University of Padova

Department of Mathematics Tullio-Levi Civita

Master Thesis in Data Science

Performance Estimation Methods for Diagnosing Model Quality Degradation under Covariate Shift

Supervisor
Bruno Scarpa
University of Padova

Master Candidate
Santiago Víquez Segura

Academic Year

2022-2023

Dedication.

To my mom, for encouraging me to follow the path that I enjoy the most. To Kathu, for telling me that I could. And to Maple, for reminding me to enjoy the little things in life.

Abstract

This study explores the underexplored domain of performance degradation in machine learning models, a common challenge in dynamic data environments. While the academic community often prioritizes the development of new training methods to enhance model performance on benchmark datasets, little attention has been given to sustaining those high levels of performance post-deployment or identifying when performance starts to decade. In a rapidly evolving data landscape, where covariate shifts can significantly impact model performance, understanding and mitigating performance degradation issues becomes essential.

To tackle this issue, this study introduces three comprehensive tests. The Temporal Degradation Test examines how various models perform when trained on different samples of the same dataset, shedding light on degradation patterns. The Continuous Retraining Test simulates a production environment by assessing the impact of continuous model retraining. Finally, the Performance Estimation Test explores the potential of performance estimation methods, such as Direct Loss Estimation (DLE), to identify degradation without ground truth data. Our findings reveal diverse degradation patterns influenced by machine learning methodologies, with continuous retraining offering partial relief but not complete resolution. Performance estimation methods emerge as vital early warning systems, enabling timely interventions to maintain model efficacy.

Contents

Listing of figures

Listing of tables

Listing of acronyms

DLE	Direct Loss Estimation
EDA	Exploratory Data Analysis
MAPE	Mean Average Percentage Error
TLC	Taxi & Limousine Commission
LGBM	Light Gradient-Boosting Machine
MLPRegressor	Multilayer Perceptron Regressor

1 Introduction

In the academic world, we often focus on investigating new methods and training procedures to achieve better performance results across different benchmark datasets. However, less has been written about how to maintain those performance levels or how to identify performance degradation after a model has been deployed.

Most of the time, Machine Learning models are trained to make predictions about future data. In those situations, ideally, the training data is a good representation of what the model will see once in production—also referred to as serving data [1].

But in fast-changing environments, the feature's distribution in the serving data might evolve with time. These changes are known as covariate shifts. When the changes are so that the training data is no longer a good representation of the serving data, the model's predictive performance is expected to decrease. This phenomenon is known as model degradation or model performance degradation to be more strict.

In the industry, model performance degradation is hard to spot since; to evaluate the performance of a model on new data, one needs access to ground truth. Most of the time, depending on the type of prediction problem, the ground truth is either delayed, time-consuming to get, or unavailable. Circumstances like this leave most Machine Learning practitioners without any visibility of how well a deployed Machine Learning model is performing on serving data.

Continuous retraining and online learning have been the proposed solutions to keep models up-to-date with their environments. But as mentioned in [2] we still don't have a systematic framework on what should trigger model retraining, how often we should do it, or whether it is needed at all.

In the present study, we implement three different tests to evaluate the predictive performance degradation of Machine Learning models and measures to overcome and identify such changes. Chapter 2 starts by describing the Temporal Degradation Test used to investigate the performance degradation patterns that different models might produce on different training samples of the same dataset—followed by the Continuous Retraining Test, which investigates the effects of a continuous retraining process by setting up a framework that emulates a retraining process of a Machine Learning model in production. And finishes with the Performance Estimation Test used to study how performance estimation methods [3] can help us know if the model's predictive performance has degraded without needing access to ground truth data.

We close the investigation by evaluating what percentage of the previously observed degradation patterns might have been avoided by retraining and estimating the models' predictive performance.

2 Methods

This chapter explains in detail the three main contributions this study brings to the table: the Temporal Degradation, the Continuous Retraining, and the Performance Estimations Tests. These three frameworks are used across all the experiments in Chapter 4, and their implementation code is open to the general public [4].

2.1 Temporal Degradation Test

The temporal degradation test is based on the model aging framework developed in a previous work by Vela et al. in 2022, as referenced in [2]. They define the "age" of a Machine Learning model as the time passed since the model was last trained. We designed a similar framework to investigate temporal degradation patterns emulating a typical Machine Learning model training and deployment phases. We can understand the Temporal Degradation Test as an iterative process, starting from a dataset previously arranged in chronological order based on when each observation was gathered. Next, we randomly sample three subsets from this dataset.

- The first subset will be the training set.
- The second subset, which is adjacent to the training set, will be used as the test set.
- Finally, a third subset is randomly selected from any point in the future. This set will serve as the production set, which will play the role of serving data.

After sampling these three subsets, we evaluate different hyperparameters by performing cross-validation in a TimeSeriesSplit [5] manner across the training set. After selecting the best hyperparameters, we fit the model on all the training sets and test on the test sets to collect its test scores. If the test scores are acceptable, meaning they are smaller than an arbitrary threshold, set up at the moment of the experiment (e.g., Test MAPE < 10%), we consider that the model generalizes well on test data and flag it as a good model since it has a good initial fit.

This way, we study only the performance degradation of *good models*. Checking if a model is good is fundamental to the study since analyzing the degradation of models with a poor initial fit is not worth it and can bias the results. After this procedure is complete, we run an inference step on the production subset and collect its production performance errors. In total, we run a large number (N=3000) of these types of sampling, training, testing, and production simulations. Collect all the production performance errors from the *good models* and plot what we call a model aging plot, where we study the performance degradation patterns.

Figure 2.1 exemplifies visually the temporal degradation test. The long grey box represents the original dataset, and the green, yellow, and blue boxes represent the training, test, and production set randomly sampled from the original dataset. We see how the training and test sets are always adjacent to each other. The gap between the test set's end and the production set's beginning is defined as the model age. Simulations 2, 3, ... N in the same figure are different that can occur during a single Temporal Degradation Test. In summary, a single Temporal Degradation Test uses a dataset that is partitioned differently on each simulation and a model type that is defined prior to the experiment. So, each degradation test allows us to study the model aging patterns between a model-dataset pair.

As we will discuss in Chapter 4, for all the experiments with the methods explained here, we will use the two datasets explained in Section 3 and four Machine Learning approaches (linear models, ensembles, boosted models, and neural networks) intending to explore the degradation pattern from different mathematical approaches when building Machine Learning models.

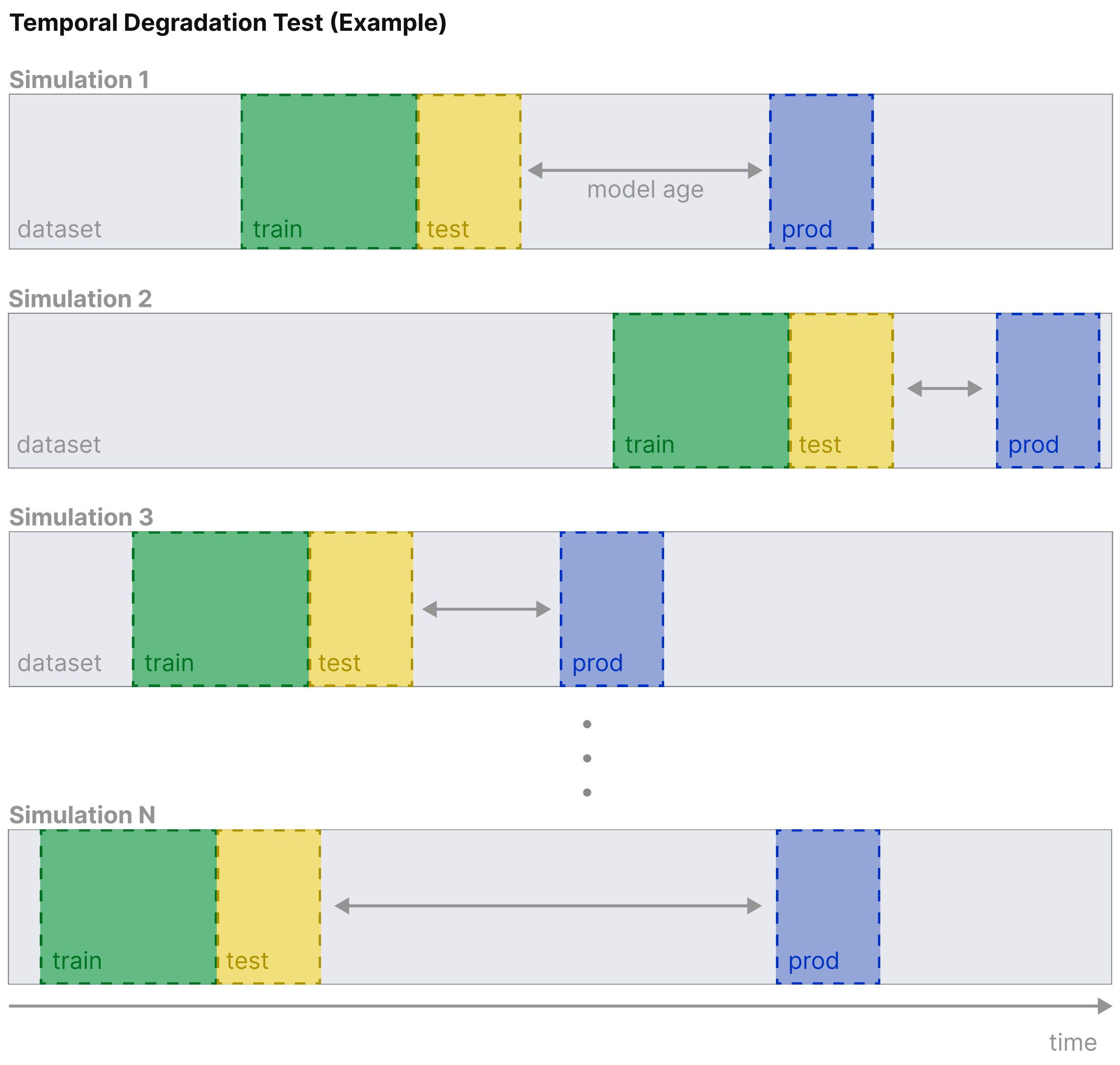

Figure 2.1: Temporal Degradation Experiment.

2.2 Continuous Retraining Test

To combat the performance degradation issue, companies tend to re-fit/retrain their models occasionally with more recent data to build a more robust model that hopefully does not degrade in the near future [1]. The continuous retraining test aims to check if these updates can decrease the number of observed performance degradation patterns.

Similar to the Temporal Degradation Test in 2.1, the Continuous Retraining Test is an iterative process where we start from a dataset previously arranged in chronological order based on when each observation was gathered.

Then, we randomly sample N_{train_0} adjacent observations and build an initial training set. Next, adjacent to the end of the training dataset, we select N_{test} observations that will represent the test set. Similarly, the following N_{prod} adjacent observations to the test set will be the examples used as a production set.

A visual representation of this can be seen in the initial state row of Figure 2.2, where we start from a random training point and then partition the training, test, and production set.

After this initial sampling step is performed, we perform a training procedure as described in Section 2.1 and collect the production performance scores. In the second iteration, we expand the training set by setting $N_{train_1} = N_{train_0} + N_{test}$. The test and production sets are shifted accordingly. And perform the first retraining step. In the second row of Figure 2.2, we can see how the train (green) set grows, and the test and production sets shift accordingly. We continue this process for n retraining iterations.

The assumption is that as the training set grows, we allow the model to capture more patterns, making it less prone to predicted performance degradation. By analyzing the model performance from the production set results, we can test that assumption.

In the end, we plot the model's performance evolution and evaluate if the performance stays stable thanks to continuous retraining or if we still observe degradation patterns.

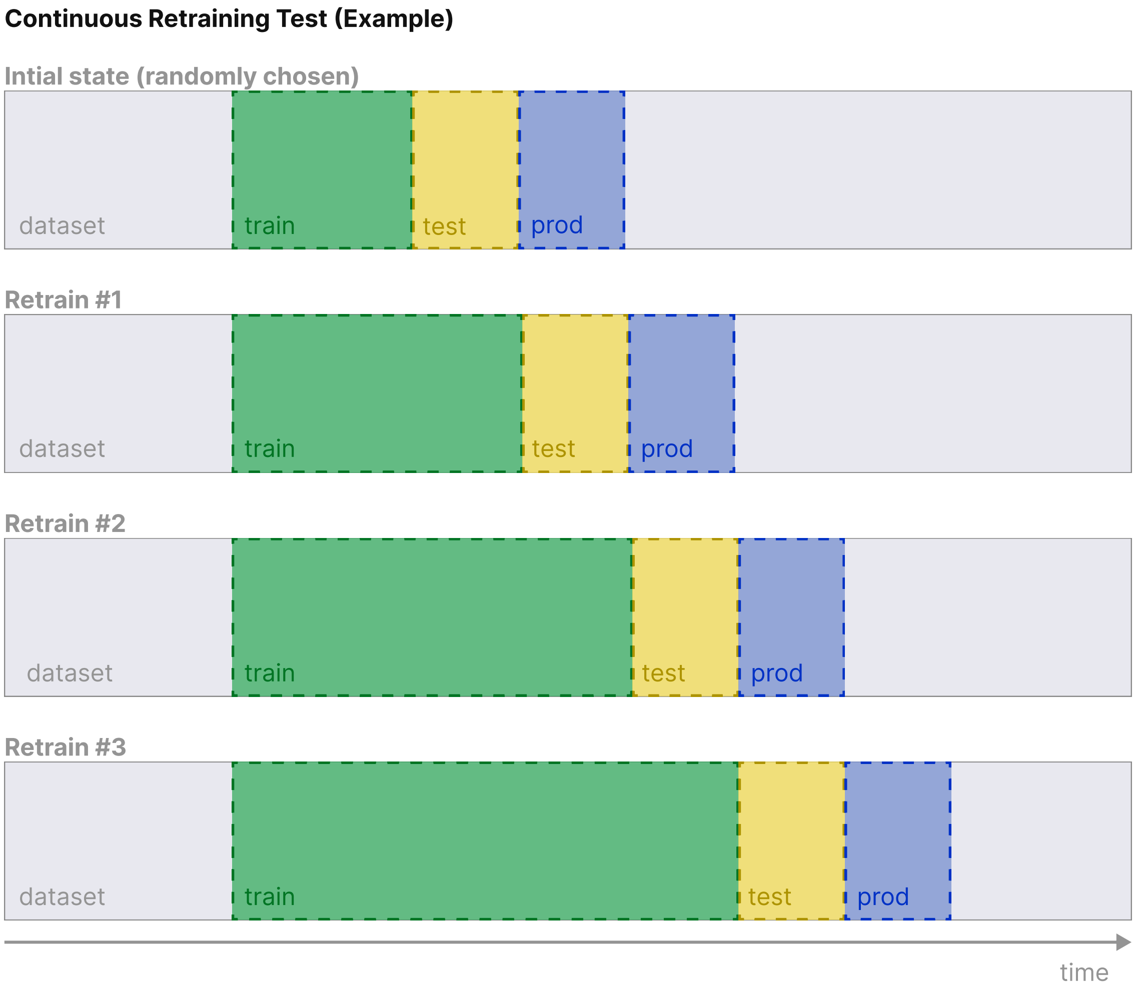

Figure 2.2: Continuous Retraining Experiment.

