and TLAL, and study whether the mitigation improves predictions over time compared to a static model.

Mitigation effectiveness of LEAF vs. baselines. We study how different mitigation techniques perform across models and KPIs. In particular, we aim to understand (1) whether LEAF's mitigation schemes outperform standard baselines, and (2) which schemes work best depending on model and KPI characteristics. We use naïve retraining Kantchelian et al. [2013], Thomas et al. [2011] and "Triggered Learning Always using Latest data (TLAL)" as the baselines. The former retrains the model every 90 days (because a retrain window of 90 days have 12 retrains and LEAF schemes are triggered to retrain 5 to 10 times, which makes the comparisons fair), while the latter uses drift detection to retrain on the latest data.

In Table 5.1, we present a summary of the $\Delta\overline{NRMSE}$ (the more negative, the better). We show the best schemes for each model and KPI in gray, with schemes from LEAF consistently outperforming the two baselines. In particular, for the LSTM we find that our schemes are

| Model	Schemes	$\Delta\overline{NRMSE}$ of Target KPIs				
		DVol	REst	DTP	CDR	PU
CBoost	Naïve	-0.45%	0.08%	-14.99%	-3.88%	-1.09%
	TLAL	-12.10%	-9.90%	-7.92%	44.06%	-5.69%
	TRAO	-18.16%	-18.78%	-11.43%	-7.85%	-5.77%
XTrees	Naïve	4.88%	-2.15%	-13.24%	2.44%	-4.33%
	TLAL	-12.01%	-11.58%	-9.66%	43.36%	-8.76%
	TRCA	-14.58%	-21.37%	-9.29%	-3.23%	-5.47%
KNDist	Naïve	0.57%	-7.35%	-6.72%	4.96%	12.32%
	TLAL	5.15%	-7.67%	2.73%	9.83%	0.76%
	TAAO	-1.52%	-16.67%	-4.71%	1.63%	-4.26%
LSTM	Naïve	68.46%	87.19%	-34.16%	1.71%	54.11%
	TLAL	39.09%	53.09%	25.24%	90.83%	17.09%
	TEAO	-11.82%	17.87%	-35.90%	5.09%	13.02%

Table 5.1: *Effectiveness of different mitigation schemes measured in $\Delta\overline{NRMSE}$ (the lower the better). The most effective LEAF schemes of each model are presented with two baselines. We include representative models from different model families over a variety of KPIs. Based on the amount of retrains, we choose naïve retraining every 90 days as one baseline. The full tables are in Appendix B.*

drastically better at reducing the NRMSE. The KPI for which naive retraining works best across models is DTP (Downlink Throughput) due to its low coefficient of variance and lack of abrupt drift (Figure 3.1(b)). Failures to mitigate drift (positive $\Delta\overline{NRMSE}$) for naïve retraining across multiple models and KPIs echo with our findings in Section 3.3. Detailed results for all schemes can be found in Appendix B.

Mitigation effectiveness vs. static models. We study how LEAF's mitigation schemes improve model performance over the duration of our dataset. Figure 5.1(a) shows the daily NRMSE evolution over time for the CatBoost model applied to downlink volume prediction. We show results for the static model and TRAO, the best performing mitigation scheme for the model. Compared to the static model, we observe that the LEAF mitigation consistently improves model performance (in terms of NRMSE)throughout the duration of the experiment. We observe that mitigation promptly occurs once drift is detected. This is particularly evident for sudden changes in the NRMSE distribution (*e.g.*, around January and April 2020). Throughout the time series, the NRMSE of TRAO for CatBoost never surpasses 0.125 (with
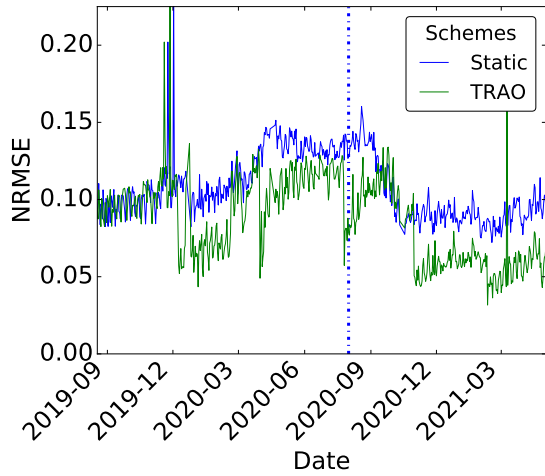
((a)) *Daily NRMSE. August 1, 2020 is labeled with blue dashed line.*
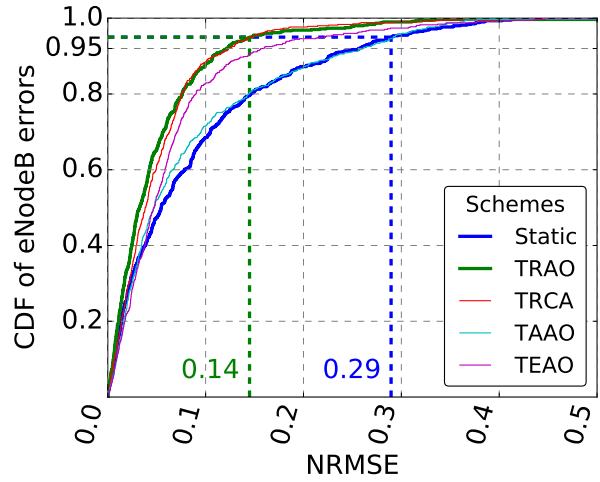
((b)) *CDF of errors on August 1, 2020.*

