Project
Report ML Ops
Group I

NOVA IMS 2023/2024

Master in Data Science

Students:

Carlos Costa, 20230543

Dzmitry Nisht, 20230776

Jannik Himmelsbach, 20230550

Mohamed Taha Ben Attia, 20230742

GitHub Repository:

dmitrynisht/bcase2-sales-forecasting (github.com)

https://github.com/dmitrynisht/bcase2-sales-forecasting

Index

Project Scope

The main objective of this project was to apply the MLOps principles and techniques we studied during this course to a project completed earlier in the semester. Our goal is to transition from static development in a single Jupyter notebook to a more sophisticated, robust project structure with the potential for production deployment. This project acts as a proof of concept, showcasing the feasibility of applying MLOps practices in a real-world context.

The original project aimed to forecast sales for various products of the company Siemens. Siemens provided different training datasets, including daily sales data of their products, monthly market data from various sectors and countries they operate in. During the project development an additional data on the German GDP was obtained. The challenge was to implement effective models to forecast the sales data of different products, which was measured against provided test data (future dates and sales figures). The final metric for evaluation, given by Siemens, was the Root Mean Square Error (RMSE) with lower RMSE values indicating better model performance.

Workflow

For project planning, we organized our workflow using Git, allowing us to work simultaneously on different branches to develop various pipelines. Each team member focused on specific aspects of the project, ensuring parallel development. Before merging a feature (pipeline) into the main branch, we tested it individually to ensure functionality and reliability, aligning with continuous integration principles. We maintained regular communication to effectively combine individual outputs, merging our work into the main branch in a smooth and efficient way.

Tools

We employed a diverse set of tools to optimize our workflow and ensure project success. For version control and collaboration, we relied on Git and GitHub, which enabled us to manage and merge different branches seamlessly. Python, our primary programming language, supported the development of various libraries and frameworks along with Pytest in order to build and run Unit Tests. Kedro was instrumental in providing modular code organization and pipeline orchestration, which helped maintain a well-structured codebase. To ensure data quality and integrity, we utilized Great Expectations for data unit testing. Hopsworks served as our feature store, making feature management and reuse efficient. MLflow played a key role in tracking model training, allowing us to monitor and document our experiments thoroughly. Finally, Docker was used for containerization, ensuring a consistent and reproducible environment for eventual production.

Implementation

During the implementation phase of the project, we followed a systematic approach, starting from data unit testing and ingestion to containerization. Each stage was designed to ensure

the integrity, efficiency, and scalability of our solution. The following sections will briefly discuss each of these stages in terms of their purpose and functionality.

Although the Siemens datasets contain data for many kinds of products, the data is logically split by product because the models need to forecast sales for a specific product. Our pipeline is designed to handle one product at a time, but it is parameterized to accept any product code, making the code reusable for different products.

Some steps in our pipelines, such as data preprocessing and data ingestion, include unit tests to ensure that data transformations are performed correctly and do not silently break subsequent pipeline steps..

Data Unit Tests

The initial stage of our pipeline focuses on data validation to ensure the integrity and reliability of our datasets. We implemented a data unit testing function that leverages the Great Expectations library. This function validates the raw sales and market data by checking for specific column counts and data types. For the raw sales data, we confirmed that there are exactly three columns with names as expected. Similarly, for the raw market data, we verified that the dataset contains 48 columns. Successful validation is logged, ensuring that only accurate and well-structured data progresses to the next stages of our pipeline.

Data Ingestion

The next stage in our pipeline focuses on data ingestion, integrating and preprocessing raw data to ensure it is ready for analysis. For each of our datasets - "sales_raw_data," "market_raw_data," "german_gdp_raw_data," and "test_raw_data" - we performed tasks such as renaming columns and sanitizing column names to ensure consistency and conformity to Great Expectation requirements. Formatting dates was done once during ingestion, and Kedro supports the seamless flow of data, preserving all data types. We then validated the data using Great Expectation suites and stored the processed data in Hopsworks, our feature store. The output datasets are "ingested_sales," "ingested_markets," "ingested_german_gdp," and "ingested_test_data."

Data Preprocessing

The subsequent stage in our pipeline is data preprocessing, where we transform and clean the ingested data to ensure it is in an optimal format for analysis and modeling. For each dataset, we performed type conversions. For "ingested_sales," we carried out tasks such as grouping and outlier removal. The "ingested_markets" and "ingested_german_gdp" datasets were merged and resampled. This consolidation and optimization process resulted in the "preprocessed_sales" and "processed_markets" datasets.