2.3 Performance Estimation Test

The performance estimation test leverages the Direct Loss Estimation (DLE) method from NannyML, an open-source Python library for post-deployment data science. The DLE method estimates the performance metric of regression models [3].

The way it works is as follows. Under the hood, DLE fits an additional Machine Learning model to estimate the value of the loss function of the monitored model. In our case, the monitored model will be each model from each simulation of the Temporal Degradation test of Section 2.1. After estimating the value of the loss function, DLE calculates the difference between the estimated loss and the actual loss of the model and turns this difference into the regression performance metric of our choice. We designed the Performance estimation test to study if DLE can correctly estimate the degradation patterns observed in the Temporal degradation test.

For each simulation of the Temporal degradation Test, we fit DLE with a reference dataset. The reference set consists of the previously used test dataset plus the most recent available data before the model was put into production. The reference set also contains the model predictions on that data. This is in order to calibrate the DLE method before it is used to estimate the performance of the production set. After each iteration, collect the estimated performance results and repeat for n simulations.

A visual representation of this process can be seen in Figure 2.3, where we start with the results of a simulation from the Temporal Degradation Test and later build the reference set to fit the DLE method. And estimate the performance of the production set. It is important to emphasize that at no time does DLE have access to the production set targets.

So, as in a real scenario, we are simulating that a model is in production, and we do not have a way to tell how well the model is performing on this new serving data. At the end of this performance estimation analysis, we will bring back the production target to validate and check the accuracy of NannyML's DLE method.

2.4 Methods and Hyperparameter Tunning

For all the experiments of the previously described methods, we will use the two datasets explained in Section 3 and four Machine Learning approaches (linear models, ensembles, boosted models, and neural networks). This is with the intention to explore the degradation pattern from different mathematical approaches when building Machine Learning models. The present section introduces these methods and their hyperparameter tunning procedures.

For each model, we validated the hyperparameter choice by using a TimeSeriesSplit cross-validation technique and used Optuna [6], an open-source hyperparameter optimization framework, to perform the optimal search. For every model-dataset pair, we performed 25 trials to find the combination of hyperparameter values that reduces the most the mean cross-validated coefficient of determination of the predictions.

The search space for the optimal hyperparameters is defined in the following subsections.

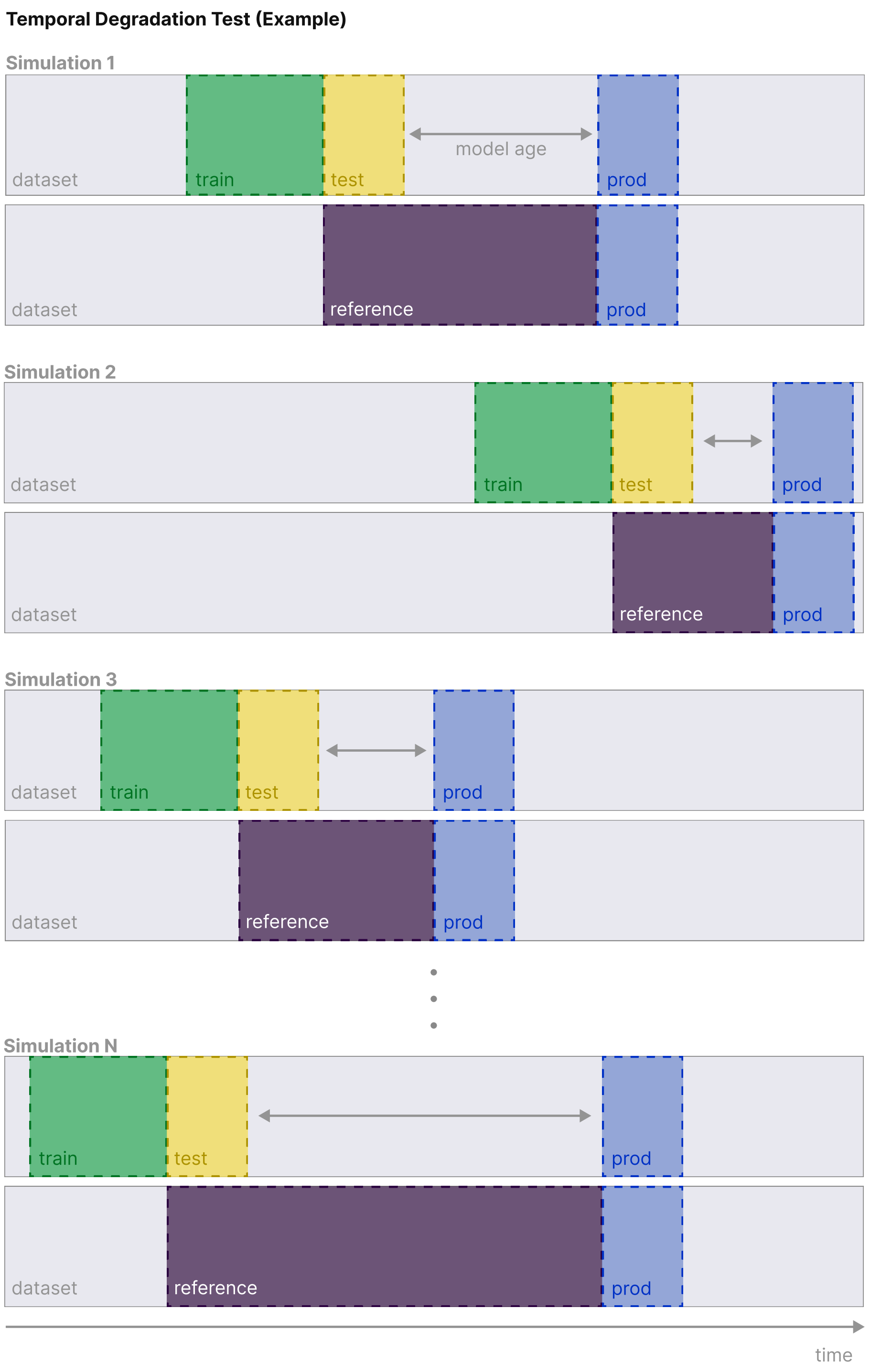

Figure 2.3: Performance Estimation Test.

2.4.1 LGBMRegressor

As our choice for a gradient-boosted model, we use a light gradient-boosting machine regressor (LGBMRegressor) [7]. The search space for the optimal hyperparameters was defined as follows.

- **num_leaves**: integer distribution in the domain [2, 24]
- **max_depth**: integer distribution in the domain [1, 13]
- **min_child_samples**: integer distribution in the domain [10, 50]
- **n_estimators**: integer distribution in the domain [100, 2000]
- **learning_rate**: float distribution in the domain [0.0001, 0.3]
- **reg_alpha**: float distribution in the domain [0, 1000]
- **reg_lambda**: float distribution in the domain [0, 1000]
- **colsample_bytree**: float distribution in the domain [0, 1]
- **subsample**: float distribution in the domain [0, 1]

2.4.2 ElasticNet

An ElasticNet [8] was our choice for a linear model since it allows us to play with the lasso and ridge regularization coefficients. The search space for the optimal hyperparameters was defined as follows.

- **alpha**: float distribution in the domain [0, 1000]
- **l1_ratio**: float distribution in the domain [0, 1]
- **max_iter**: integer distribution in the domain [1000, 2000]

2.4.3 RandomForestRegressor

A RandomForestRegressor [9] was our choice for an ensemble model. The search space for the optimal hyperparameters was defined as follows.

- **n_estimators**: integer distribution in the domain [100, 400]
- **max_depth**: integer distribution in the domain [1, 13]
- **min_samples_split**: integer distribution in the domain [2, 10]

2.4.4 MLPRegressor

Finally, a Multilayer Perceptron Regressor (MLPRegressor) [10] was used as the neural network alternative. The search space for the optimal hyperparameters was defined as follows.

- **hidden_layer_sizes**: integer distribution in the domain [20, 150]
- **activation**: categorical distribution in the domain ['relu', 'tanh']
- **solver**: categorical distribution in the domain ['lbfgs', 'sgd', 'adam']
- **alpha**: float distribution in the domain [0.0001, 0.01]
- **learning_rate_init**: float distribution in the domain [0.0001, 0.01]
- **max_iter**: integer distribution in the domain [300, 1000]

3 Datasets

In this chapter, we review in detail the two datasets that we will be using for all the experiments related to the Temporal Degradation, Continuous Retraining, and Performance Estimation Tests. We will describe the pre-processing and feature engineering process necessary to build the final time-series datasets. Then, we will explore them in order to gain some valuable insights before the models are designed, and finally, we will build a baseline model and describe a first post-deployment analysis.

3.1 US Avocado Hass Sales

The US Avocado Hass Sales dataset is an open dataset provided by the Avocado Hass Board [11]. It contains weekly sales of Hass avocados across the United States. Apart from date-time features, the dataset contains information about sales from different regions, avocado types, average prices, and the total volume of avocados sold weekly.

This is the first dataset that we will use to investigate the performance degradation in Machine Learning models. As mentioned before, we will fit Machine Learning models from different mathematical natures on different training and testing intervals to estimate the weekly sales amount and evaluate their performance over time. In the following subsection, we describe in detail the pre-processing and feature engineering pipeline that we applied to create the time-series dataset for this specific task.

3.1.1 Data Pre-processing and feature engineering

We took the original dataset and kept only the date, average price, and demand columns. Then, we created 6 new date features.

- **month:** month of the inference time.
- **week:** week of the inference time.
- **week_in_month:** week number in the month of inference time.
- **day:** day of the inference time.
- **days_since_last_holiday:** how many days have passed since the last holiday.
- **days_since_until_next_holiday:** how many days need to pass for the next holiday.

Apart from date-related features, we created two categories of lag columns for the weekly average price and weekly avocado demand. The categories are:

- **average_demand_in_last_n_weeks:** weekly average avocado demand in the last n weeks.
- **demand_n_weeks_ago:** weekly avocado demand n weeks ago.
- **average_price_in_last_n_weeks:** average avocado selling price in the last n weeks.
- **price_n_weeks_ago:** avocado selling price n weeks ago.

Where n goes from 1 to 7, so, during each prediction, the model has access to the past 7 weeks of avocado price and demand. The final dataset contains 33 features: 5 date-related and 28 lag features.

3.1.2 Exploratory data analysis

In this section, we will perform an Exploratory Data Analysis (EDA) to gain a better understanding of the US Avocado Hass Sales dataset. We will focus on exploring the demand variable since it is the variable that we will try to model. We will also study how other variables might impact it.

Let's remember that the focus of this study is to investigate how Machine Learning models degrade with time and what tools can help us detect or avoid those degradations. As mentioned in Chapter 2, to achieve this, we will run simulations where we sample different training sets from a dataset and create different training-model pairs to study their performance evolution. So, the EDA explained here applies only to one of those train samples. To simplify the analysis, we have selected the first 15 months of the dataset to perform the EDA. We decided to use only a portion of the dataset to not be biased about the findings here when we get to build the models.

We begin the exploration by checking the weekly avocado units sold per week. Looking at the left plot of Figure 3.1, we see how the weekly demand changes over time. During almost all of 2015, we see that the demand stays between the 80-90 million range. The month of December 2015 is when we observed the lowest demand point, followed by two big spikes in the first half of 2016. The right plot of the same figure shows the distribution of the demand column. It shows a slightly right-skew distribution with a median, red dashed line, close to 90 million units. Furthermore, it is worth noting that 50% of the observations fall between the 80-90 million units range.

