Part I: Project Status Update

Project Name: DKomplex Intelligent Machine Learning Forecast Models

Team Members: Nicole Al-Sabah, Chika Starks, Deborah Shaw, Joshua Okero, Daniel Arias

Class: CST 499 Computer Science Capstone

Term: Summer 2025

The project started with the goal of building a forecasting model that helps DKomplex predict monthly sales quantities for their clients. Initially, the plan was to run AutoML experiments on cleaned invoice and economic data, evaluate model performance using metrics like R^2 and MAPE, and deliver a set of accurate predictions. Over time, we made some adjustments to the approach due to AutoML platform limitations, data formatting challenges, and model compatibility issues. For example, when using the AutoML UI, we originally considered several models (like ARIMA), but had to narrow our configuration after finding that some models were unsupported by Microsoft Fabric's FLAML integration. Additionally, AutoML caused bottlenecks in our project's progress due to the platform only allowing enough compute access to run one notebook at a time. The notebooks take at least an hour to two hours and prevent other team members from working on their tasks. Because of this, the team had to stay in constant communication and be mindful of notebook scheduling and runtime.

Currently, the project is going into its final stage. We've completed several forecasting runs using Microsoft Fabric AutoML, trained models on over 10 years of cleaned data, and generated 3, 6, and 12 month predictions. The data was merged from multiple sources, including invoice quantity history and global economic indicators, and underwent a full cleaning, formatting, and feature engineering process. We used Pandas, Scikit-learn, and PySpark for data transformation and evaluation, and stored results in Microsoft Lakehouse. The final models

and predictions were registered and saved using MLflow so they can be reused or improved later. One challenge was that the 12 month model gave a very low R^2 score, which might be due to noise or overfitting, but other runs (like the 6 month forecast) had better monthly quantity predictions. These results helped us refine our feature set and understand the trade-offs between short and longer-term forecasts.

Nicole contributed to data preprocessing, model configuration, using the AutoML UI, and debugging issues related to reporting, predictions, and model registration. Deborah contributed as project coordinator and worked on data engineering tasks and forecasting experiments. Chika focused on data featurization and forecasting model evaluations. Josh collected data, ran existing notebooks, and added new features to improve model performance. Daniel contributed by gathering data and running the notebooks to support feature validation and forecasting tests.

The team plans to review the final forecasts and the results that were organized into Excel reports. These reports will support the presentation and documentation deliverables. The project will not include specific client details due to the NDA we signed, but we will be able to provide valuable insights. Overall, we've made good progress toward our goal, learned a lot about Azure AutoML, and prepared a foundation for future students to build on. The code, models, and outputs are ready to be reused or improved by future students and the DKomplex team.

Part II: Project Testing

Introduction

Testing played a central role throughout our DKomplex Intelligent Machine Learning Forecast Models project. In addition to traditional model validation, much of our project's "testing" was performed through an extensive analysis process—specifically, by running forecasting experiments with different data configurations, model parameters, and filtering approaches. This iterative process was critical both for assessing the robustness of our pipeline and for understanding the sensitivity of our models to various project decisions.

Analysis Process and Testing Approach

Our testing strategy began with careful data examination and filtering. We started by cleaning invoice quantity histories and economic indicator datasets, removing incomplete records and irrelevant columns, and normalizing data types across sources. We tested different filtering thresholds for outlier removal and missing value imputation to see how the forecast results would be impacted. Each of these pre-processing variations was tracked and compared, with notes on how dataset "shape" affected model convergence and error rates.

Once data was merged and finalized, we designed a series of AutoML runs using Microsoft Fabric. Each run constituted a test of new parameters or feature engineering techniques. For instance, we experimented with:

- **Feature inclusion/exclusion:** We tested the effectiveness of adding lagged variables (prior month sales, rolling averages, economic leading indicators), and the impact of dropping less predictable features. For every change, we documented the effect on metrics like R^2 and MAPE.

- **Forecast windows:** We tested and compared 3, 6, and 12-month forecast intervals. This allowed us to observe the degradation in model performance as the prediction horizon increased, particularly with the 12-month window, which frequently gave the lowest R^2.
- **Data splits:** We validated models using both time-based and random splits to ensure that "look-ahead bias" was minimized, and to check whether the model overfitted to recent trends.
- **AutoML configuration:** We ran tests using different learner families (where supported), model tuning parameters, and varying runtime limits to see how automated search affected result quality.

Filters and Their Impact on Model Performance

Since forecasting models are highly sensitive to the quality and structure of the input data, we systematically tested a range of data filters and transformation steps. Examples include:

- **Date range filters:** Limiting data to more recent years improved accuracy on short-term forecasts but sometimes caused worse generalization for longer-term predictions.
- **Economic indicator selection:** Including or dropping certain macroeconomic variables measurably shifted forecast accuracy, especially for longer horizons.
- **Outlier treatment:** Removing or capping extreme sales spikes reduced noise and stabilized some model runs, but over-filtering occasionally diminished the signal needed for anticipating high-variance months.

Each filtering decision was validated by comparing pre- and post-filtering model results, which we tracked in an experiment log and Excel results files. In many cases, filtering and feature

engineering had a bigger effect on MAPE and R^2 than even changing model types within AutoML.

Results & Ongoing Validation

After each round of testing, we evaluated results not just by model scores, but by how sensibly the forecasts matched expected business patterns. Deborah, Nicole, and Chika handled model scoring and charted error distributions, while Josh and Daniel documented how subtle changes to pipeline steps could push the forecast in or out of practical accuracy bounds. This process of repeated experiment runs, result review, and parameter tuning constituted both our model validation strategy and our testing methodology.

Ultimately, the rigorous testing and filtering process led to identifying the most reliable feature set and forecast interval for our target use cases. The process also highlighted the trade-offs between model complexity and interpretability, ensuring that our results were not just statistically valid but also business-relevant.