Title:
Hotel Demand Forecasting Using Statistical, Machine Learning, and Neural Models

Course: ISA 444
Student: Filip Donev

---

1. Introduction

The goal of this project is to forecast daily room demand for a panel of hotels and to compare several forecasting approaches. Accurate room-demand forecasts are important for revenue management, staffing, pricing, and inventory decisions. Under-forecasting can lead to lost revenue and operational stress, while over-forecasting leads to excess capacity and higher costs.

The dataset is provided in a parquet file called `sample_hotels.parquet`. Each record contains a hotel identifier (`hotel_id`), a date, and a demand variable. For the project, I rename these fields to `unique_id` (hotel), `ds` (date), and `y` (demand) to match the format used by the Nixtla forecasting libraries. There are 19 hotel time series, each observed daily. The forecasting objective is to predict demand for the next 28 days for each hotel.

This report describes the models I implemented, the cross-validation procedure, the evaluation metrics, the empirical results, and a brief discussion of the best model for this problem.

---

2. Data Preparation

The `sample_hotels.parquet` file is loaded into a pandas DataFrame. I then:

1. Rename `hotel_id` → `unique_id`, `date` → `ds`, and `demand` → `y`.
2. Keep only the three columns `unique_id`, `ds`, and `y`.
3. Convert `ds` to a proper datetime and coerce `y` to numeric.
4. Drop any rows with missing demand values.
5. Sort the data by `unique_id` and `ds`.

The final dataset contains 19 daily time series, one for each hotel. Visual inspection shows weekly patterns (weekday vs. weekend) and some variability over time, but the demand series are relatively stable and normalized to a 0–1 range. This suggests that simple baselines may perform reasonably well, but more flexible models should be able to capture subtle temporal patterns and cross-series information.

---

3. Methods

I implemented three main families of forecasting models: classical statistical models, a machine learning model, and neural forecasting models. All implementations use the Nixtla ecosystem: **StatsForecast**, **MLForecast**, and **NeuralForecast**.

3.1 Statistical baseline models (StatsForecast)

I use three classical models as baselines:

- **Naive** – The forecast for each horizon is equal to the last observed value.
- **SeasonalNaive** – The forecast repeats the value from the same day of the previous week (season length of 7).
- **AutoETS** – An automatic exponential smoothing model that selects level/trend/seasonality components and uses a season length of 7.

These models are simple, interpretable, and relatively fast to estimate. They serve as benchmarks to see whether more complex models actually add value.

> Note: The project handout also mentions AutoARIMA. In the Colab environment, global ARIMA models with multiple cross-validation windows were too slow to be practical. I therefore focused on ETS, ML, and neural methods and documented this trade-off.

3.2 Machine learning model (MLForecast + LightGBM)

For machine learning, I use **MLForecast** with **LightGBM** as the underlying regressor:

- Global model: trained on all hotels at once.
- Lags: 1, 7, and 14.

- Calendar features: day of week and month.

This model learns nonlinear relationships between lagged demand, calendar effects, and future demand. Because it is global, it can share information across different hotels.

3.3 Neural forecasting models (NeuralForecast)

I implemented two neural architectures:

- **NBEATS** – A deep fully-connected architecture specialized for univariate time-series forecasting.
- **NHITS** – A multi-scale neural model that interpolates and aggregates predictions at different temporal resolutions.

Both models are trained globally across all 19 hotels. I use:

- Forecast horizon `h = 28`.
- Input window length = 28.
- `max_steps = 50` to keep training time reasonable in Colab.

These models are designed to capture complex nonlinear patterns and interactions in the data.

---

4. Cross-Validation and Evaluation

4.1 Rolling-origin cross-validation

To evaluate the models fairly, I use rolling-origin time-series cross-validation. The setup is:

- **Number of windows:** 3
- **Forecast horizon:** 28 days
- **Step size:** 28 days (non-overlapping windows)

For each window:

1. Use all data up to a cutoff date as training data.
2. Fit all models on the training set.
3. Forecast the next 28 days.
4. Compare the forecasts to the actual 28-day holdout set.