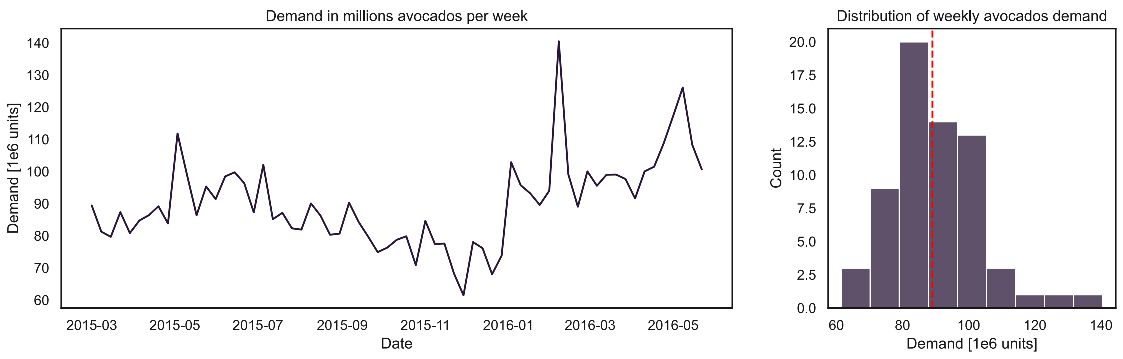

Figure 3.1: Left: Weekly demand in million units of avocados. Right: Distribution of the avocados demand variable.

Additionally, we wanted to investigate if the week in a month or the month impacts the demand for avocados throughout the year. To analyze that, we checked how the demand is distributed per week and month. On the left of Figure 3.2, we see that the week in a month plays almost no role in the demand for avocados. The median stays very stable at the 90 million mark across different weeks in a month. Probably the most significant difference is that week four seems to have the biggest variability. What it tells us is the lowest and highest demand peaks are more likely to happen at the end of the month.

On the contrary, when looking at the right-hand side of Figure 3.2, we do see an effect of the month on the response variable. On average, the demand seems to be high in the first half of the year, with its highest peak in April and lower in the rest of the year.

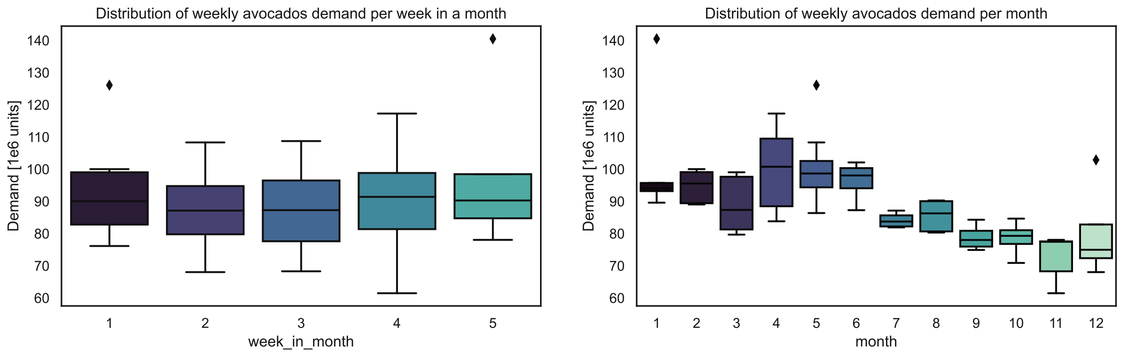

Figure 3.2: Left: Boxplots of the avocados demand at different weeks in a month. Right: Boxplots of avocados demand per month

Next, we wanted to verify if there was a relationship between avocado demand and price. In Figure 3.3, we see how there is a slight trend suggesting that there tends to be an increase in demand when the price is lower. In this case, we compare the demand with the previous week's price. This is because the inference time occurs one week before the actual event happens. For this reason, the actual avocado price in the week of interest is still unknown when the model makes a prediction.

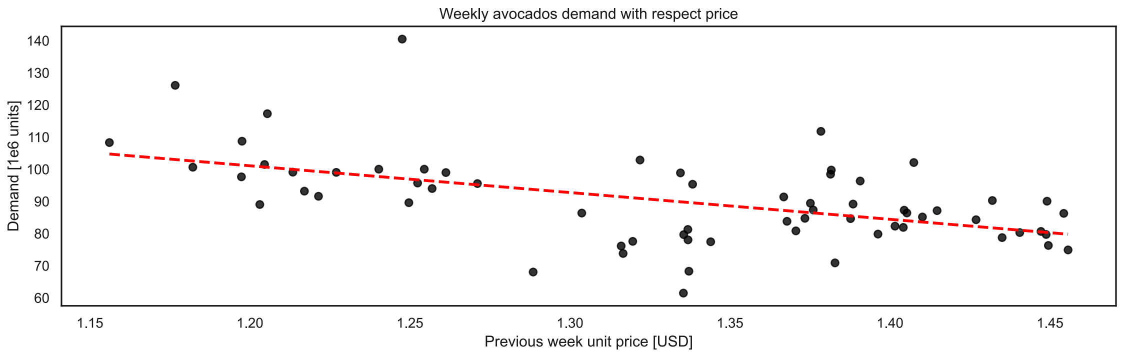

Figure 3.3: Avocado's demand with respect to the previous week's unit price.

Finally, we close the US Avocados Sales EDA with Figure 3.4 showing a heatmap correlation matrix with the correlation between all the continuous features and the demand response variable. We see that the variables that correlate the most with the demand are the total volume of units sold in the previous 1 to 7 weeks. It is important to note that the price-related variables also show the same correlation magnitude but in the opposite direction.

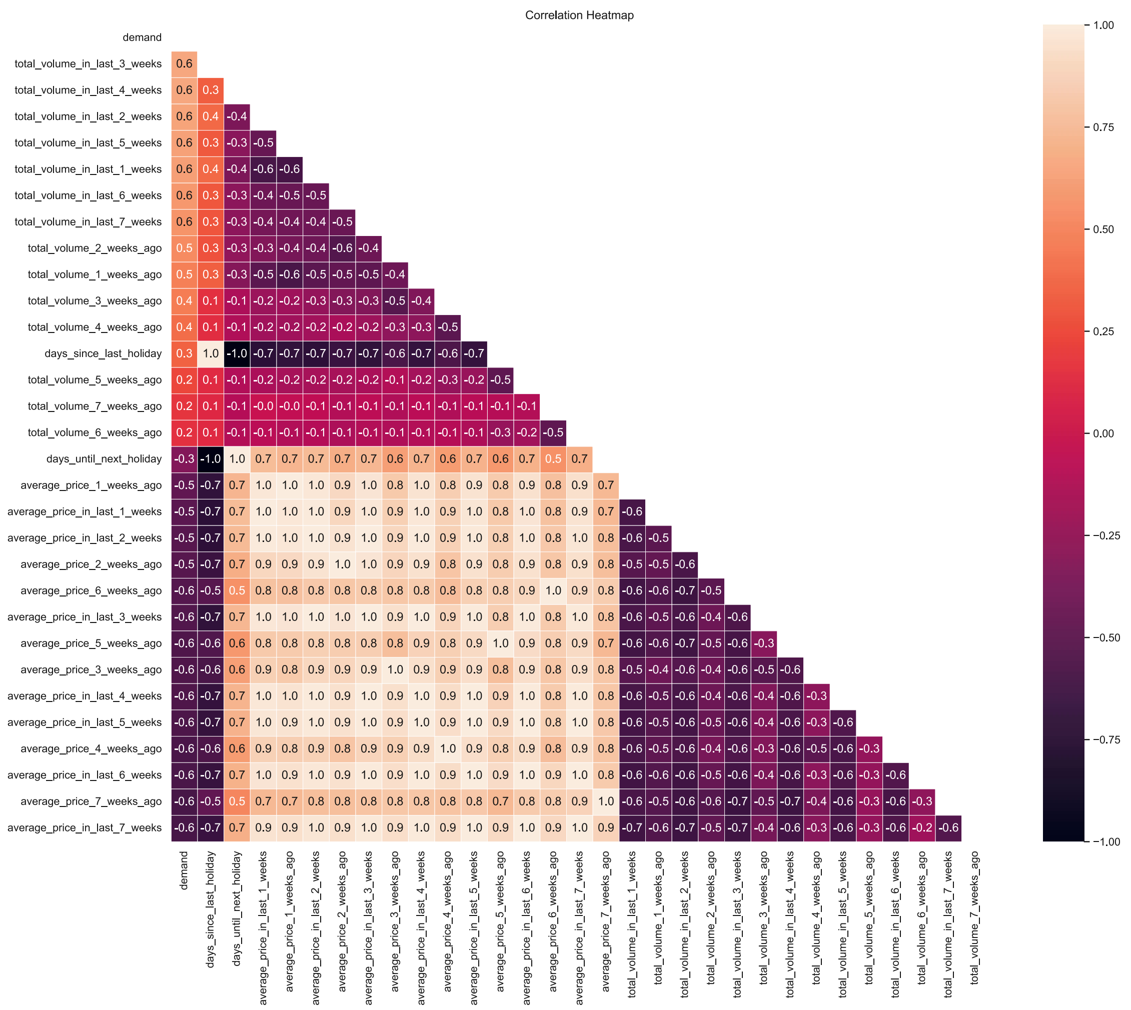

Figure 3.4: Correlation heatmap between all continuous features and the demand variable.

3.1.3 Baseline model and first post-deployment analysis.

Before formally starting the Temporal Degradation Test described in Section 2.1, we built a baseline model to set up some expectations and check whether or not the independent variables that we have designed would be sufficient to model the avocado demand curve. For this process, we used the first 15 months of the dataset as the training set and fitted an LGBMRegressor on its default parameters. We tested it on the following 3 months after the end of the training data. Figure 3.5 shows the goodness-of-fit plots for both the train and test sets and their respective Mean Average Percentage Error (MAPE). The value after the < > symbols is the MAPE of a model whose predictions are always equal to the average demand observed on the train set.

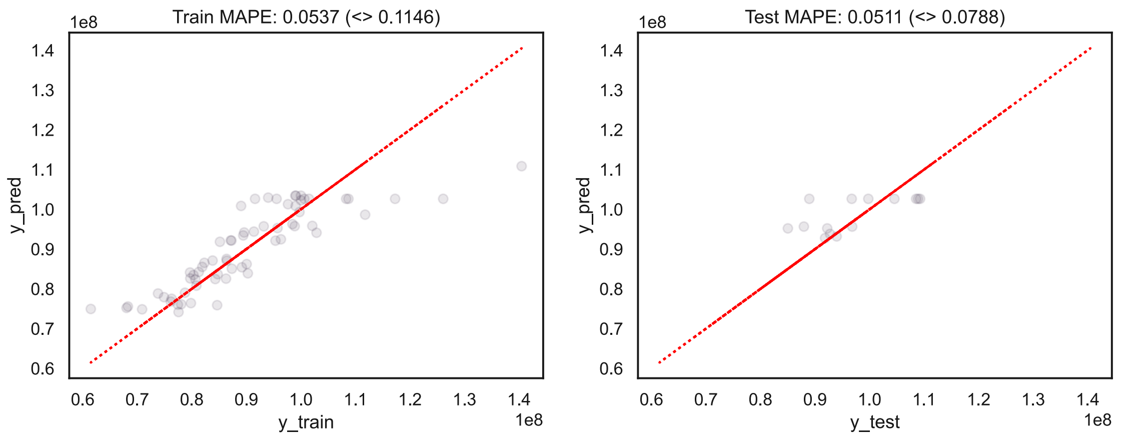

Figure 3.5: Goodness-of-fit plot for the Avocados demand forecasting task. Left: Train results. Right: Test results.

Figure 3.6 shows the top 10 features with the highest importance level for the LGBM Regressor model. The average price five and six weeks prior to the prediction ended up being the two most influential features of the model. The importance_type parameter when measuring the importance was set to 'split', meaning that each value in Figure 3.6 represents the number of times a feature was used in the model [12].

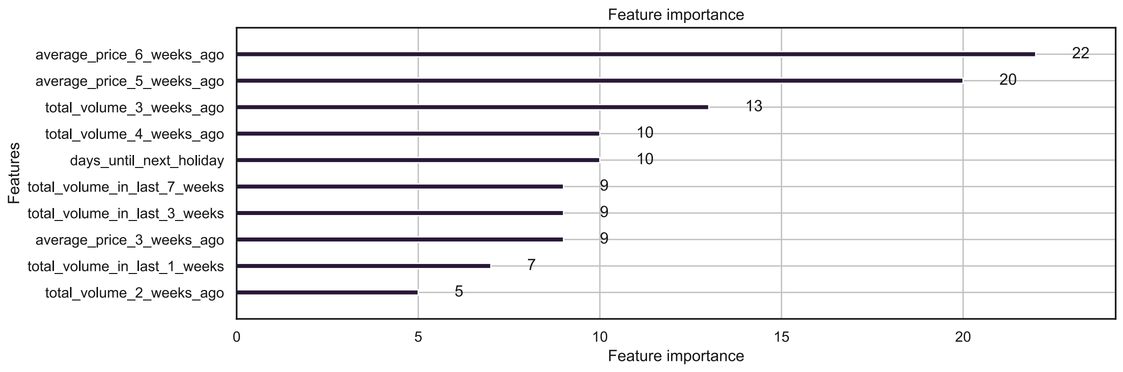

Figure 3.6: Top 10 feature importance's for the avocados demand prediction LGBM Regressor model.

We followed the analysis with an appetizer of what we will study on the Temporal Degradation Test. Figure 3.7 shows in blue the actual demand curve throughout all the available time periods. In red, we see the predicted values from the LGBM Regressor model. The light green and yellow areas represent the training and testing periods, respectively.

We see how, during the training period, the model catches relatively well the demand curve behavior. And in the months after, it is able to approximate the general trend but without completely catching its ups and downs.