Feature Engineering and Selection

The next step in the pipeline was feature engineering and selection. We began by removing columns from the "processed_markets" dataset that had a correlation greater than 0.95 with each other. Then, we created features based on the lag of macroeconomic (market) values, selecting the most significant lag period by examining the correlation with sales data. Specifically, we created buckets of 1-month lag periods and considered only market data with a correlation of at least 0.1 with sales. This step was crucial because not all macroeconomic events significantly impact sales and thus are not useful for the model.

Finally, we used a tree-based model (Extreme Gradient Boosting) to identify the 10 most important features for predicting sales. At the end of this step, only the most important lag features and the 10 key features were stored.

Prepare Model Input Data

The purpose of the "prepare model input data" pipeline is to transform the preprocessed data and selected features into suitable formats for model input. This step generates multiple datasets essential for different stages of model development and evaluation, specifically tailored for time series models. The datasets created include:

- **Training and Validation Sets**: These are used to test and choose the best model.
- **Full Training Set**: This set includes all the training data and is used to build the final model for production.
- **Test Set**: This set is used to make predictions and see how well the model performs on new data.

In this pipeline, we focus on making sure the data is formatted correctly for time series models. We only include the essential columns: the date and the target values. Due to time constraints and the simplicity needed for this proof of concept, we haven’t included additional features created in earlier steps. However, the pipeline is designed to be easily scalable, so more features can be added later as needed.

Model Selection

The model selection pipeline's purpose is to identify the best-performing model for the given time series data. It takes the training and validation datasets, along with specific parameters, to train multiple models and evaluate their performance. This process involves comparing different models, such as NeuralProphet and FB Prophet, by training them and assessing their validation performance using metrics like RMSE. The pipeline logs model training details and scores using MLflow for tracking experiments. After evaluating all models, it selects the model with the best performance, providing the best model and its parameters for further use.

Model Training

The purpose of the model training pipeline is to finalize the training of the best-performing model by utilizing the full training data, preparing it for deployment. The selected champion model is then reinitialized, and fitted with the complete training data. The trained model, along with its parameters, is logged and stored, ensuring that the optimal, fully trained model is ready for future forecasting tasks.

Model Prediction

The model prediction pipeline is designed to generate forecasts using the trained production model to generate future predictions.The performance of the model is evaluated by calculating the RMSE between the actual and predicted values. The results include a DataFrame of predictions, a dictionary containing the RMSE score, and a plot comparing actual versus predicted values.

Data Drift

Last but not least, we implemented a data drift detection pipeline to monitor changes in our data over time. This pipeline uses the Kolmogorov-Smirnov test to compare our preprocessed feature data with the prepared model input datasets for the production model. This ensures that the input of the production model is aligned with our target product input; otherwise, preprocessing would need to be executed again. By focusing on specific features, it can identify significant changes that might affect model performance. The detection process involves extracting the relevant feature columns from both datasets and applying the KSDrift detector. If a drift is detected, a warning is logged with detailed information about the feature and p-value, ensuring prompt attention. Otherwise, it logs that no drift was found. The results are then compiled into a structured format for further analysis.

Docker Support

To make deployment easier and ensure everything runs the same way everywhere, we offer Docker support. We include a Dockerfile to build the application container and a .dockerignore file to keep sensitive data out of the Docker image. This setup helps you deploy and run the pipeline in any environment that supports Docker, while keeping your data secure.

Results and Conclusions

In conclusion, this project successfully applied MLOps principles to transition from a simple Jupyter notebook to a robust and sophisticated project structure ready for production. Our main goal was to forecast sales for various Siemens products using multiple datasets. We designed our pipeline to handle different stages of data processing and model training smoothly, ensuring a seamless workflow from raw data ingestion to model prediction.

