Joakim Abdinur Iusuf

Sales Project Report

Note: I think the notebook gives a better idea of my thought processes and how I approached the test. But I've tried to summarize the most important things.

Initial thoughts

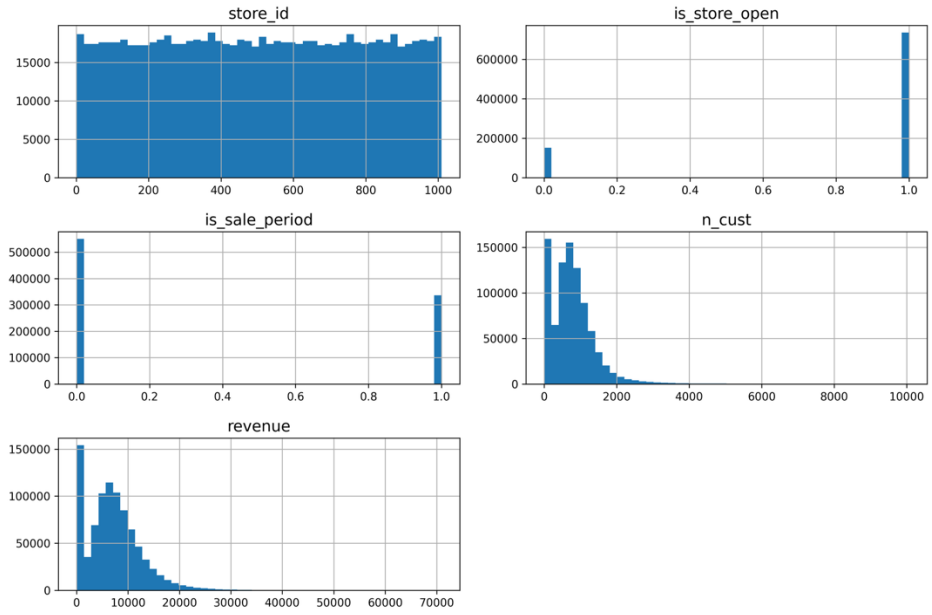
This is a supervised learning task, since we can train the models with labelled examples. It's also a typical regression task, since the models will be asked to predict a value. Specifically, it's a multiple regression task (we use multiple features to make a prediction), and it's also a multivariate regression problem since we are predicting both the number of customers and the sales figures for each store.

I will use root mean square error (RMSE) as the performance measure. However, if there are many outlier stores, I might use the mean absolute error (MAE, also known as the average absolute deviation). When outliers are exponentially rare (as in a bell-shaped curve), RMSE performs very well and is generally preferred.

Initial assumptions:

- Sales likely follow seasonal/temporal patterns (such as weekend vs weekdays, and end-of-month effects when people get their salary, although this varies from country to country).
- Performance will likely vary by store/location.
- The number of customers and the revenue should be 0 when is_store_open = 0 and should increase when is_sale_period = 1.
- Modelling assumptions: Historical patterns will continue into the future, store-specific effects are relatively stable, and external factors that are not in the data remain relatively constant.

Exploring the data



In the notebook, we can see that the outcome variables follow a right-skewed distribution. This suggests we might want to use a tree-based model such as XGBoost, since it works with any distribution (as it just tries to find the best splits regardless of the shape of the data) and we would not need to do any transformations. Linear models perform better when we have normally distributed data. I could potentially try linear models after log transformations. The shape of the data also suggests using Mean Absolute Error (MAE), since RMSE is more sensitive to outliers. RMSE can perform better when outliers are exponentially rare. But I'd argue that they occur fairly frequently (we have a lot of 0 values, and a lot of high values).

Creating a test set

To avoid data snooping bias, I'm creating the test set at this stage. Data snooping bias is the idea that we could peek at the test set, find interesting patterns, and make decisions that lead to overfitting (e.g., choosing a particular model).

We are stratifying the test set by town. The idea is that we want a test set that is representative of the overall data. I considered stratifying by store_id, but they appear more or less an equal amount of time in the dataset (see the notebook). However, towns are unevenly distributed, with Dukem appearing 407 155 times and four out of twenty stores only appearing 908 times (presumably there is one store in each of those small towns, which appear for each day in sales_data.csv). There's a similar distribution in sales_data_forecast.csv. Random sampling risks completely missing smaller towns in the test set. To make sure that the test set I've created from the sales_data file is representative of the data in the forecast file, I'm stratifying based on town.

Feature Engineering and Preprocessing

I created temporal features and applied one-hot encoding to the town variable (20 categories), and used StandardScaler on all numerical features, resulting in 27 total features. No transformation was applied to the outcome variables since they were zero-inflated, so I opted for tree-based models, although I tried linear regression.

Model Selection and Performance

I tested Linear Regression (baseline), Random Forest, and XGBoost (selected as final model). I chose Mean Absolute Error (MAE) over RMSE due to right-skewed target distributions with frequent outliers and zero-inflation, as explained above. I used 3-fold cross-validation. The tree-based models performed better. Due to time constraints, I only did CV and fine-tuning on XGBoost; I explain more in the notebook.

Hyperparameter Tuning and Validation

GridSearchCV with 3-fold CV (which is how I split the development set into training and validation sets): n_estimators = [50,100], max_depth = [10,20], and

learning_rate = [0.1,0.5]. Best parameters were learning_rate = 0.5, max_depth = 10, n_estimators = 50. Performance of the test set: customers MAE 181.3 (vs 182.5 CV) and revenue MAE 1484.8 (vs 1503.4 CV).

Model Tracking and Implementation

See the notebook to see how the different models performed and how they progressed. Final models (XGBoost) were retrained on all available data using the best hyperparameters, which were used to predict n_cust and revenue in the forecast CSV.

What could be improved

- If I had more time, I would've tried more models. If I had even more time, I would've maybe analyzed the strengths and weaknesses of each model and used an ensemble method.
- More extensive fine-tuning to improve model performance.
- More feature engineering, maybe by combining more features.