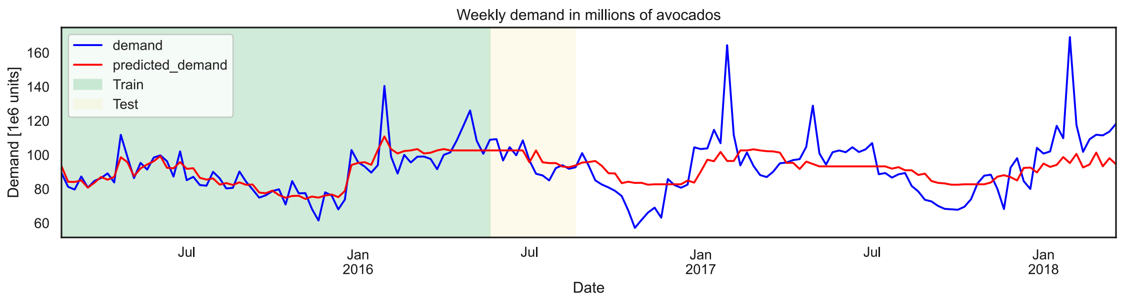

Figure 3.7: Weekly predicted and actual avocado demand through time.

Finally, we plot the evolution of the absolute error through time. We see how, during the training and testing periods, the error was relatively low, but in the months after testing, the absolute error reached its highest magnitudes. This is the type of model performance degradation that we will try to avoid by implementing a continuous retraining workflow and or by estimating the model performance.

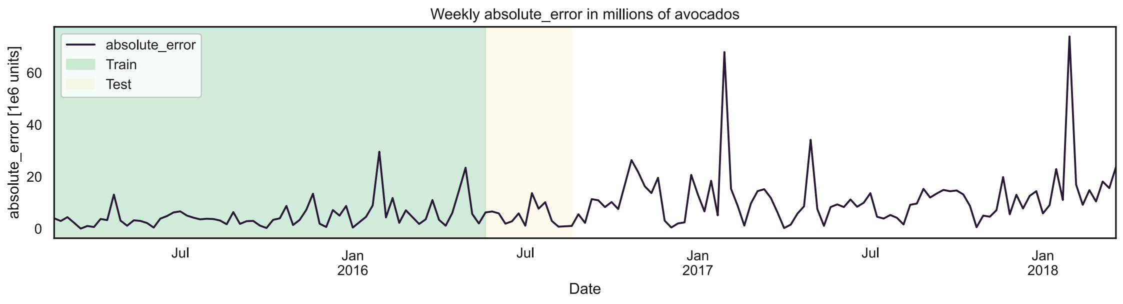

Figure 3.8: Evolution of the absolute error for the Avocado's demand forecasting problem.

3.2 NYC Taxi Service Demand

The NYC Taxi Service Demand dataset is a collection of datasets from the New York City Taxi & Limousine Commission (TLC). It contains information about yellow taxi trip records and includes fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts [13]. The dataset contains hourly information about taxi trips from 2011 to 2022.

Similar to the US Avocado Hass Sales dataset in Section 3.1. We will use this dataset to investigate the performance degradation in Machine Learning models. We will train different

Machine Learning models with different training and testing intervals to estimate the number of hourly taxi trips. In the following section, we describe in detail the pre-processing and feature engineering pipeline that we applied to create the time-series dataset for this task.

3.2.1 Data Pre-processing and feature engineering

We took the original granular dataset containing trip-level data and aggregated it per hour. So we end up with a dataset having information such as total_trips total_passenger_count, total_trip_distance, total_tip_amount, total_amount, and total_trip_duration, per hour.

Then we created 8 date-related features.

- **month:** month of the inference time.
- **week:** week of the inference time.
- **day:** day of the inference time.
- **day_of_week:** day of the week of the inference time.
- **hour:** hour of the inference time.
- **holiday:** boolean flag stating if the day is a holiday or not.
- **days_since_last_holiday:** how many days have passed since the last holiday.
- **days_since_until_next_holiday:** how many days need to pass until the next holiday.

Additionally, for all the original features: total_trips, total_passenger_count, total_trip_distance, total_tip_amount, total_amount, and total_trip_duration. We created two categories of lag columns:

- **{feature_name}_in_last_{n}_days**
- **{feature_name}_{n}_days_ago**

Where feature_name is every feature from the original dataset features, and n goes from 1 to 7. This means that the resulting dataset contains 92 features: 8 date-related features and 84 lag features.

3.2.2 Exploratory data analysis

In this section, we will follow a similar approach as in the EDA of the US Avocado Hass Sales dataset on Section 3.1.2. We will focus on exploring the total trip demand variable since it is the variable that we will try to model. We will also study how other variables might impact it. The EDA here is applied to the first 15 months of the NYC Taxi Service Demand dataset. As mentioned before, we decided to do the EDA for only a portion of the dataset in order not to be biased about the findings here when we get to build the models.

We begin the exploration by checking the daily demand for taxi trips. Looking at the left plot of Figure 3.9, we see how the daily demand behaves during all of 2011 and the first half of 2012. The daily demand seems to stay most of the time between the 400-600 thousand trips range, with the apparent exception of some drastic downs, especially the one in September 2011. The right plot on the same figure shows the distribution of the taxi demand column. It shows a left-skew distribution with a median, red dashed line, of 490 thousand daily trips. Furthermore, it is worth noting that 50% of the observations fall between the 453-490 thousand trips range.

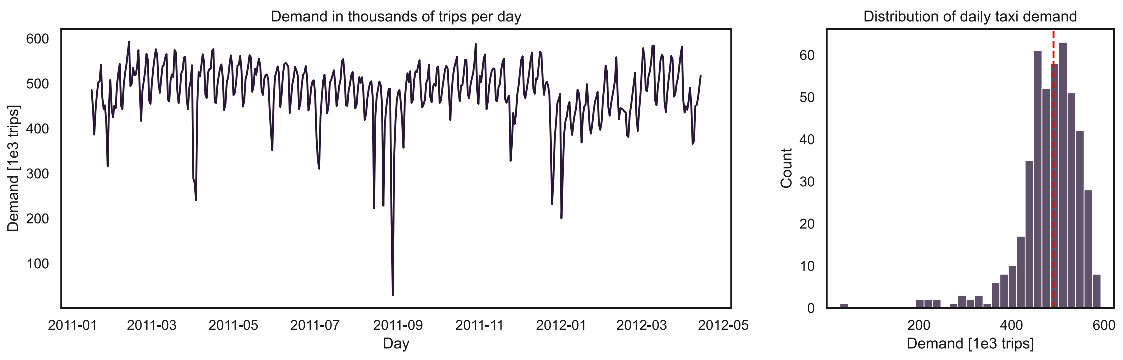

Figure 3.9: Daily taxi demand in thousands of trips.

Additionally, we wanted to investigate if the day in a week or the month has an impact on the taxi demand throughout the year. In order to analyze that, we checked how the demand is distributed per day and month. On the left of Figure 3.10, we see that the day in a week plays a slight role in the taxi demand. The median seems to increase as the week approaches the sixth day (Friday); Saturday seems very similar to Friday's values, except that it has more variability. And Sunday is the day where it is likely to have the least taxi demand during the week. On the other side, when we look at the daily trip distributions per month, the results seem a bit more stable with the exception of January, which is the month with the least amount of taxi trips.

Finally, we wanted to verify if there was a relationship between taxi demand and holidays.

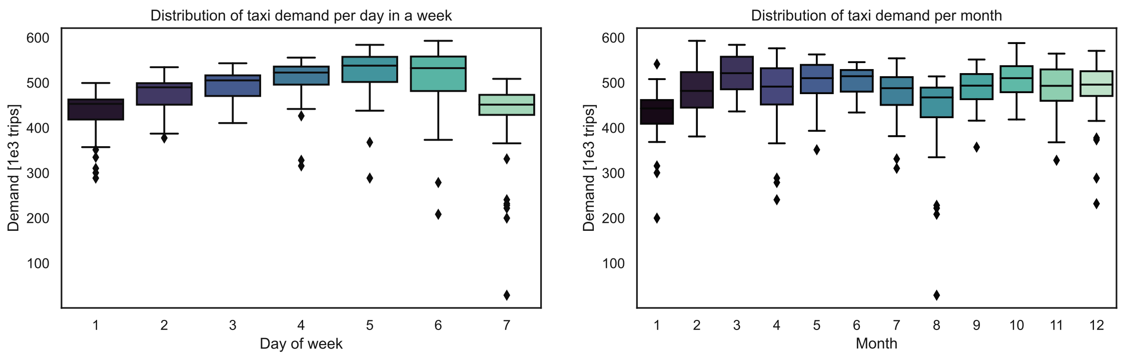

Figure 3.10: Left: Boxplots of the taxi demand on different days in a week. Right: Boxplots of taxi demand per month

Figure 3.11 suggests that there tends to be less taxi demand on holidays. Non-holidays also show more outliers than their counterparts.

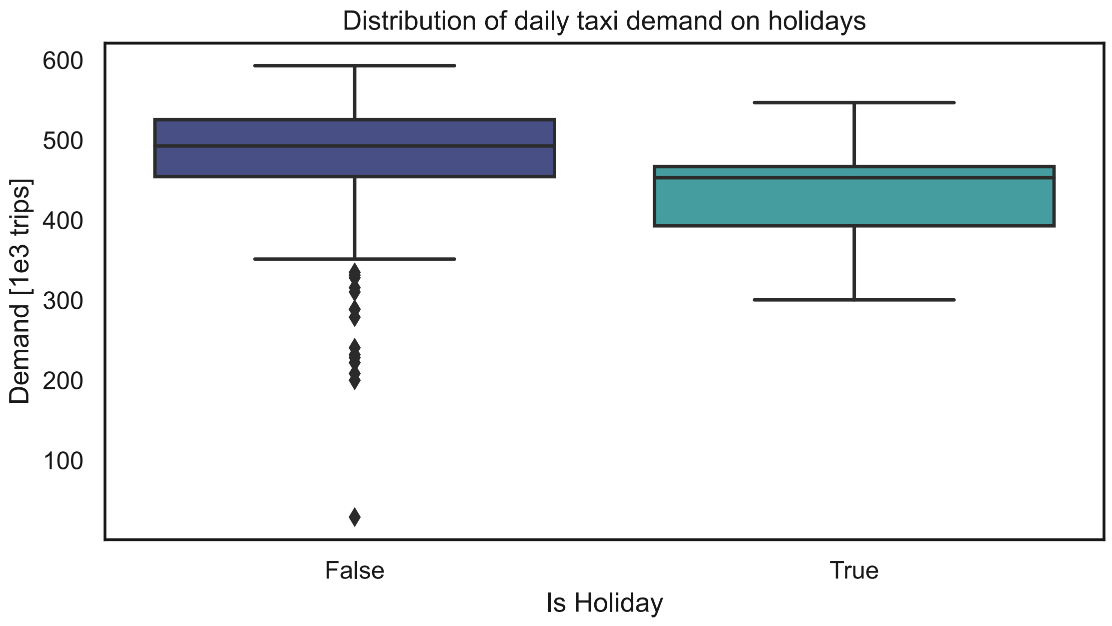

Figure 3.11: Distribution of daily taxi demand on holidays.

3.2.3 Baseline model and first post-deployment analysis

Again, similar to the baseline model for the US Avocados Hass Sales dataset in Section 3.1.3, we decided that before formally starting the Temporal Degradation Test described in Section 2.1, we will build a baseline model to set up some expectations and check whether or not the independent variables that we have designed would be sufficient to model the taxi demand curve. For this process, we used the first 15 months of the dataset as the train set and fitted an LGBM Regressor on its default parameters. We tested it on the following 3 months after the end of the training data. Figure 3.12 shows the goodness-of-fit plots for both the train and test sets and their respective Mean Average Percentage Error (MAPE). The value after the < > symbols is the MAPE of a model whose predictions are always equal to the average demand observed on the train set.

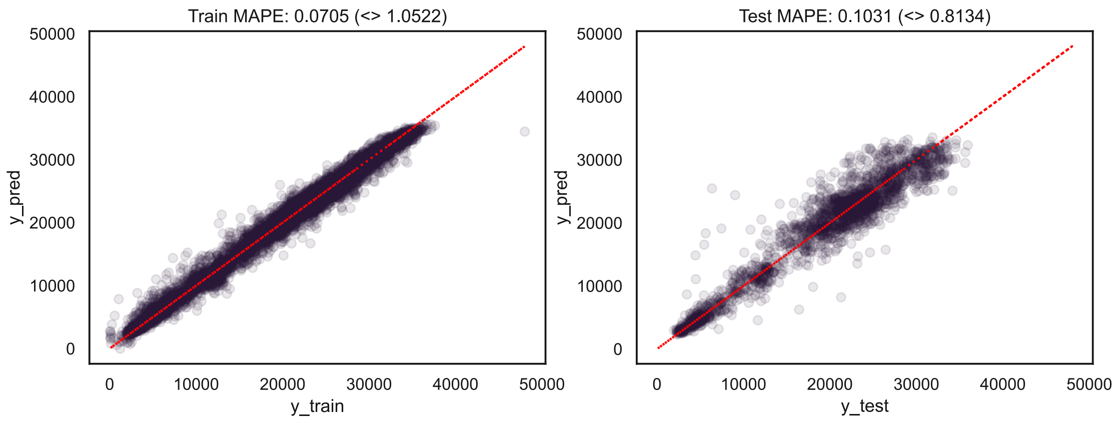

Figure 3.12: Goodness-of-fit plot for the taxi demand forecasting task. Left: Train results. Right: Test results.