The project pipeline was broken down into several key parts: raw data tests, data ingestion, preprocessing, feature selection, model input preparation, model selection, model training,

model prediction, and data drift detection. Furthermore, to streamline our workflow, we created several connected pipelines:

- **data_preparation_pipe**: This pipeline manages the entire data preparation process, starting from raw data and ending with model input. It includes steps for raw data tests, data ingestion, preprocessing, feature selection, and preparing model input data.
- **production_model_pipe**: This pipeline checks if the data is valid and hasn't changed (data drift), then trains and predicts with the production model.
- **model_selection_pipe**: This pipeline focuses on choosing the best model (the champion model) and training it for production. It includes model selection, training, and prediction steps.
- **complete_pipe**: This comprehensive pipeline runs all stages, from raw data to producing the final model output. It combines all the individual pipelines to ensure a full end-to-end process.

We parameterized the entire setup, making it easy to adjust and make predictions for different products. Additionally, the report includes a plot showing the actual sales values versus the predicted values for Product 1 (ANNEX).

Suggestions for Improvement

While our pipeline works well for processing and preparing data for sales forecasting, there are some areas we could improve.

The “Data Preprocessing” pipeline

The “Data Preprocessing” pipeline and its outlier removal step is very case-specific and in this terms it is not resilient and scalable.

The “Feature Engineering and Selection” pipeline

The "Feature Engineering and Selection" pipeline could be improved to be robust enough to handle any incoming groups of products, processing them in the same way. Alternatively, it could be designed to focus on a predefined list of 'major' product groups, ignoring all others. Either way, it should be more explicit about which product groups are being considered by the pipeline.

Hyperparameter Tuning

Another thing we didn't do was hyperparameter tuning, which means adjusting the settings of our models to get the best performance. We kept things simple for this project, but adding hyperparameter tuning in the future could make our models more accurate.

List of the packages and versions

- All packages and versions are documented in the requirements.txt file.

Appendix

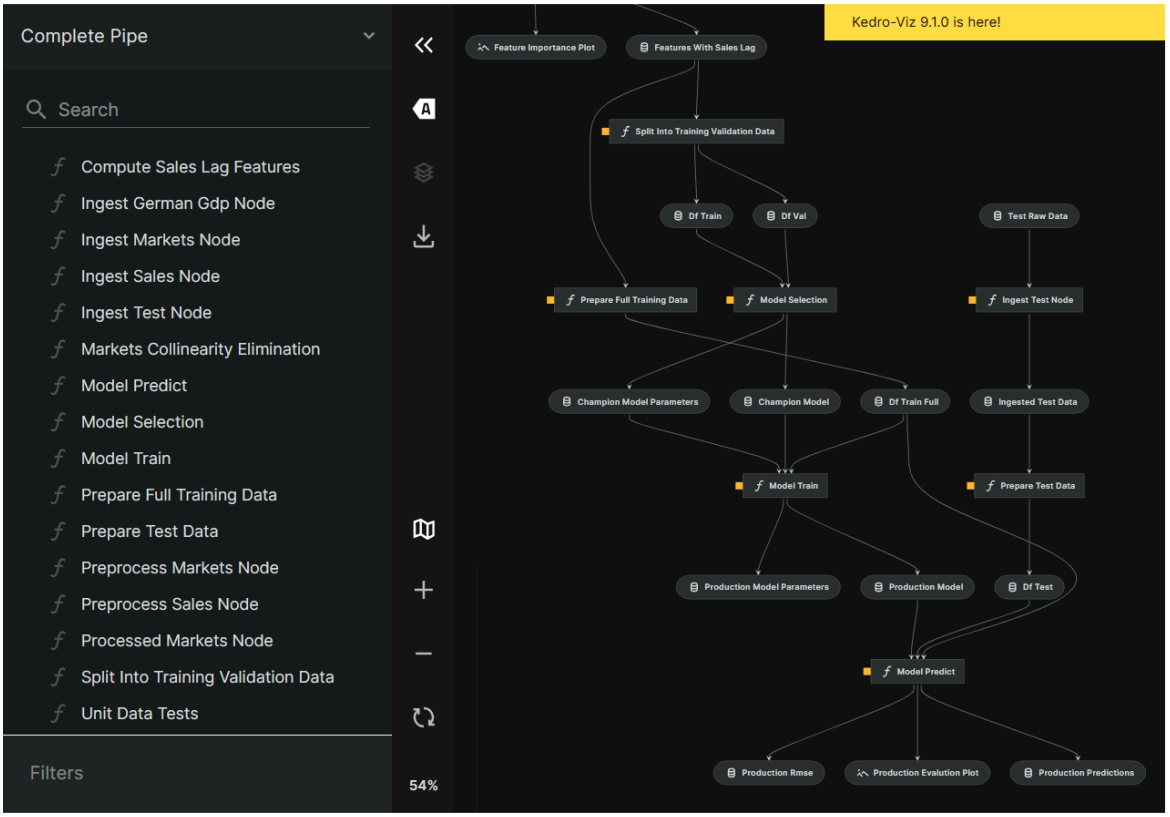


Figure - Visualised pipeline structure (kedro viz). Partial sequence of several pipelines