Figure 5.1: *CatBoost model performance for downlink volume predictions before and after mitigation under different mitigation schemes.*

the exception of spikes due to data errors) and goes as low as 0.04. This ensures a bounded error distribution and decreases the average error by 18.16%. We observe similar behavior for all models and KPIs (Figures 5.2, 5.3, and 5.4).
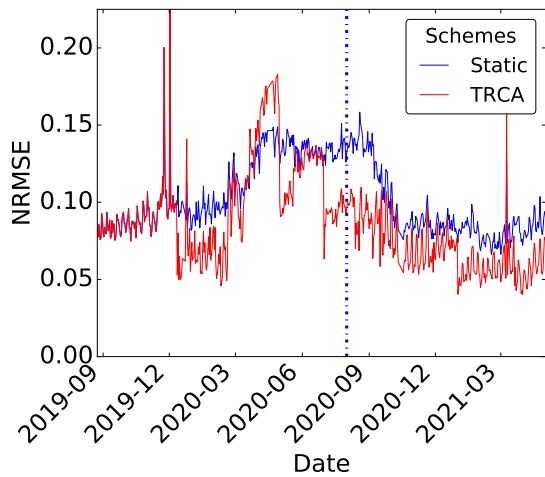
((a)) *Daily NRMSE.*
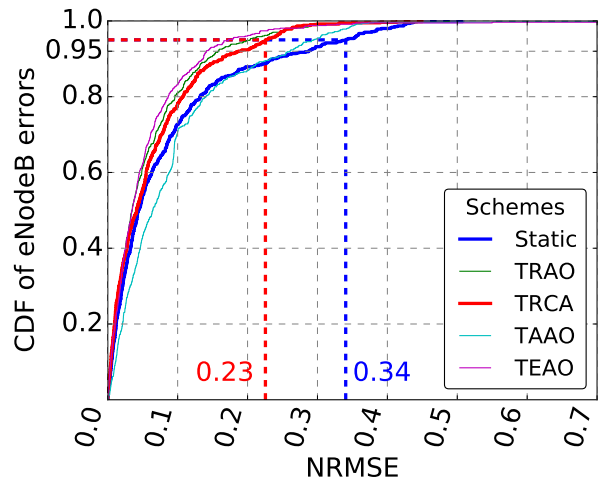
((b)) *CDF of errors on August 1st, 2020.*

Figure 5.2: *ExtraTreesMSE model performance for downlink volume predictions before and after mitigation under different mitigation schemes.*
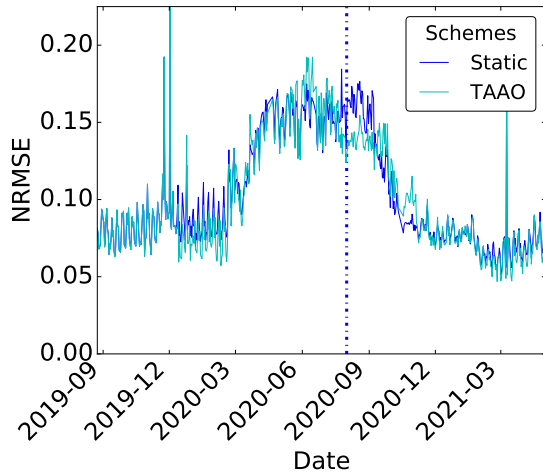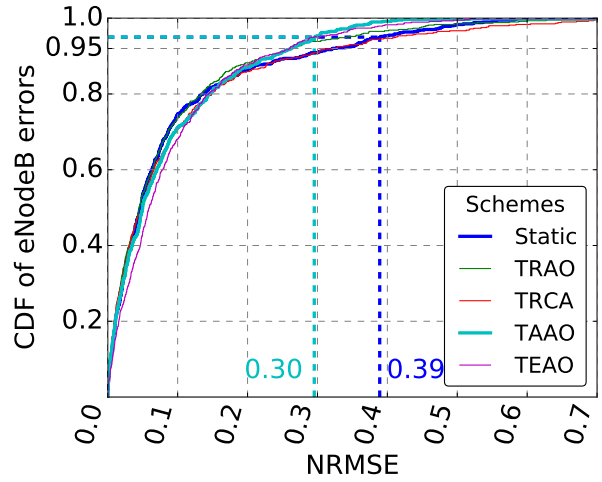
((a)) *Daily NRMSE.*