Figure 3.13 shows the top 10 features with the highest importance level for the LGBM Regressor model. Interestingly, week, hour, and day were by far the three most influential features of the model.

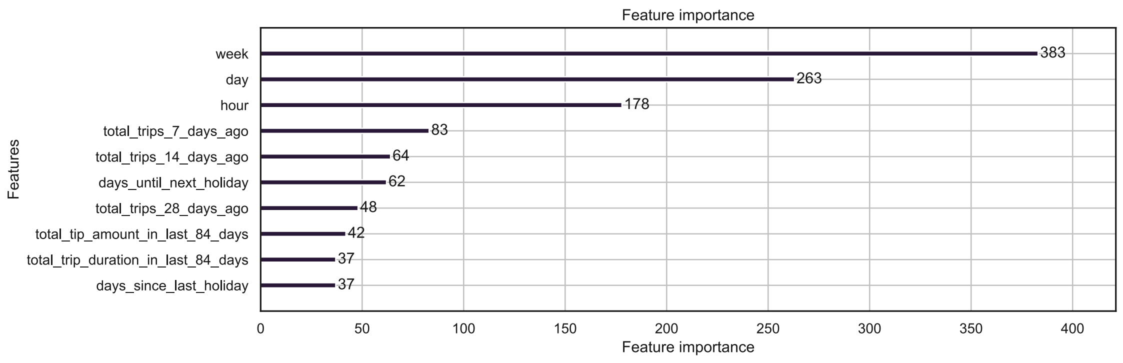

Figure 3.13: Top 10 feature importance's for the taxi demand prediction LGBM Regressor model.

Again, we followed the analysis with an appetizer of what we will study on the Temporal Degradation Test. Figure 3.14 shows in blue the actual demand curve throughout all the available periods. In red are the predicted values from the LGBM Regressor model. The light green and yellow areas represent the training and testing periods, respectively.

We see how, during the training period, both the blue and red curves overlap almost perfectly. Then, between 2013 and 2017, the model behaved relatively well, but after that, the gap between the actual and the predicted demand increased. It is still surprising that the model catches the downside trend in 2020 (COVID-19), although it still overestimates the expected demand.

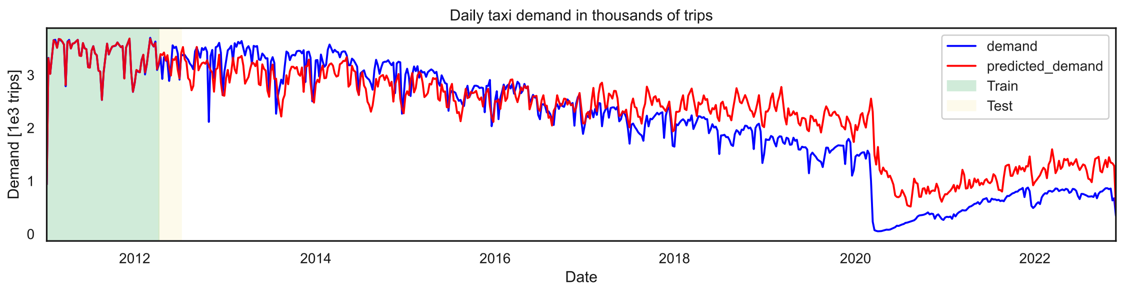

Figure 3.14: Weekly predicted and actual taxi demand through time.

Finally, we plot the evolution of the absolute error through time. Figure 3.15 shows how, during training, the absolute error was basically zero and relatively low during testing. But in the following years, after testing, the absolute error reached its highest magnitudes, especially during 2020. As mentioned before, this is the type of model performance degradation that we will try to avoid by either implementing a continuous retraining workflow or by estimating the model performance on the production serving data.

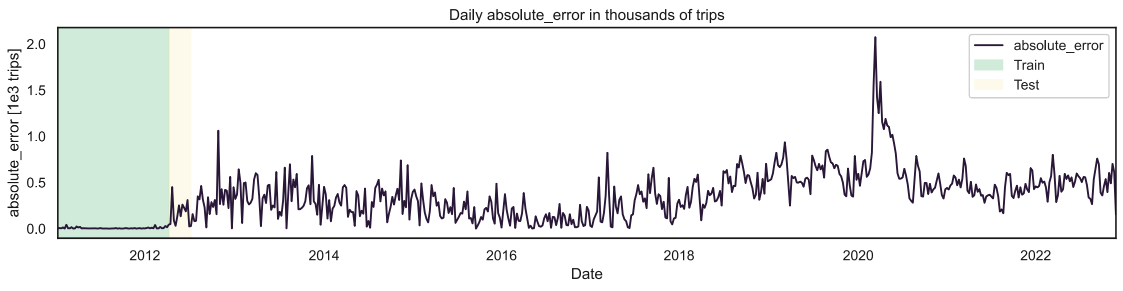

Figure 3.15: Evolution of the absolute error for the taxi's demand forecasting problem.

4 Results and Discussion

In this chapter, we present the results from the experiments described in Chapter 2 using the datasets described in Chapter 3. As mentioned before, for each experiment, we analyze the results of applying four different Machine Learning approaches (linear models, ensembles, boosted models, and neural networks). This will allow us to study whether the same dataset trained on different model approaches shows different degradation results. Or if a type of model is more prone to lose its performance over time. It is worth mentioning that all the results that we will observe are from models with a good initial fit. By good initial fit, we refer to models with a test MAPE of lower than 0.1. We will consider it a performance degradation any production performance above the 0.15 MAPE value.

4.1 Temporal degradation results

The previously described Performance Degradation Test was performed on the US Avocado Hass Sales dataset and on the NYC Taxi Trip Demand dataset. The following two subsections describe the experiment settings, results, and discussion.

4.1.1 Dataset: US Avocado Hass Sales

For every simulation of the Temporal Degradation Test with the US Avocado Hass Sales dataset, we used a randomly sampled training set of 52 weeks, a test set of 12 weeks, and a production set of 24 weeks. In total, we ran 3000 model-dataset pair simulations. As mentioned before, we performed a hyper-parameter tuning for each of these 3000 simulations to find the optimal parameters. For this, as mentioned in Chapter 2 we used Optuna to automate the hyperparameter search.

LGBMRegressor Results

From the original 3000 LGBMRegressor-Avocado pair simulations, 2684 qualified as models with a good initial fit, indicating that 316 models had a testing MAPE greater than 0.1. In Figure 4.1, you can observe the model aging plot of an LGBMRegressor trained on the US Avocado Hass Sales dataset. Each purple dot represents the observed performance result of the production dataset at a given model age. The yellow, black, and red lines indicate the weekly 25th, 50th, and 75th percentiles, respectively. It is notable that the percentile line remains relatively stable during the first 50 weeks of the production period. However, after week 50, a slight increase in performance error becomes apparent, which is expected, given that the model is approximately one year old at that point. When using a 0.15 MAPE threshold to measure degradation, it was found that only 11.7% of the predictions had a MAPE greater than 0.15. However, among all 2684 models with a good initial fit, 2453 (91.4%) exhibited at least 5 instances of degradation.

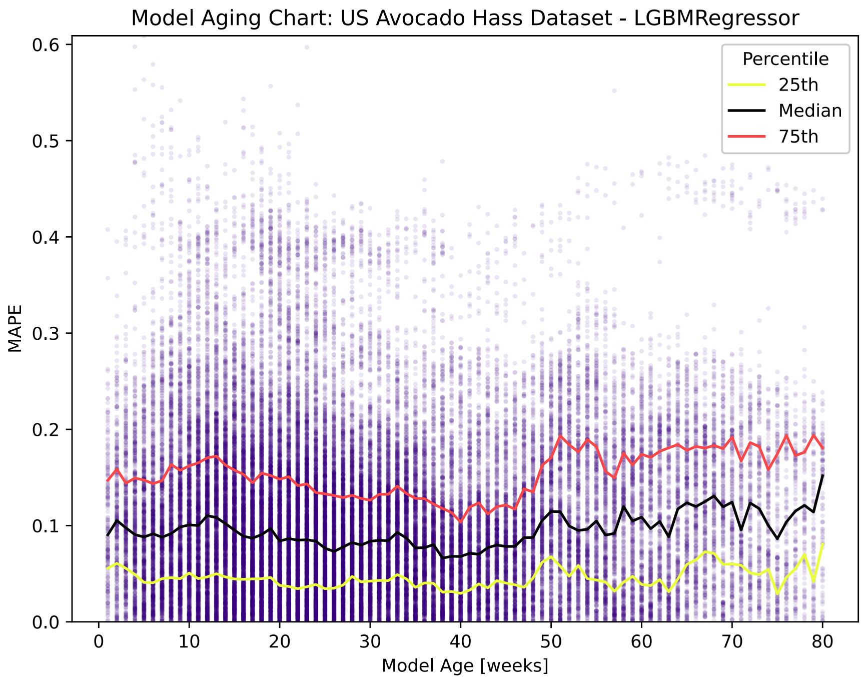

Figure 4.1: Model Aging plot of the LGBMRegressor model trained on the US Avocado Hass Sales dataset.

ElasticNet Results

Let's now examine the results of the same experiment, but this time using an ElasticNet model. Out of the original 3000 ElasticNet-Avocado pair simulations, 2423 models qualified as having a good initial fit, indicating that 577 models had a testing MAPE greater than 0.1. Figure 4.2 illustrates the model aging plot of an ElasticNet model trained on the US Avocado Hass Sales dataset.

This time, we can clearly discern two distinct patterns. During the first 40 weeks of the model's production, the weekly median MAPE remains below 0.1 but then increases rapidly, with the median weekly values approaching 0.2. It's worth noting the widening gap between the 25th and 75th percentiles, signifying significant performance degradation. Observing the percentile lines, the ElasticNet model exhibits more pronounced degradation than the other aging plots, with errors appearing clustered and less variable.

When using a 0.15 MAPE threshold to measure degradation, only 14.6% of the predictions had a MAPE greater than 0.15. However, out of all 2423 models with a good initial fit, 2423 (100%) exhibited more than 9 instances of degradation, indicating that every ElasticNet model experienced degradation at some point.

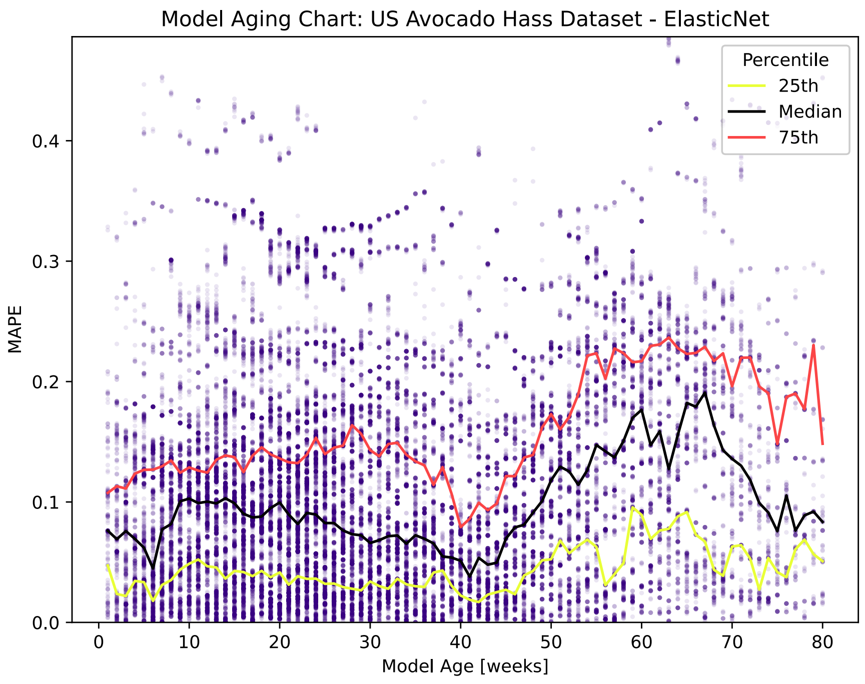

Figure 4.2: Model Aging plot of the ElasticNet model trained on the US Avocado Hass Sales dataset.

RandomForestRegressor Results

Let's now examine the results of the same experiment, this time using a RandomForestRegressor model. Out of the original 3000 RandomForestRegressor-Avocado pair simulations, 2675 models qualified as having a good initial fit, with 325 models exhibiting a testing MAPE greater than 0.1. Figure 4.3 depicts the model aging plot of a RandomForestRegressor trained on the US Avocado Hass Sales dataset.

These results initially mirror those observed for the LGBMRegressor. The primary distinction lies in the fact that, around week 60, the gap between the 25th and 75th percentile lines narrows, indicating that most predictions at that time demonstrate poor performance.

When using a 0.15 MAPE threshold to assess degradation, it was found that only 11.4% of the predictions exceeded this threshold. However, out of all 2675 models with a good initial fit, 2492 (93.2%) exhibited more than 5 instances of degradation.