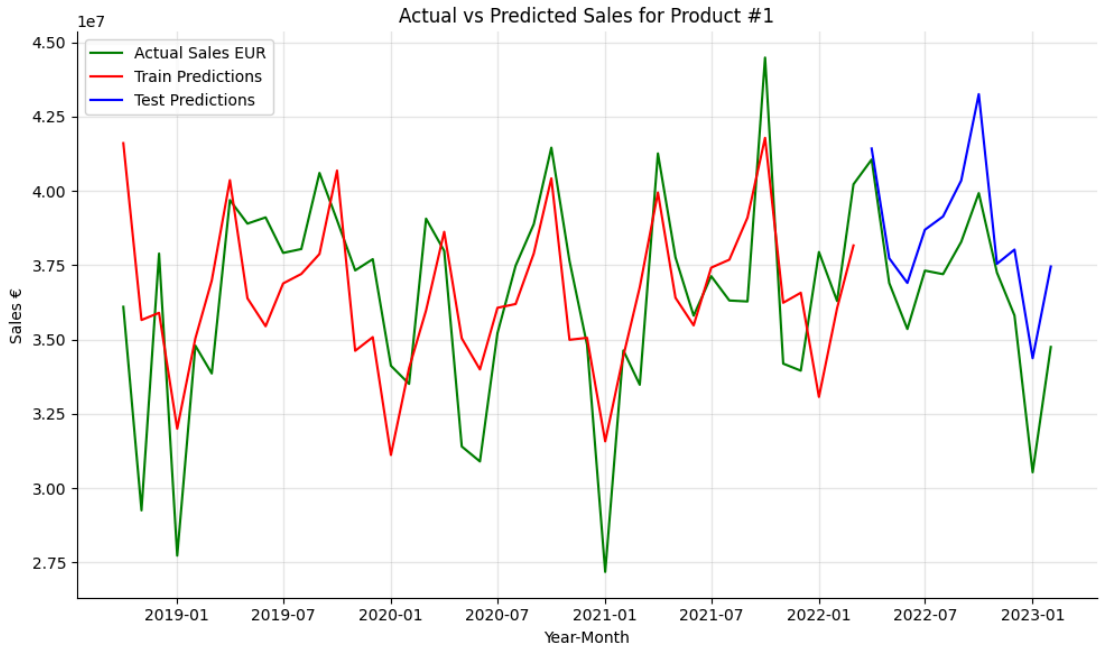
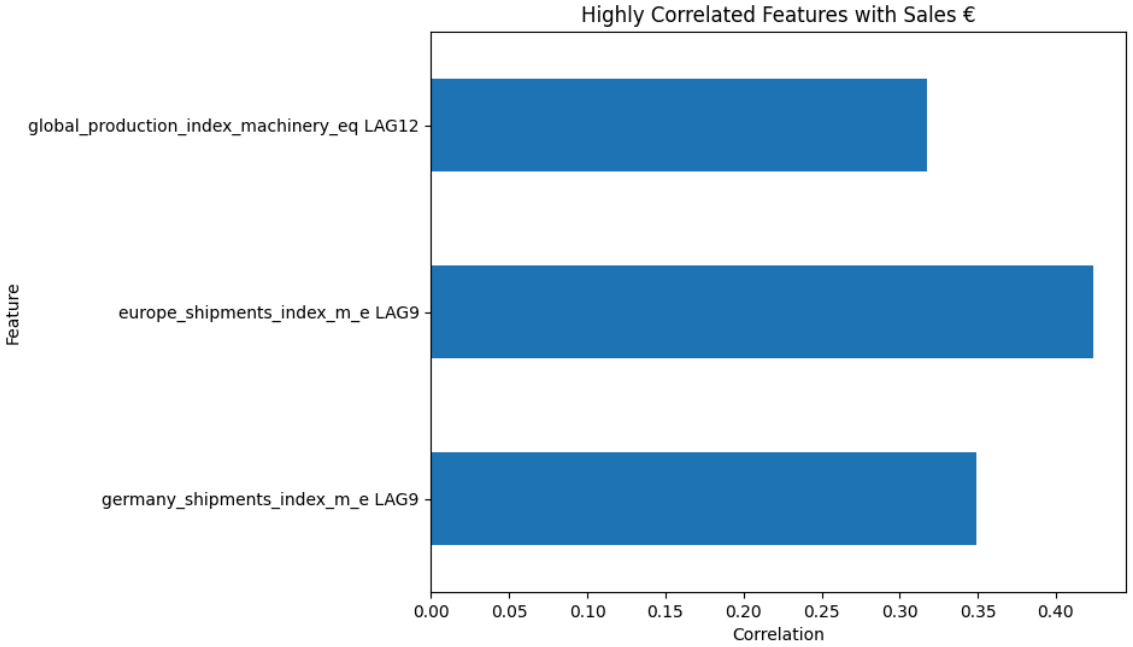
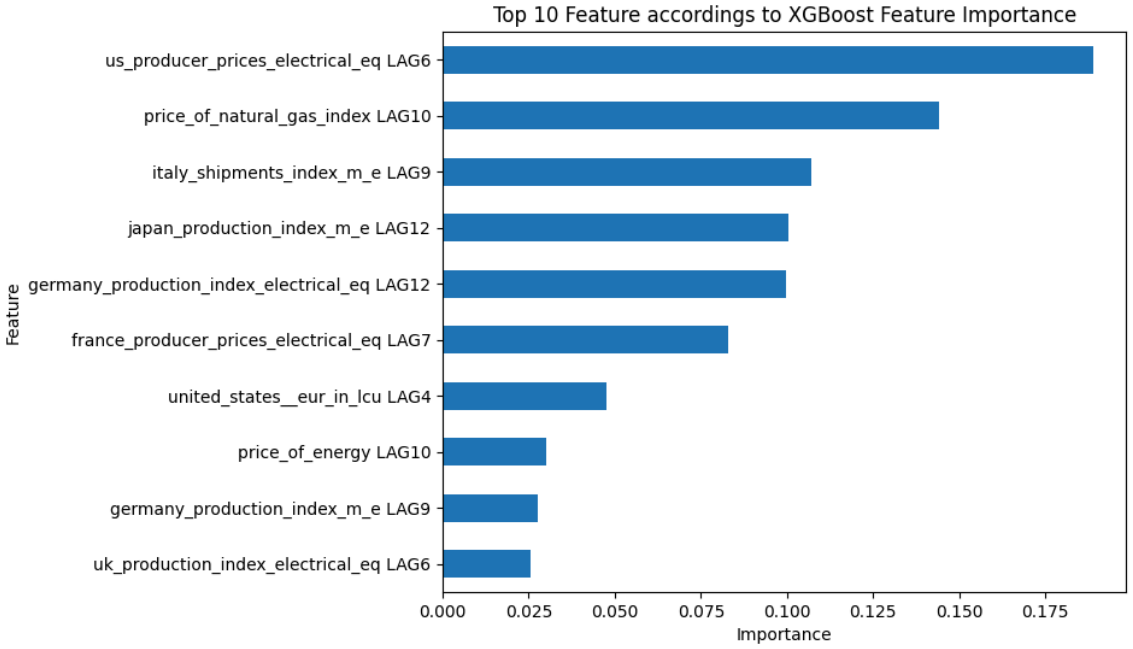


Figure - Final Results of Production Model for Product 1.


Highly Correlated Features with Sales €
global_production_index_machinery_eq LAG12
europe_shipments_index_m_e LAG9
germany_shipments_index_m_e LAG9
Feature
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Correlation


Top 10 Feature accordings to XGBoost Feature Importance
us_producer_prices_electrical_eq LAG6
price_of_natural_gas_index LAG10
italy_shipments_index_m_e LAG9
japan_production_index_m_e LAG12
germany_production_index_electrical_eq LAG12
france_producer_prices_electrical_eq LAG7
united_states__eur_in_lcu LAG4
price_of_energy LAG10
germany_production_index_m_e LAG9
uk_production_index_electrical_eq LAG6
Feature
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
Importance