((b)) *CDF of errors on August 1st, 2020.*

Figure 5.3: *KNeighborsDist model performance for downlink volume predictions before and after mitigation under different mitigation schemes.*
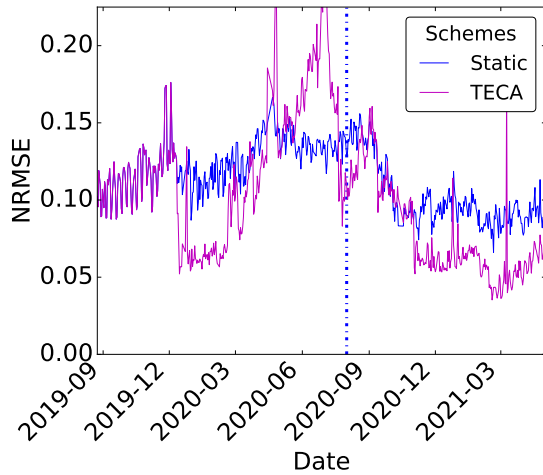
((a)) *Daily NRMSE.*
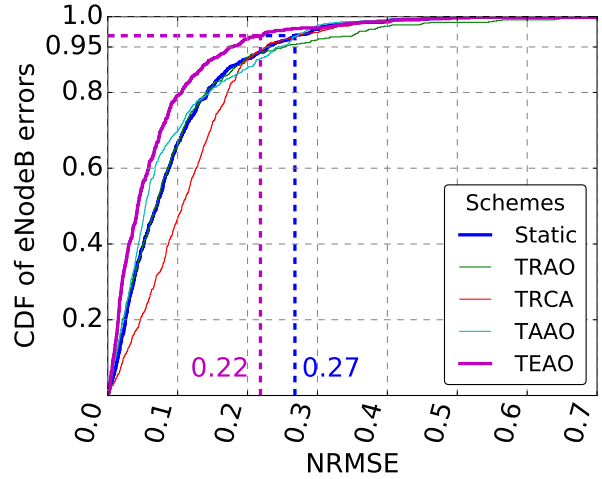
((b)) *CDF of errors on August 1st, 2020.*

Figure 5.4: *LSTM model performance for downlink volume predictions before and after mitigation under different mitigation schemes.*

To better understand the effectiveness of different mitigation schemes, we study their relative average improvement. Figure 5.1(b) shows the CDF of errors for CatBoost across schemes and all eNodeBs on August 1, 2020. We observe that the 95% NRMSE of CatBoost is effectively reduced from 0.29 for the static to 0.14 for TRAO. There is however, a variation

29

in performance across different schemes, which we study in the next section for varying models and KPIs.

5.2   Sensitivity Analysis

In this section, we discuss the impact of the choice of KPI and model on our mitigation scheme.

Different KPIs. We find that different KPIs, depending on the characteristics of their NMRSE time-series, are mitigated more or less effectively. For example, PU has the highest NMRSE among KPIs across models, and is among the two most challenging KPIs to mitigate, along with CDR. It is intuitive that PU has the highest NMRSE since it reflects a *peak* value, which has inherent high variability. CDR has a low baseline NMRSE, but is difficult to mitigate since its variation is of very high frequency, and would possibly require daily mitigation to be effective. We validate this with the coefficient of variance, shown in Table 2.2, where CDR and PU clearly have the highest coefficients of variation of 2.48 and 1.76, respectively. These KPIs, when mitigated by a method such as TLAL, actually lead to a large increase in NMRSE.

Different Models. As expected, we find that each model responds differently to LEAF's mitigation schemes. The targeted retraining schemes outperform the baselines across most combinations of models and KPIs, but as expected, models vary in their reaction to the choice of which data is used for retraining and how this data is added to the training set.

For different models, resampling is the best for bagging and boosting methods (CatBoost and XTrees); augmentation for distance based approaches; and ensembling for LSTM. We validate this by further testing on other boosting (LightGBM), bagging (Random Forest) and distance-based (KNeighborsUnif) algorithms, and this general pattern holds. These observations are rooted back in the capacity of each model and how they generalize. For instance, LSTMs can make most effective use of the ensembling strategy because they learn good features that are effective when using the smaller amounts of data available to each

model in the ensemble. The extra data available in the resampling and augmentation methods helps all the other models.

In terms of data-appending schemes, we observe that AO is better for CatBoost, KNeighborsDist and LSTM, and CA is better for Extra Trees. Note that since the resampled or augmented data are selected based on the weights generated from the error distribution of the previous model, continuously absorbing "biased" data could make mitigation less effective. For example, KNeighbors prefers AO, since with CA, the density of nearby points with incorrect outputs is too high, leading to incorrect predictions. Overall, however, for models with sufficient capacity like LSTMs, either CA or AO can be the most effective depending on the particular KPI, so we encourage model trainers to try both approaches.

Correlated Feature Subsets. The LEAplot and LEAgram do not show the effect of a single feature, but rather a joint effect of a correlated feature group. We find that the distance of LEA distributions between the training set and any given test set are consistent across correlated features. This indicates that these features naturally share the same error distributions and therefore can be grouped together. The feature subset is identified by the similarity of their error distributions measured by Wasserstein distance (WD). WD is a metric that measures the distance between two distributions Ramdas et al. [2017]. It can be interpreted as the minimum amount of "work" required to transform one distribution into another. We can thus identify these joint effects. For instance, to forecast downlink volume, the feature group including `avg_ue_downlink_pkts` also contains `avg_ue_uplink_pkts` (number of average user equipment uplink packets), `ul_pdcp_sdu_received_withinpdb` (the uplink Packet Data Convergence Protocol Service Data Unit received within Packet Delay Budget), `erab_rels_normal_enb` (E-UTRAN Radio Access Bearer normal release for this eNodeB)[1]. These are ranked by using permutation-based feature importance; features with a zero importance score are excluded.