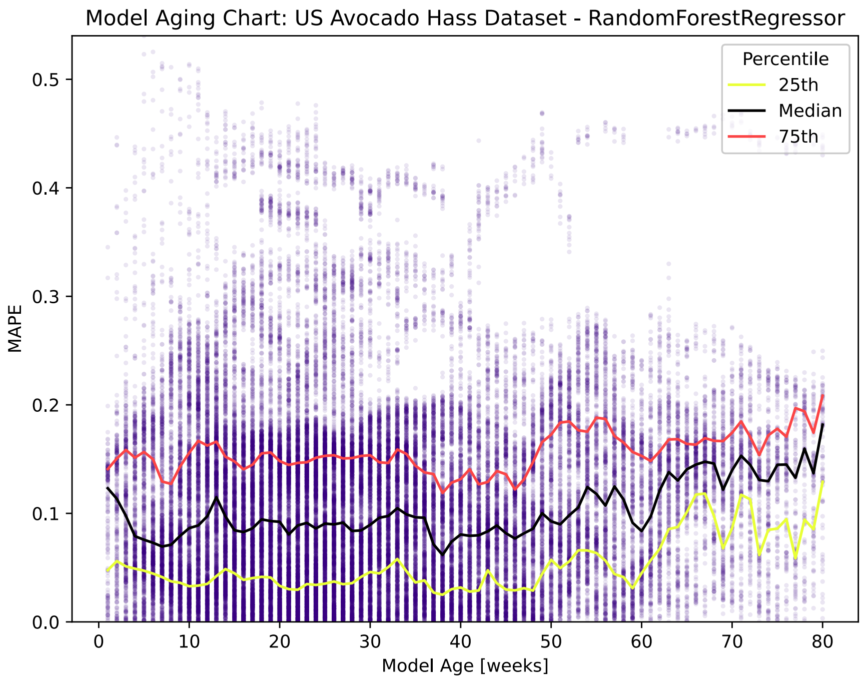

Figure 4.3: Model Aging plot of the RandomForestRegressor model trained on the US Avocado Hass Sales dataset.

MLPRegressor Results

Finally, let's now examine the results of the same experiment, this time using an MLPRegressor model. Out of the original 3000 MLPRegressor-Avocado pair simulations, 2045 models qualified as having a good initial fit, while 955 models exhibited a testing MAPE greater than 0.1. Interestingly, this model type generated the highest number of models that didn't meet the criteria for validity. Figure 4.4 illustrates the model aging plot of an MLPRegressor trained on the US Avocado Hass Sales dataset.

The weekly percentile lines appear relatively consistent throughout the entire period. Unlike the other models, the MLPRegressor did not experience as much fluctuation during the 40th to 50th weeks. However, upon closer inspection, the MLPRegressor exhibits the greatest variability in predictive errors, with some predictions exceeding a 0.8 MAPE.

When using a 0.15 MAPE threshold to assess degradation, it was found that 16.0% of the predictions surpassed this threshold. Nonetheless, among all the 2045 models with a good initial fit, 2032 (99.4%) displayed more than 5 instances of degradation.

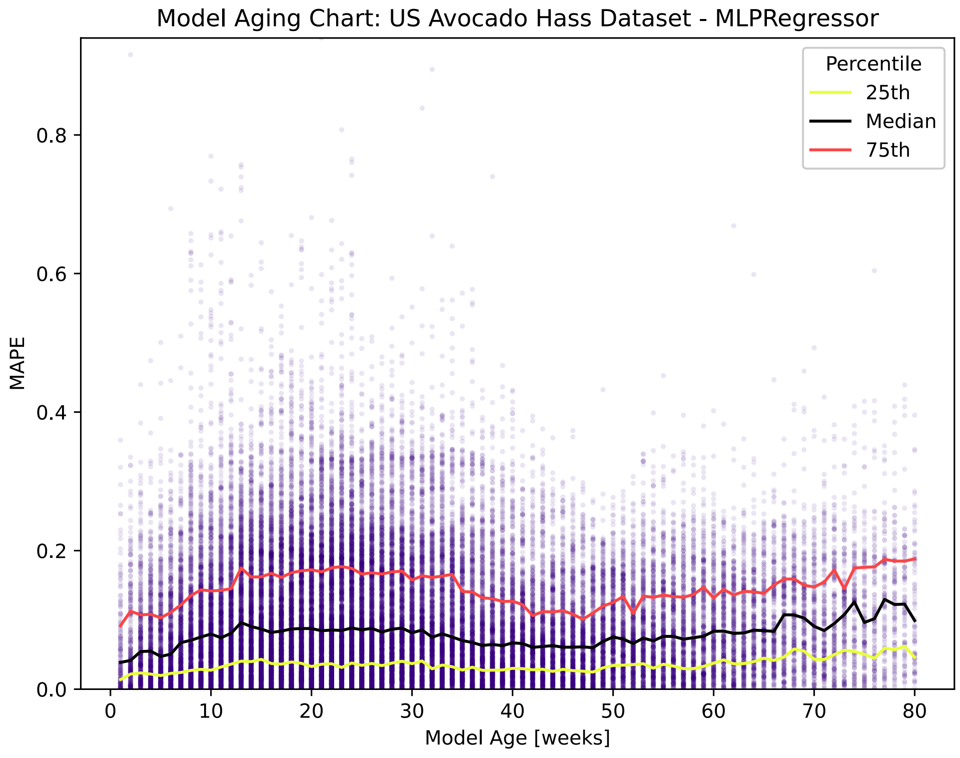

Figure 4.4: Model Aging plot of the MLPRegressor model trained on the US Avocado Hass Sales dataset.

4.1.2 Dataset: NYC Taxi Trip Demand

For each simulation of the Temporal Degradation Test using the NYC Taxi Trip Demand dataset, we employed a randomly sampled training set spanning 365 days, a testing set comprising 90 days, and a production set spanning 180 days. In total, we conducted 1500 simulations for model-dataset pairs. As mentioned previously, we carried out hyperparameter tuning for each of these 1500 simulations to determine the optimal parameters.

As discussed in the following results, predicting NYC Taxi Trip Demand presents a more challenging task than Avocado Sales prediction. This dataset spans eight years of data, and our predictions are made on a daily basis, in contrast to the weekly predictions for the Avocado dataset. You will observe that degradation patterns are more pronounced, and even certain models, such as the MLPRegressor, struggle to find a good initial fit.

LGBMRegressor Results

From the original 1500 LGBMRegressor-Taxi pair simulations, 1129 models met the criteria for a good initial fit, while 371 models exhibited a testing MAPE greater than 0.1. Figure 4.5 depicts the model aging plot of an LGBMRegressor trained on the NYC Taxi Trip Demand dataset. Each purple dot represents the daily observed performance result of the production dataset at a specific model age.

During the first year, the model maintains a relatively low MAPE, and the gap between the 25th and 75th percentiles remains small. However, shortly thereafter, a noticeable deterioration occurs. The gap between the 25th and 75th percentiles widens abruptly, indicating a significant increase in error variability and a degradation issue. When using a 0.15 MAPE threshold to assess degradation, it was discovered that 22.9% of the predictions exceeded this threshold. Nevertheless, among all 1129 models with a good initial fit, 1129 (100%) exhibited at least 27 instances of degradation. This implies that even the best model experienced degradation on at least 27 occasions during its lifetime.

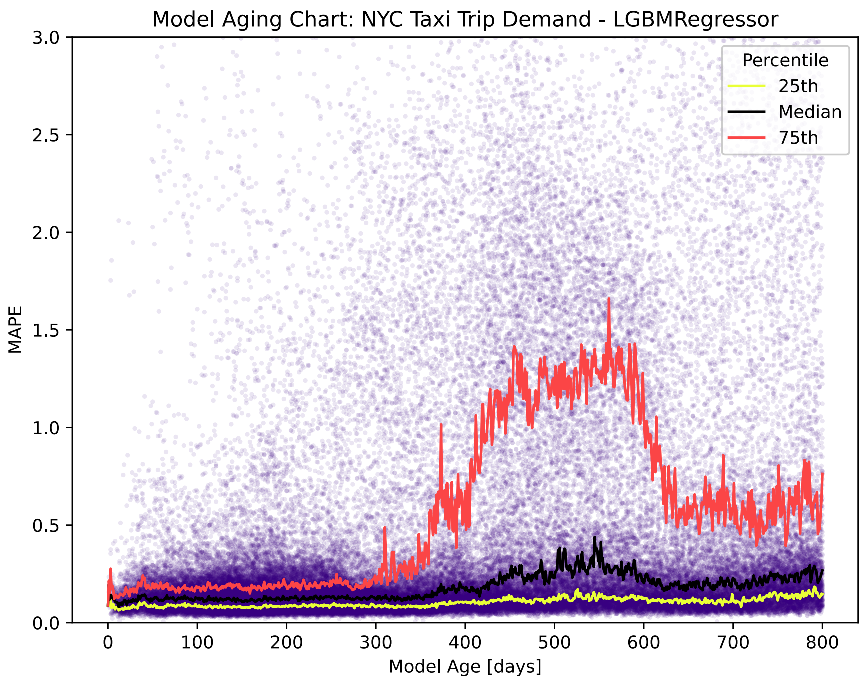

Figure 4.5: Model Aging plot of the LGBMRegressor model trained on the NYC Taxi Trip Demand dataset.

ElasticNet Results

Let's now examine the results of the same experiment, this time using an ElasticNet model. Out of the original 1500 ElasticNet-Taxi pair simulations, only 170 models met the criteria for a good initial fit, while 1330 models exhibited a testing MAPE greater than 0.1. This number is surprisingly low.

Figure 4.6 illustrates the model aging plot of an ElasticNet trained on the NYC Taxi Trip Demand dataset. This time, we observe a gradual degradation rather than an extreme increase in error variability. This highlights a notable difference between the LGBMRegressor and the ElasticNet model. The LGBMRegressor was able to produce more models with a good initial fit, although, on average, they experienced more pronounced degradation. Conversely, with ElasticNet, it was more challenging to find good initial models, but those few were more reliable in production. However, in the long term, they still exhibited degradation.

When using a 0.15 MAPE threshold to assess degradation, it was found that 48.5% of the predictions exceeded this threshold. However, out of all 170 models with a good initial fit, each one ended up showing at least 169 instances of degradation. The main distinction from the LGBMRegressor result is that these 169 degradations were less extreme.

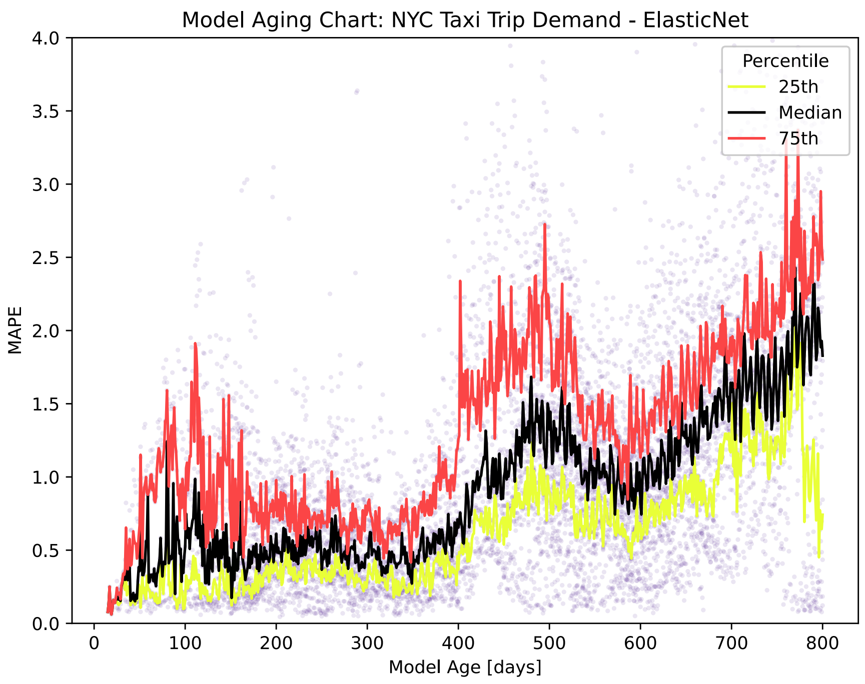

Figure 4.6: Model Aging plot of the ElasticNet model trained on the NYC Taxi Trip Demand dataset.

RandomForestRegressor Results

Finally, let's take a closer look at the results of the same experiment, this time using a RandomForestRegressor model. Out of the original 1500 RandomForestRegressor-Taxi pair simulations, only 1007 models met the criteria for a good initial fit, while 493 models exhibited a testing MAPE greater than 0.1.

Figure 4.7 illustrates the model aging plot of a RandomForestRegressor trained on the NYC Taxi Trip Demand dataset. Similar to the LGBMRegressor, we observe a rapid increase in error variability after the model's first year in production. One notable difference between this result and the LGBMRegressor is that around the 600th day, the median performance error begins to degrade more rapidly, whereas with the LGBMRegressor, it remains relatively stable.

When using a 0.15 MAPE threshold to assess degradation, it was found that 23.3% of the predictions exceeded this threshold. However, all 1007 models with a good initial fit experienced degradation at some point in time. Even the best model exhibited at least 27 instances of degradation during its lifetime.

Regarding the MLPRegressor, none of the 1500 simulations were able to generalize well enough on the test set to be considered models with a good initial fit, even when applying Optuna's automatic hyperparameter optimization algorithm.