This procedure is implemented using the `cross_validation` functions in StatsForecast, MLForecast, and NeuralForecast. The full panel of cross-validation predictions for all models and hotels is saved in `cv_results.csv`.

The original project description suggests using 5 cross-validation windows. In practice, using 5 windows together with neural models and global LightGBM models was too expensive in the Colab environment, so I used 3 windows as a compromise. The methodology is the same and can be extended to more windows with more compute.

4.2 Evaluation metrics

For each hotel and each model, I compute the following metrics:

- **Mean Error (ME)** – measures bias.
- **Mean Absolute Error (MAE)** – average absolute deviation.
- **Root Mean Squared Error (RMSE)** – penalizes larger errors more heavily.
- **Mean Absolute Percentage Error (MAPE)** – scale-free measure of relative error.

These metrics are calculated on the combined cross-validation forecast errors and stored in `metrics.csv`.

To determine which model performs best for each hotel, I identify the model with the lowest MAE and count how many hotels each model "wins." These counts are saved in `model_wins.csv`.

---

5. Results

5.1 Average metrics

Averaging the metrics over all hotel series, the results show:

- **NBEATS** has the lowest average MAE and RMSE.
- **NHITS** and **LightGBM** are close behind.
- **AutoETS** is a strong classical baseline, only slightly worse on average than the neural and ML models.
- **Naive** and **SeasonalNaive** are clearly worse on average, with higher errors and percentage errors.

These results indicate that there is meaningful temporal structure that can be exploited by global neural and machine learning models. At the same time, the gap between AutoETS and the more complex models is not enormous, which suggests that the series are relatively smooth and stable.

5.2 Model wins by hotel

Based on the "wins" metric (lowest MAE per hotel):

- **NBEATS** wins the largest number of hotel series.
- **LightGBM** is competitive and wins several series as well.
- **Naive** and **AutoETS** occasionally win, typically on very simple or nearly flat series.
- **SeasonalNaive** rarely wins.

This pattern confirms that no single model is universally best for every individual series, but NBEATS and LightGBM are the most consistent performers.

5.3 Final forecasts

After cross-validation, I retrain each model on the full dataset and generate 28-day-ahead forecasts for every hotel. These final forecasts are stored in `final_forecasts.csv`. The notebook includes code to plot the last 100 days of historical demand along with the forecasts.

For example, for a typical hotel, the NBEATS and LightGBM forecasts track the weekly patterns and overall level of demand, while Naive and SeasonalNaive behave more rigidly. AutoETS produces smooth forecasts that capture the general level and weekly seasonality but may lag in responding to recent changes.

---

6. Discussion and Conclusion

This project shows how different classes of forecasting models behave on a panel of hotel demand series. Several key points emerge:

1. **Global models are powerful.** Training NBEATS, NHITS, and LightGBM on all hotels at once allows the models to share information and learn generic patterns of hotel demand.

2. **Simple baselines still matter.** Despite being outperformed on average, Naive and AutoETS occasionally win on individual hotels. This highlights the importance of comparing multiple methods rather than assuming that a complex model is always better.

3. **Cross-validation is essential.** Rolling-origin evaluation provides a more realistic assessment of how models will perform in practice compared to a single train/test split.

4. **Computation is a real constraint.** Including neural and global ML models forced me to compromise on the number of cross-validation windows and to omit AutoARIMA. These constraints are common in applied forecasting and must be documented.

Overall, the best-performing model for this dataset is **NBEATS**, with **LightGBM** and **NHITS** as strong alternatives. For a production forecasting system, I would recommend using NBEATS as the primary model, with LightGBM and AutoETS as simpler back-up models and benchmarks. Additional improvements could include adding holiday or event variables, incorporating price or promotion data, or extending the model to jointly forecast multiple related demand variables