---

1. As a sanity check, we verified with the network operator that these features naturally correlate with one another.

CHAPTER 6

RELATED WORK

Concept drift is a pervasive phenomenon in machine learning that has been well-studied in various contexts and scenarios. We survey various aspects of concept drift, within networking and more broadly, as well as previous work in drift detection, explainable AI, and drift adaptation—all of which are increasingly active research areas.

Drift in network management. Machine learning has been applied to many networking problems, including anomaly detection Shon and Moon [2007], Shon et al. [2005], intrusion detection Sinclair et al. [1999], Dong and Wang [2016], cognitive network management Ayoubi et al. [2018], and network forecasting Mei et al. [2020], Chinchali et al. [2018]. Few studies have explicitly explored concept drift in the networking context. It is, however, well-known that network traffic is inherently variable over time. For example, mobile networks exhibit different patterns of activity based on time of day, day of week, and location in Rome, Italy Reades et al. [2009]. Features such as latency can also drift over long periods of time Nikravesh et al. [2014]. Moreover, the COVID-19 pandemic has provided a unique example of sudden drift and its effects on network traffic patterns, network usage, and resulting model accuracy Liu et al. [2021], Feldmann et al. [2020], Lutu et al. [2020], Comcast [accessed October, 2020], McKeay [accessed October, 2020]. Sommer and Paxson articulate that applying machine learning to anomaly detection is fundamentally difficult in large-scale operational networks Sommer and Paxson [2010] due to these (and other) challenges. All these factors contribute to concept drift in predictive models, yet, to our knowledge, this paper is the first to explore the effects of these types of changes on model accuracy for network management tasks.

Drift in other related fields. Concept drift has also been studied across other real-world contexts. For example, spam detection faces drift challenges, as spammers may actively change the underlying distribution of their messages Fdez-Riverola et al. [2007]. Recommender systems are another context where drift occurs: user side-effects such as preferences and item

side-effects such as popularity both change over time Gama et al. [2014], Lu et al. [2015]. Similar situations occur in monitoring systems Pechenizkiy et al. [2010], fraud Žliobaitė [2010a], and malware detection Jordaney et al. [2017], Barbero et al. [2020], Yang et al. [2021].

Drift characterization and detection. Concept drift has been characterized both quantitatively based on probability distribution Webb et al. [2016], Gama et al. [2014], Lu et al. [2018], as well as based on drift subject, frequency, transition, recurrence, and magnitude Webb et al. [2016]. Sources of drift are also discussed using data distribution and decision boundary changes Lu et al. [2018]. A long line of research has focused on detecting concept drift in deployed systems. A significant body of detection methods are based on error rate Lu et al. [2018]. For example, Drift Detection Method (DDM) Gama et al. [2004] provides a warning level and a detection level by detecting changes in online error rate within a landmark time window. Another approach is ADaptive WINdowing (ADWIN) Bifet and Gavalda [2007], which uses two adjustable sliding windows to store older and recent data, and detect drift based on the differences between them. KSWIN Raab et al. [2020] is a recent improvement upon ADWIN which uses KS-test to capture distances between windows. Large scale comparisons across drift detection methods have been conducted Gonçalves Jr et al. [2014], Barros and Santos [2018], Raab et al. [2020] showing the effectiveness of DDM and KSWIN. LEAF applies KSWIN to detect drift a large cellular network. LEAF's contribution is not to develop a new drift detection method; rather, the contribution of this work is in developing new explanation and mitigation techniques for cellular networks; integrating detection, explanation, and mitigation into a unified end-to-end framework; and evaluating the utility of various end-to-end mitigation strategies.

Explainable AI and drift explanation. Explainable AI has recently attempted to address the challenge of black-box model interpretation. Global model-agnostic methods, such as Partial Dependence Plot (PDP) Friedman [2001] and Accumulated Local Effects (ALE) plot Apley and Zhu [2020], Arzani et al. [2021], provide visual clues on the effect of different

features and set of features on the prediction accuracy of black-box models. LEAF's LEAplot and LEAgram are inspired by PDP and ALE, but extend these techniques by (1) showing errors instead of effects; (2) identifying the correlated sets of features that contribute to drift (not just individual features), and (3) helping to visualize how drift evolves over time in an operational setting. Local agnostic methods, such as LIME Ribeiro et al. [2016], Alvarez-Melis and Jaakkola [2018] and LEMNA Guo et al. [2018], substitute a local model region with an interpretable one, enabling targeted patching to improve accuracy. Recent research has developed techniques to explain concept drift in the context of malware detection. Transcend Jordaney et al. [2017] uses statistical comparison of samples to identify model decay thresholds (*i.e.*, decision boundary-based explanation). CADE Yang et al. [2021] uses contrastive learning to develop a distance-based explanation to find the feature sets that have the largest distance changes towards the centroid in the latent space. Unfortunately, both explanation approaches only apply to classification problems, and do not apply to prediction in general, such as the regression-based prediction problems we study in this paper. To our knowledge, LEAF is the first framework to explain concept drift for regression-based prediction problems for network management tasks.