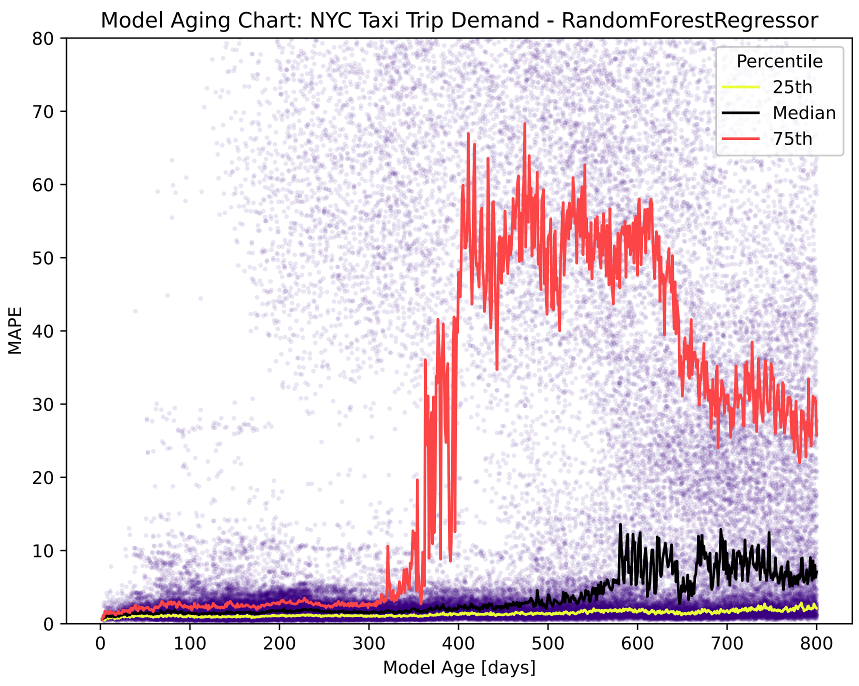

Figure 4.7: Model Aging plot of the RandomForestRegressor model trained on the NYC Taxi Trip Demand dataset.

4.2 Continuous retraining results

In this section, we review the results obtained from the Continuous Retraining Test, as described in Section 2.2. We conducted these experiments with both the US Avocado Hass Sales dataset and the NYC Taxi Trip Demand dataset. The following two sections describe the experiment settings, results, and discussions for each dataset, respectively.

4.2.1 Dataset: US Avocado Hass Sales

Similar to the Temporal Degradation Test, for each simulation, we utilized the US Avocado Hass Sales dataset and randomly selected a training set of 52 weeks, a testing set of 12 weeks, and a production set of 12 weeks. We conducted 3000 model-dataset pair simulations and executed 2 retraining steps after the initial fitting. Each retraining step followed the procedure described in Section 2.2.

Additionally, we performed hyper-parameter tuning for the initial fittings of all 3000 simulations. The subsequent two retraining steps utilized the discovered hyperparameters.

LGBMRegressor Results

Figure 4.8 summarizes the median MAPE results for the 3000 LGBMRegressor models following their initial fit and two subsequent retraining updates. Interestingly, the model with no retraining update and the one with a second retraining update exhibit very similar behavior, whereas the model after the first retraining update demonstrates the worst performance. As mentioned earlier, this experiment involved using 52 weeks of training data, 12 weeks for testing, and 12 weeks of production data. Consequently, the models were retrained every 12 weeks with the latest 12 weeks of available data. The observation that all three median MAPE lines display an upward trend suggests that re-fitting the models does not eliminate the possibility of predictive performance degradation.

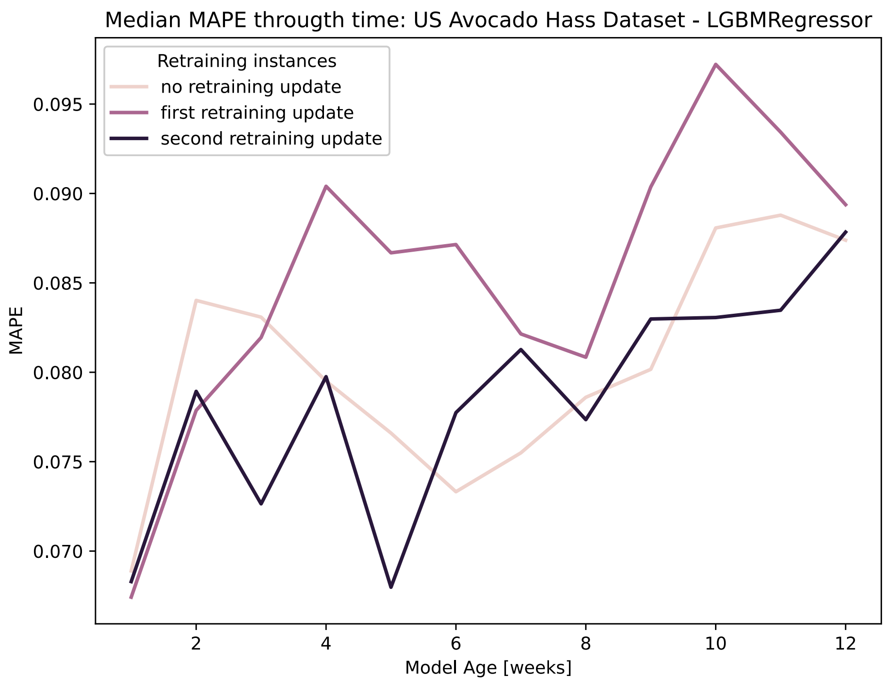

Figure 4.8: Model retraining plot of the LGBMRegressor model trained on the US Avocado Hass Sales dataset.

ElasticNet Results

Figure 4.9 presents the summary of median MAPE results for the 3000 ElasticNet models following their initial fit and two subsequent retraining updates. This outcome aligns with the common intuition that model performance tends to improve as we refit them with more data. However, similar to the previous result, the median MAPE lines also exhibit an upward trend, indicating performance degradation over time.

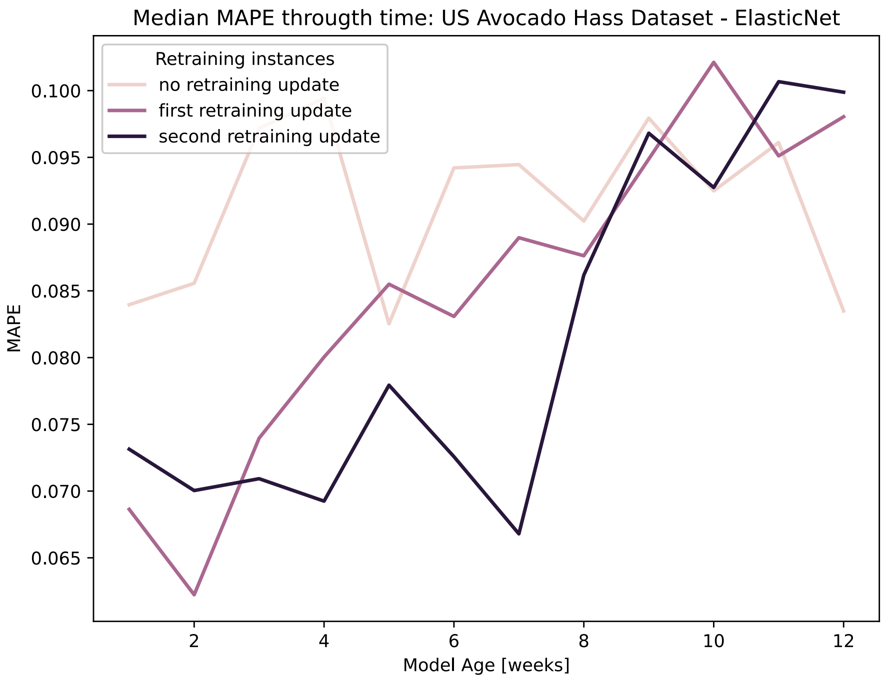

Figure 4.9: Model retraining plot of the ElasticNet model trained on the US Avocado Hass Sales dataset.

RandomForestRegressor Results

Figure 4.10 presents a summary of the median MAPE results for the 3000 RandomForestRegressor models following their initial fit and two subsequent retraining updates. This plot reveals a counter-intuitive finding, with the second retraining update displaying the highest median MAPE values. Like the other models, it also exhibits an upward degradation trend.

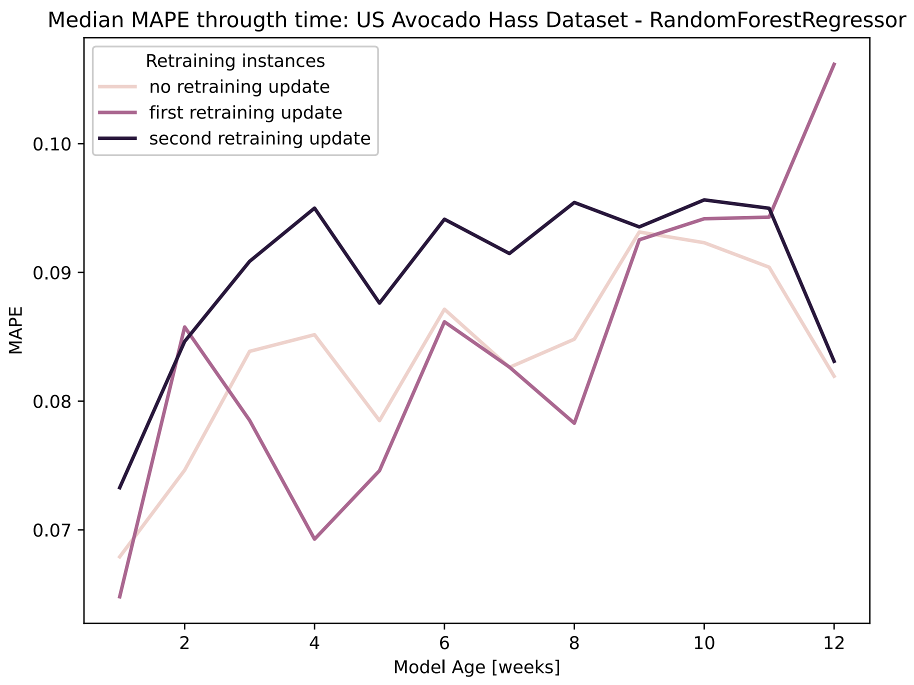

Figure 4.10: Model retraining plot of the RandomForestRegressor model trained on the US Avocado Hass Sales dataset.

MLPRegressor Results

Finally, Figure 4.11 provides a summary of the median MAPE results for the 3000 MLPRegressor models after their initial fit and two subsequent retraining updates. This plot exhibits more extreme behavior in terms of the difference between the no retraining and the second retraining update lines. While this result is not necessarily positive, it's noteworthy that the MLPRegressor stands out among our experiments. Unlike the other models, its median MAPE lines did not display a significant upward trend; they remained relatively constant throughout each retraining instance.

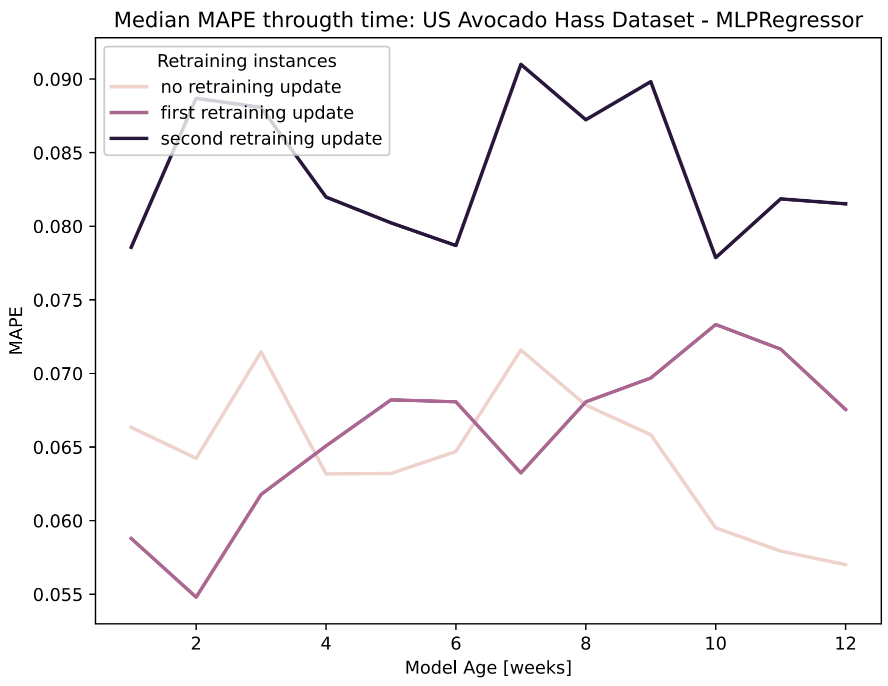

Figure 4.11: Model retraining plot of the MLPRegressor model trained on the US Avocado Hass Sales dataset.

4.2.2 Dataset: NYC Taxi Trip Demand

Similar to the Temporal Degradation Test, for each simulation, we utilized the NYC Taxi Trip Demand dataset and randomly selected a training set spanning 365 days, a testing set of 90 days, and a production set of 90 days. We conducted a total of 1500 model-dataset pair simulations and executed 2 retraining steps following the initial fitting. Each retraining step followed the procedure described in Section 2.2. Additionally, we performed hyper-parameter tuning for the initial fittings of all 1500 simulations. The subsequent two retraining steps utilized the discovered hyperparameters.

LGBMRegressor Results

Figure 4.12 presents a summary of the median MAPE results for the 1500 LGBMRegressor models following their initial fit and two subsequent retraining updates. The median MAPE remains relatively consistent over time between retraining instances. Around day 40 of the second retraining update, the median MAPE tends to be lower than the rest. However, in general, all three trends remain relatively stable. As mentioned earlier, this experiment involved utilizing 365 days of training data, 90 days for testing, and an additional 90 days for production data. Consequently, the models were retrained every 90 days using the latest 90 days of available data.

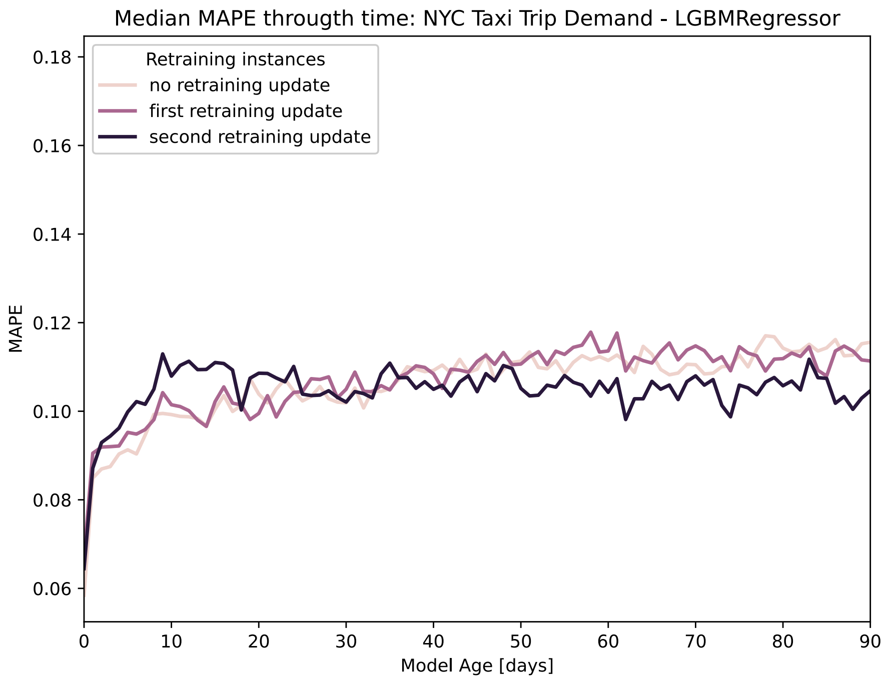

Figure 4.12: Model retraining plot of the LGBMRegressor model trained on the NYC Taxi Trip Demand dataset.

ElasticNet Results

Figure 4.13 provides a summary of the median MAPE results for the 1500 models. This result exhibits greater variability than the previous one and follows a counterintuitive pattern: the models with no retraining and those with the first retraining instance behave very similarly, while the second retraining instance shows even more degradation. This suggests that retraining alone is insufficient for maintaining model performance within acceptable bounds.

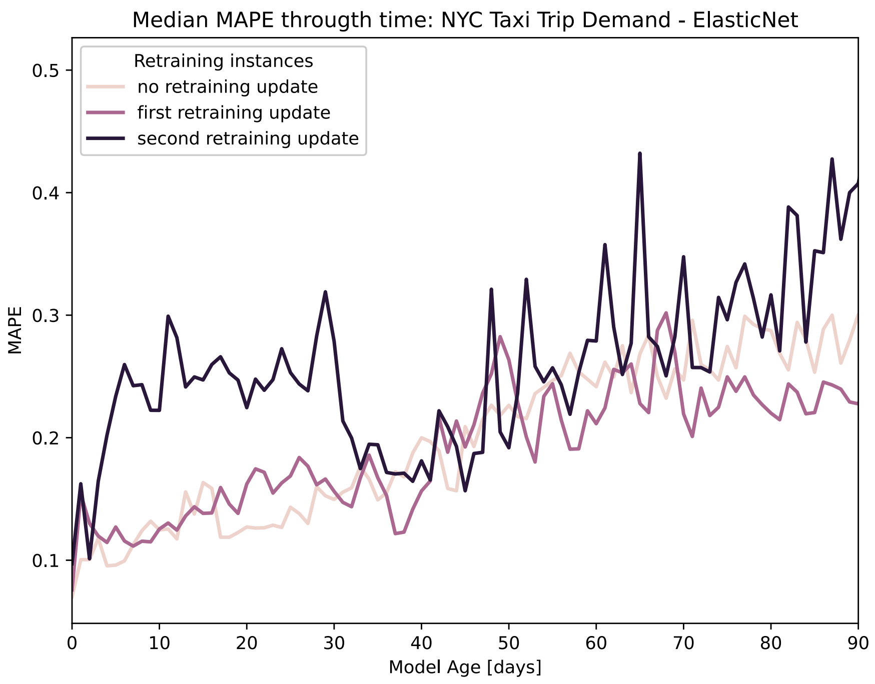

Figure 4.13: Model retraining plot of the ElasticNet model trained on the NYC Taxi Trip Demand dataset.

RandomForestRegressor Results

Figure 4.14 provides a summary of the median MAPE results for the 3000 RandomForestRegressor models after their initial fit and two subsequent retraining updates. The plot displays very similar patterns across retraining instances, with no clear indication that retraining is significantly improving or degrading performance. Although there is a slight upward trend in the first 30 days, the performance trend stabilizes thereafter.

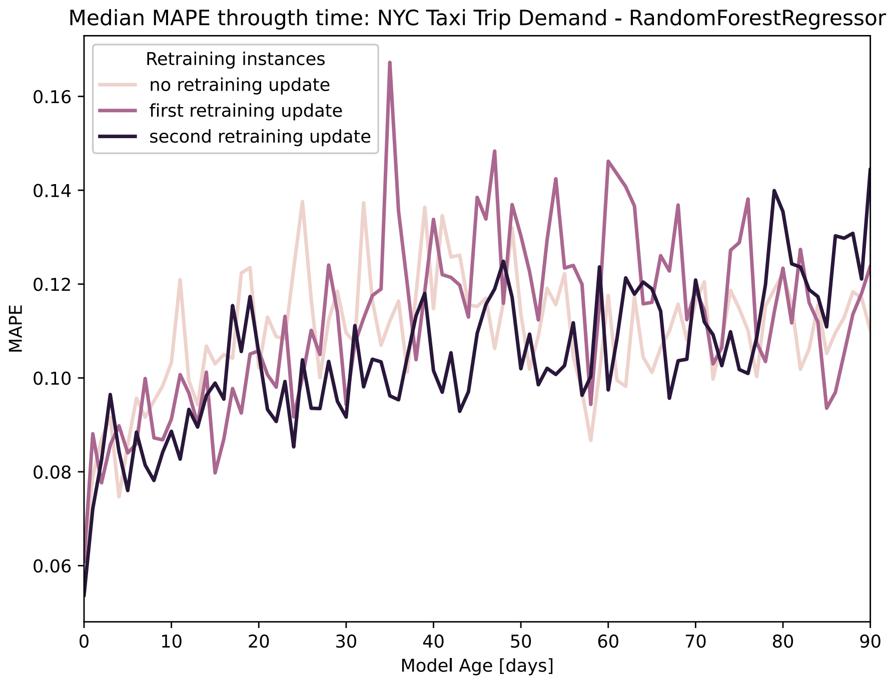

Figure 4.14: Model retraining plot of the RandomForestRegressor model trained on the NYC Taxi Trip Demand dataset.

4.3 Performance estimation results

In this section, we analyze the results of the Performance Estimation Test described in Section 4.3. To do this, we leverage the results obtained from the Temporal Degradation Test and assess whether a performance estimation method such as DLE, short for Direct Loss Estimation, could accurately estimate the observed degradations.

We applied this analysis only to models with a good initial fit (valid models). For each simulation, DLE was fitted with the simulation's respective reference set, which consisted of the simulation's test set and the latest available data before the model was put into production. We then performed DLE estimations on the simulation's respective production set.

Finally, we gathered the results and analyzed how many of the degraded models DLE managed to estimate as degrading without needing access to ground truth data. Since each prediction event is later associated with a degradation or no degradation outcome, we also measured how well DLE estimated the performance behavior of each prediction event. This means we were interested in understanding both the true positive rate (TPR) and the true negative rate (TNR).

Table 4.1 summarizes some of the results of the Temporal Degradation and the Performance Estimation tests. In the context of the US Avocado Hass Sales datasets, we see that LGBMRegressor had the highest number of valid models (2684 out of 3000), followed closely by RandomForestRegressor (2675 out of 3000). ElasticNet had all of its models (100%) classified as degraded.

When looking at a model-level analysis, we found that, on average, DLE was able to correctly estimate the degradation of 46% of the models that actually degraded. For the case of the LGBMRegressor, this means that out of the 2684 models that degraded, DLE estimated at least one degradation in 1177 of them.

On the other side, if we analyze the results from an alert level, we distinguish between two types of results. TPR, also known as Sensitivity or Recall, measures the proportion of truly degradation alerts correctly identified as degradation alerts. TNR, also known as Specificity, measures the proportion of truly non-degradation outcomes correctly identified as regular performance points.

The US Avocado Hass Sales dataset results tend to have a relatively low TPR and a high FNR. We attribute these results to two main conditions. DLE is fitted on a reference dataset that consists of the simulation's test set and its latest available data. In these simulations, the test set is fixed at 12 weeks, and the length of the latest available data is chosen randomly, as it precisely reflects the model's initial age. So, in some cases, this dataset might be 1 week or up to 80 weeks, meaning that most of the time, DLE might be fitted with a relatively small reference dataset. This makes it difficult for DLE to capture gradual degradations, as seen in the US Avocado Hass Sales dataset.

On the other hand, the NYC Taxi Trip dataset is quite large and granular, and the likelihood that a reference set is more than a year's worth of data is higher. Thanks to this, we can see an increase in effectiveness from DLE on the TPR. On average, DLE was able to correctly estimate 95% of the degradation alerts. This result is quite impressive for a method that only uses the model inputs and its predictions.

				DLE Analysis		
Dataset	Model Type	# Valid Models	% Degraded Models	Correctly Estimated the Degradation of	TPR	TNR
US Avocado Hass Sales	LGBMRegressor	2684/3000	2453 (91.4%)	1177 (48.0%) models	0.22	0.90
	ElasticNet	2423/3000	2423 (100%)	1009 (41.6%) models	0.12	0.95
	RandomForestRegressor	2675/3000	2492 (93.2%)	1144 (45.9%) models	0.20	0.88
	MLPRegressor	2045/3000	2032 (99.4%)	986 (48.5%) models	0.20	0.92
NYC Taxi Trip Demand	LGBMRegressor	1129/1500	1129 (100%)	1048 (92.8%) models	0.92	0.33
	ElasticNet	170/1500	170 (100%)	161 (94.7%) models	0.99	0.36
	RandomForestRegressor	1007/1500	1007 (100%)	961 (95.4%) models	0.95	0.32

Table 4.1: DLE Analysis Summary.

5 Conclusion

5.1 Summary of results

In summary, the findings validate several well-established insights in the field regarding the performance degradation of machine learning models [2]. We observed that different machine learning mathematical approaches can exhibit varying degradation patterns, particularly in terms of prediction error variability. The continuous retraining test demonstrates that refitting the model with more data tends to help but does not entirely resolve the degradation issue, as most of the error trend lines exhibited an upward tendency.

To address the challenge of performance degradation, performance estimation methods such as DLE prove to be valuable tools. DLE can serve as an early detection mechanism for identifying deteriorating model performance. By monitoring key performance indicators and assessing deviations from expected levels, we can intervene in a timely manner and identify the factors contributing to the performance problem.

5.2 Limitations

Expanding the scope of the present study to include classification tasks is a crucial step in gaining a better understanding of how machine learning models deteriorate across various prediction tasks. While the study's focus on regression tasks provides valuable insights, classification tasks present unique challenges and considerations.

5.3 Concluding words

Performance degradation effects will always be intrinsic to Machine Learning models, given that data continues to change, sometimes in unpredictable ways. To confront this challenge, we have emphasized the role of performance estimation methods. To serve as valuable early detection mechanisms for identifying declining model performance.

Furthermore, in a commitment to fostering collaboration and knowledge dissemination, we have made the source code of the core methods employed in this study accessible to the public [4]. This open-source initiative allows fellow researchers, practitioners, and enthusiasts to leverage our work as a foundation for their own endeavors. Whether it involves adapting our methods for different prediction tasks, exploring alternative datasets, or utilizing diverse machine learning techniques, our aim is to facilitate the broader exploration of performance degradation in machine learning.

References

[1] C. Huyen, *Designing Machine Learning Systems: An Iterative Process for Production-ready Applications*. O'Reilly Media, Incorporated, 2022. [Online]. Available: https://books.google.it/books?id=BAy_zgEACAAJ

[2] D. Vela, A. Sharp, R. Zhang, T. Nguyen, A. Hoang, and O. S. Pianykh, "Temporal quality degradation in AI models," *Scientific Reports*, vol. 12, no. 1, p. 11654, Jul. 2022, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-022-15245-z

[3] "Estimation of Performance of the Monitored Model — NannyML 0.9.1 documentation." [Online]. Available: https://nannyml.readthedocs.io/en/stable/how_it_works/performance_estimation.html

[4] S. Viquez, "santiviquez/ageml." [Online]. Available: https://github.com/santiviquez/ageml

[5] "sklearn.model_selection.TimeSeriesSplit." [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

[6] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[7] "lightgbm.LGBMRegressor — LightGBM 4.0.0.99 documentation." [Online]. Available: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html

[8] "sklearn.linear_model.ElasticNet." [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.linear_model.ElasticNet.html

[9] "sklearn.ensemble.RandomForestRegressor." [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[10] "sklearn.neural_network.MLPRegressor." [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

[11] "Hass Avocado Board." [Online]. Available: https://hassavocadoboard.com/

[12] "lightgbm.plot_importance — LightGBM 4.0.0.99 documentation." [Online]. Available: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.plot_importance.html

[13] "TLC Trip Record Data - TLC." [Online]. Available: https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

Acknowledgments

I want to thank Professor Bruno Scarpa for his assistance and guidance throughout the development of this study. And express my appreciation to the NannyML team for introducing me to the concepts of model performance estimation. Their knowledge and dedication are what set the foundations for this work.