Drift mitigation. Adapting a model to migitate concept drift is also a well-explored area Gama et al. [2014], Lu et al. [2018]. Adaptation can be based on retraining or model ensemble. Retraining-based approaches often require complex data management and significant storage and memory. Online learning algorithms such as VFDT Domingos and Hulten [2000] keep the latest instance that adapts to slow changes. Paired Learners Bach and Maloof [2008] use one stable learner to learn on old data and another reactive learner to learn from the latest data. DELM Xu and Wang [2017] adaptively adjusts the number of hidden layer nodes to combat drift during retraining. Ensemble methods have also been used for model adaptation Krawczyk et al. [2017]. Dynamic Weighted Majority Kolter and Maloof [2007] deals with non-stationary streams by allocating weights on different models. Accuracy

Updated Ensemble (AUE2) Brzeziński and Stefanowski [2011], Brzezinski and Stefanowski [2013] incrementally updates sub-models on a small portion of data to adapt to drift. LEAF is inspired by several of these approaches, but focuses more heavily on identifying features that lead to concept drift, and performing mitigation at scale.

CHAPTER 7

CONCLUSION

An important aspect of the applicability of machine learning models to networking tasks in practice is that they continue to perform well over time, in a variety of settings—and help operators understand why they don't perform well when they don't. A significant contributor to degraded model accuracy in practice is concept drift, which can occur periodically, suddenly, or gradually over time. Although this phenomenon has been explored in other contexts, it has received limited attention in the networking domain. And yet, if machine learning models are to be applied in practice, concept drift needs to be detected, explained, and mitigated. Unfortunately, as this paper has also demonstrated, the simple approach of periodically retraining a model is both extremely costly and can in some cases even reduce prediction accuracy.

To address this critical problem, this paper has developed, presented, and evaluated LEAF, a framework to detect, explain, and mitigate drift for machine learning models applied to networks. The LEAF framework applies and extends several well-established techniques in machine learning and explainable AI for drift detection, explanation, and mitigation to the networking context; integrates them into an end-to-end drift mitigation system; and evaluates the framework for network forecasting problems in a large metropolitan area for one of the largest cellular networks in the United States. To our knowledge, the LEAF framework is the first approach—in any setting—to explain drift in regression models. LEAF also helps operators interpret error metrics in an operational context (e.g., to determine whether target predictions are under- or over-estimated).

Our results based on more than three years of KPI data from this network show that LEAF consistently outperforms periodic retraining while reducing costly retraining operations by as much as an order of magnitude. Because of LEAF's effectiveness for concept drift mitigation in this context, the cellular carrier is now integrating the framework into its

operational forecasting and provisioning processes.

We believe that the LEAF framework can be applied beyond cellular networks, to other network management problems that use black-box models for regression-based prediction, from demand forecasting to application performance prediction in ISPs and service provider networks. Yet, these hypotheses are yet to be explored and thus make excellent avenue for future work. Another caveat is that the evaluation LEAF to date has been conducted on fully labeled datasets; thus, a promising direction could be improve LEAF to cope with semi-supervised or unsupervised models with partial or no labels.

BIBLIOGRAPHY

3GPP. *LTE standard, 3GPP TS 32.425 version 9.5.0 Release 9*, accessed July, 2021. `https://www.etsi.org/deliver/etsi_ts/132400_132499/132425/09.05.00_60/ts_132425v090500p.pdf`.

AutoGluon AI. *AutoGluon: AutoML for Text, Image, and Tabular Data*, accessed July, 2021. `https://auto.gluon.ai/stable/index.html`.

David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

Amazon. *Evaluating model accuracy: Regression - Amazon machine learning developer guide*, accessed August, 2021. `https://docs.aws.amazon.com/machine-learning/latest/dg/regression.html`.

Daniel W Apley and Jingyu Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(4):1059–1086, 2020.

Behnaz Arzani, Kevin Hsieh, and Haoxian Chen. Interpretable feedback for automl and a proposal for domain-customized automl for networking. In *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*, pages 53–60, 2021.

AT&T. *COVID-19: Our Response*, accessed October, 2020. `https://about.att.com/pages/COVID-19.html`.

Sara Ayoubi, Noura Limam, Mohammad A Salahuddin, Nashid Shahriar, Raouf Boutaba, Felipe Estrada-Solano, and Oscar M Caicedo. Machine learning for cognitive network management. *IEEE Communications Magazine*, 56(1):158–165, 2018.

Stephen H Bach and Marcus A Maloof. Paired learners for concept drift. In *2008 Eighth IEEE International Conference on Data Mining*, pages 23–32. IEEE, 2008.

Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. Transcending transcend: Revisiting malware classification with conformal evaluation. *arXiv preprint arXiv:2010.03856*, 2020.

Roberto Souto Maior Barros and Silas Garrido T Carvalho Santos. A large-scale comparison of concept drift detectors. *Information Sciences*, 451:348–370, 2018.

Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Dariusz Brzeziński and Jerzy Stefanowski. Accuracy updated ensemble for data streams with concept drift. In *International conference on hybrid artificial intelligence systems*, pages 155–163. Springer, 2011.

Dariusz Brzezinski and Jerzy Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94, 2013.

Timm Böttger, Ghida Ibrahim, and Ben Vallis. How the internet reacted to covid-19 – a perspective from facebook's edge network. In *ACM IMC*, 2020.

Massimo Candela, Valerio Luconi, and Alessio Vecchio. Impact of the covid-19 pandemic on the internet latency: a large-scale study. *arXiv preprint arXiv:2005.06127*, 2020.

Sandeep Chinchali, Pan Hu, Tianshu Chu, Manu Sharma, Manu Bansal, Rakesh Misra, Marco Pavone, and Sachin Katti. Cellular network traffic scheduling with deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Kenjiro Cho, Cristel Pelsser, Randy Bush, and Youngjoon Won. The japan earthquake: the impact on traffic and routing observed by a local isp. In *Proceedings of the Special Workshop on Internet and Disasters*, pages 1–8, 2011.

Comcast. *COVID-19 Network Update*, accessed October, 2020. `https://corporate.comcast.com/covid-19/network/may-20-2020`.

CTIA. *The Wireless Industry Responds to COVID-19: Network Performance*, accessed October, 2020. `https://www.ctia.org/homepage/covid-19#network-performance`.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of country-wide internet outages caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 1–18, 2011.

Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125, 2002.

Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

Bo Dong and Xue Wang. Comparison deep learning method to traditional methods using for network intrusion detection. In *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, pages 581–585. IEEE, 2016.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

fast.ai. *FastAI v1.0, documentation*, accessed August, 2021. `https://fastai1.fast.ai`.

Thomas Favale, Francesca Soro, Martino Trevisan, Idilio Drago, and Marco Mellia. Campus traffic and e-learning during covid-19 pandemic. *Computer Networks*, 2020.

FCC. *Measuring Broadband Raw Data Releases*, accessed October, 2020a. `https://www.fcc.gov/oet/mba/raw-data-releases`.

FCC. *Keep Americans Connected*, accessed October, 2020b. `https://www.fcc.gov/keep-americans-connected`.

Florentino Fdez-Riverola, Eva Lorenzo Iglesias, Fernando Díaz, José Ramon Méndez, and Juan M Corchado. Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications*, 33(1):36–48, 2007.

Nick Feamster. Revealing utilization at internet interconnection points. TPRC, 2016.

Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapidor, Narseo Vallina-Rodriguez, et al. The lockdown effect: Implications of the covid-19 pandemic on internet traffic. In *Internet Measurement Conference (IMC '20)*, 2020.

Wikimedia Foundation. *U.S. state and local government responses to the COVID-19 pandemic*, accessed October, 2020. `https://en.wikipedia.org/wiki/U.S._state_and_local_government_responses_to_the_COVID-19_pandemic`.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Futuriom. *Verizon Applies Machine Learning to Operations*, accessed August, 2021. `https://www.futuriom.com/articles/news/verizon-applies-machine-learning-to-operations/2018/08`.

Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

Paulo M Gonçalves Jr, Silas GT de Carvalho Santos, Roberto SM Barros, and Davi CL Vieira. A comparative study on concept drift detectors. *Expert Systems with Applications*, 41(18):8144–8156, 2014.

Google. *Keeping our network infrastructure strong amid COVID-19*, accessed October, 2020. `https://blog.google/inside-google/infrastructure/keeping-our-network-infrastructure-strong-amid-covid-19/`.

Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 364–379, 2018.

Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

John Heidemann, Lin Quan, and Yuri Pradkin. *A preliminary analysis of network outages during hurricane sandy*. University of Southern California, Information Sciences Institute, 2012.

Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 625–642, 2017.

Alex Kantchelian, Sadia Afroz, Ling Huang, Aylin Caliskan Islam, Brad Miller, Michael Carl Tschantz, Rachel Greenstadt, Anthony D Joseph, and JD Tygar. Approaches to adversarial drift. In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, pages 99–110, 2013.

Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *ICML*, pages 487–494, 2000.

J Zico Kolter and Marcus A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

Bartosz Krawczyk, Leandro L Minku, Joao Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

SK Lahiri and SN Lahiri. *Resampling methods for dependent data*. Springer Science & Business Media, 2003.

Shinan Liu, Paul Schmitt, Francesco Bronzino, and Nick Feamster. Characterizing service provider response to the covid-19 pandemic in the united states. In *PAM 2021-Passive and Active Measurement Conference*, 2021.

Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32, 2015.

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12): 2346–2363, 2018.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

Andra Lutu, Diego Perino, Marcelo Bagnulo, Enrique Frias-Martinez, and Javad Khangosstar. A characterization of the covid-19 pandemic impact on a mobile network operator traffic. In *Proceedings of the ACM Internet Measurement Conference*, pages 19–33, 2020.

Martin McKeay. *Parts of a Whole: Effect of COVID-19 on US Internet Traffic*, accessed October, 2020. `https://blogs.akamai.com/sitr/2020/04/parts-of-a-whole-effect-of-covid-19-on-us-internet-traffic.html`.

Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifan Han, Feng Li, and Jin Li. Realtime mobile bandwidth prediction using lstm neural network and bayesian fusion. *Computer Networks*, 182:107515, 2020.

Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

Robert Nau. Notes on linear regression analysis. *Fuqua School of Business, Duke University Retrieved from: $https://people.duke.edu/~rnau/Notes\_on\_linear\_regression\_analysis--Robert\_Nau.pdf$*, 2014.

NCTA. *COVID-19: How Cable's Internet Networks Are Performing: METRICS, TRENDS & OBSERVATIONS*, accessed October, 2020. `https://www.ncta.com/COVIDdashboard`.

Ashkan Nikravesh, David R Choffnes, Ethan Katz-Bassett, Z Morley Mao, and Matt Welsh. Mobile network performance from user devices: A longitudinal, multidimensional analysis. In *International Conference on Passive and Active Network Measurement*, pages 12–22. Springer, 2014.

Nokia. *Network traffic insights in the time of COVID-19: April 9 update*, accessed October, 2020. `https://www.nokia.com/blog/network-traffic-insights-time-covid-19-april-9-update/`.

OECD. *Keeping the Internet up and running in times of crisis*, accessed October, 2020. `https://www.oecd.org/coronavirus/policy-responses/keeping-the-internet-up-and-running-in-times-of-crisis-4017c4c9/`.

Seongchul Park, Sanghyun Seo, Changhoon Jeong, and Juntae Kim. Network intrusion detection through online transformation of eigenvector reflecting concept drift. In *Proceedings of the First International Conference on Data Science, E-learning and Information Systems*, pages 1–4, 2018.

Craig Partridge, Paul Barford, David D Clark, Scan Donelan, Vern Paxson, Jennifer Rexford, Mary K Vernon, and Jon Eisenberg. The internet under crisis conditions: learning from september 11. *Computer Communication Review*, 33(2):1–8, 2003.

Mykola Pechenizkiy, Jorn Bakker, I Žliobaitė, Andriy Ivannikov, and Tommi Kärkkäinen. Online mass flow prediction in cfb boilers with explicit detection of sudden concept drift. *ACM SIGKDD Explorations Newsletter*, 11(2):109–116, 2010.

Christoph Raab, Moritz Heusinger, and Frank-Michael Schleif. Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416:340–351, 2020.

Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47, 2017.

Jonathan Reades, Francesco Calabrese, and Carlo Ratti. Eigenplaces: analysing cities using the space–time structure of the mobile phone network. *Environment and Planning B: Planning and Design*, 36(5):824–836, 2009.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343. PMLR, 2018.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

RouteViews. *University of Oregon Route Views Project*, accessed October, 2020. `http://www.routeviews.org/routeviews/`.

Sandvine. *COVID-19 Global Internet Trends*, accessed October, 2020. `https://www.sandvine.com/covid-19-trends/`.

Jeffrey C Schlimmer and Richard H Granger. Incremental learning from noisy data. *Machine learning*, 1(3):317–354, 1986.

Taeshik Shon and Jongsub Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799–3821, 2007.

Taeshik Shon, Yongdae Kim, Cheolwon Lee, and Jongsub Moon. A machine learning framework for network anomaly detection using svm and ga. In *Proceedings from the sixth annual IEEE SMC information assurance workshop*, pages 176–183. IEEE, 2005.

Chris Sinclair, Lyn Pierce, and Sara Matzner. An application of machine learning to network intrusion detection. In *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*, pages 371–377. IEEE, 1999.

Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

Kenneth O Stanley. Learning concept drift with a committee of decision trees. *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *2011 IEEE symposium on security and privacy*, pages 447–462. IEEE, 2011.

Maurras Ulbricht Togbe, Yousra Chabchoub, Aliou Boly, Mariam Barry, Raja Chiky, and Maroua Bahri. Anomalies detection using isolation in concept-drifting data streams. *Computers*, 10(1):13, 2021.

Anh Viet Vu, Jack Hughes, Ildiko Pete, Ben Collier, Yi Ting Chua, Ilia Shumailov, and Alice Hutchings. Turning up the dial: the evolution of a cybercrime market through set-up, stable, and covid-19 eras. In *ACM IMC*, 2020.

Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.

Shuliang Xu and Junhong Wang. Dynamic extreme learning machine for data stream classification. *Neurocomputing*, 238:433–449, 2017.

Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. {CADE}: Detecting and explaining concept drift samples for security applications. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

Camellia Zakaria, Amee Trivedi, Michael Chee, Prashant Shenoy, and Rajesh Balan. Analyzing the impact of covid-19 control policies on campus occupancy and mobility via passive wifi sensing. *arXiv preprint arXiv:2005.12050*, 2020.

Gan Zheng, Ioannis Krikidis, Christos Masouros, Stelios Timotheou, Dimitris-Alexandros Toumpakaris, and Zhiguo Ding. Rethinking the role of interference in wireless networks. *IEEE Communications Magazine*, 52(11):152–158, 2014.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.

Indrė Žliobaitė. Adaptive training set formation. 2010a.

Indrė Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010b.

Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, pages 91–114, 2016.

APPENDIX A

CONCEPT DRIFT BEHAVIOR

In this chapter, we report concept drift behavior for models or RRC establishment success (Figure A.1).
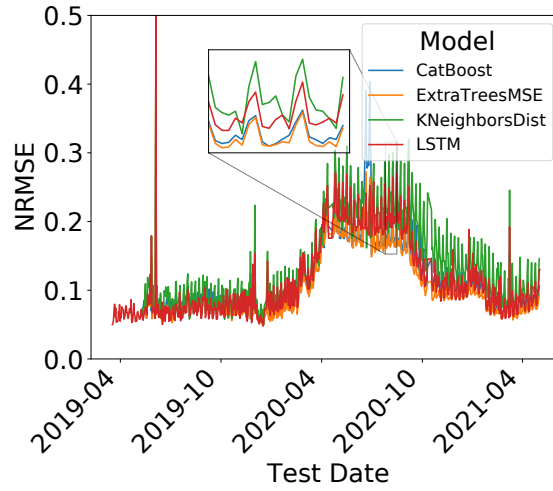

Figure A.1: *Drift of different models for RRC establishment success. Inset figures exhibit a 3-week view (all starting from Sunday) of NRMSE for the box-selected period. It shows similar patterns as the downlink volume with a stronger weekly effects.*

APPENDIX B

END-TO-END DRIFT MITIGATION

In this chapter we report the $\Delta\overline{NRMSE}$ for all LEAF schemes across all models and targeted KPIs (Tables B.1, B.2, B.3, and B.4).

Mitigt.	Change of NRMSE mean ($\Delta\overline{NRMSE}$)				
schemes	DVol	REst	DTP	CDR	PU
Naïve	−0.45%	0.08%	−14.99%	−3.88%	−1.09%
TLAL	−12.10%	−9.90%	−7.92%	44.06%	−5.69%
TRAO	−18.16%	−18.78%	−11.43%	−7.85%	−5.77%
TRCA	−17.25%	−16.85%	−14.55%	13.04%	−7.50%
TAAO	−1.09%	−2.69%	3.93%	3.35%	−2.60%
TACA	33.36%	−0.78%	9.57%	10.94%	4.33%
TEAO	−17.29%	−15.72%	−13.11%	29.26%	−6.71%
TECA	−15.67%	−12.17%	−14.22%	0.32%	−2.08%

Table B.1: *Effectiveness of different mitigation schemes on CatBoost. The best performant schemes for each KPI under different metrics are shaded in gray.*

Mitigt.	Change of NRMSE mean ($\Delta\overline{NRMSE}$)				
schemes	DVol	REst	DTP	CDR	PU
Naïve	4.88%	−2.15%	−13.24%	2.44%	−4.33%
TLAL	−12.01%	−11.58%	−9.66%	43.36%	−8.76%
TRAO	−16.80%	−14.87%	−10.67%	3.27%	−6.75%
TRCA	−14.58%	−21.37%	−9.29%	−3.23%	−5.47%
TAAO	−1.13%	−0.43%	9.24%	2.24%	−2.58%
TACA	35.45%	3.79%	9.76%	4.66%	5.60%
TEAO	−13.72%	−12.23%	−5.02%	12.85%	−6.85%
TECA	−6.77%	−12.72%	−11.64%	−1.07%	−1.04%

Table B.2: *Effectiveness of different mitigation schemes on ExtraTreesMSE. The best performant schemes for each KPI under different metrics are shaded in gray.*

Mitigt.	Change of NRMSE mean ($\Delta\overline{NRMSE}$)				
schemes	DVol	REst	DTP	CDR	PU
Naïve	0.57%	−7.35%	−6.72%	4.96%	12.32%
TLAL	5.15%	−7.67%	2.73%	9.83%	7.76%
TRAO	5.95%	−4.85%	−1.08%	4.65%	2.64%
TRCA	10.70%	−5.46%	−0.30%	55.98%	7.40%
TAAO	−1.52%	−16.67%	−4.71%	1.63%	−4.26%
TACA	7.55%	17.83%	−4.93%	0.88%	5.15%
TEAO	5.12%	−8.86%	−0.79%	3.99%	3.24%
TECA	14.43%	−1.96%	−0.08%	6.22%	10.62%

Table B.3: *Effectiveness of different mitigation schemes on KNeighborsDist. The best performant schemes for each KPI under different metrics are shaded in gray.*

Mitigt.	Change of NRMSE mean ($\Delta\overline{NRMSE}$)				
schemes	DVol	REst	DTP	CDR	PU
Naïve	68.46%	87.19%	−34.16%	1.71%	54.11%
TLAL	39.09%	53.09%	25.24%	90.84%	17.09%
TRAO	19.13%	28.46%	−31.33%	43.89%	11.70%
TRCA	28.26%	41.54%	−29.11%	148.09%	19.02%
TAAO	9.84%	48.12%	−34.62%	−9.93%	12.86%
TACA	10.25%	50.45%	−34.16%	95.96%	7.47%
TEAO	−11.82%	17.87%	−35.90%	5.09%	13.02%
TECA	−1.25%	−8.44%	−40.38%	18.19%	23.92%

Table B.4: *Effectiveness of different mitigation schemes on LSTM. The best performant schemes for each KPI under different metrics are shaded in gray.*