Machine Learning Challenge Paris-Dauphine University Master 203

Charles Chou
https://github.com/charleschou99/sncfChallenge.git

April 5, 2025

Contents

Chapter 1

Data Visualization and Outlier Detection

<visualisation.ipynb>
In this study, we undertook a comprehensive exploratory data analysis (EDA) to understand the distribution and relationships among the various features of our dataset. We visualized quantitative features—such as `p0q2`, `p3q0`, and `p4q0`—using histograms and kernel density plots, while categorical features like `gare` and `train` were examined through frequency plots and scatter plots. These visualizations provided essential insights into the data structure, allowing us to identify potential anomalies and distributional irregularities.

Outlier Detection Methodology

<outlier_detection.ipynb>
Our approach to outlier detection was executed using 3 methods:

1. **PCA-Based Outlier Detection:**
   We first applied Principal Component Analysis (PCA) on the quantitative features to reduce dimensionality and capture the major variance within the data. The first principal component, which explained a significant portion of the total variance, was then used to identify observations that deviated markedly from the main data cluster. These deviations were flagged as potential outliers.

2. **Outlier Filtering by Train feature:**
   Recognizing that an outlier within a particular `train` could compromise the integrity of the entire observation, we proceeded to remove any `train` that contained at least one outlier, as detected by the PCA analysis.

3. **Winsorization of Quantitative Features:**
   Finally, to mitigate the influence of extreme values, we applied winsorization to the quantitative features. Specifically, we trimmed the left tail at $x\%$ and the right tail at $x\%$, thereby capping the extreme values while preserving the overall distributional shape.

   This multi-step process enabled us to effectively visualize, detect, and mitigate outliers, thereby enhancing the reliability and robustness of our modeling and further data analyses.

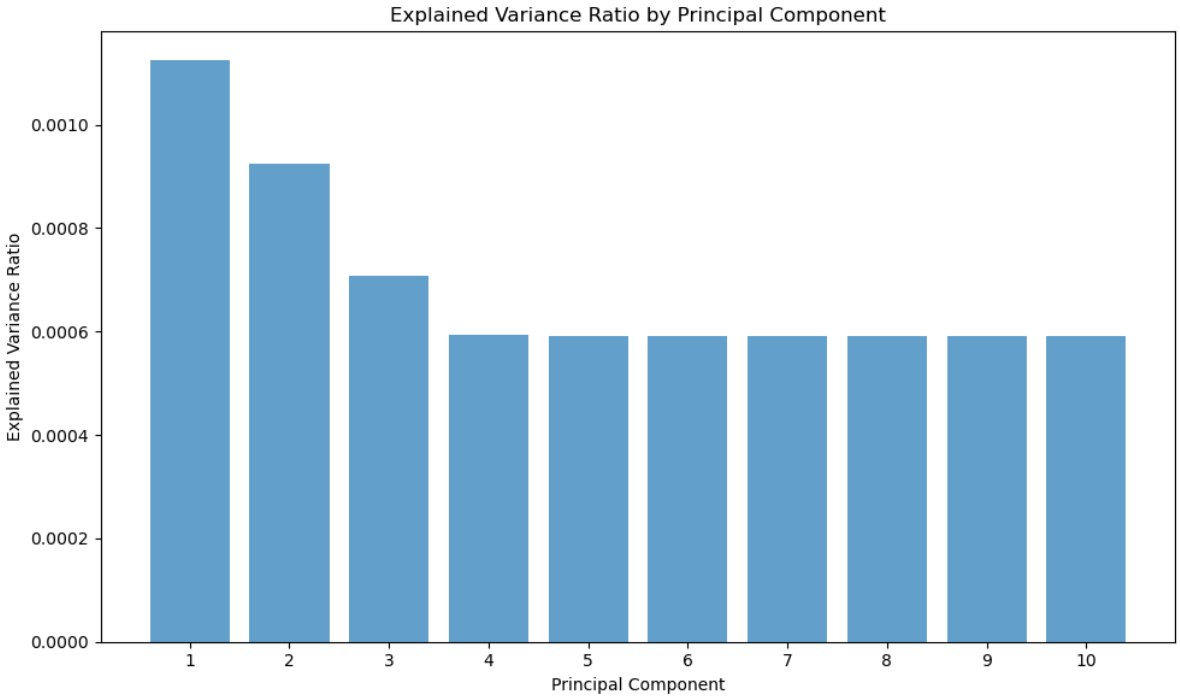

Figure 1.1: PCA explained variance

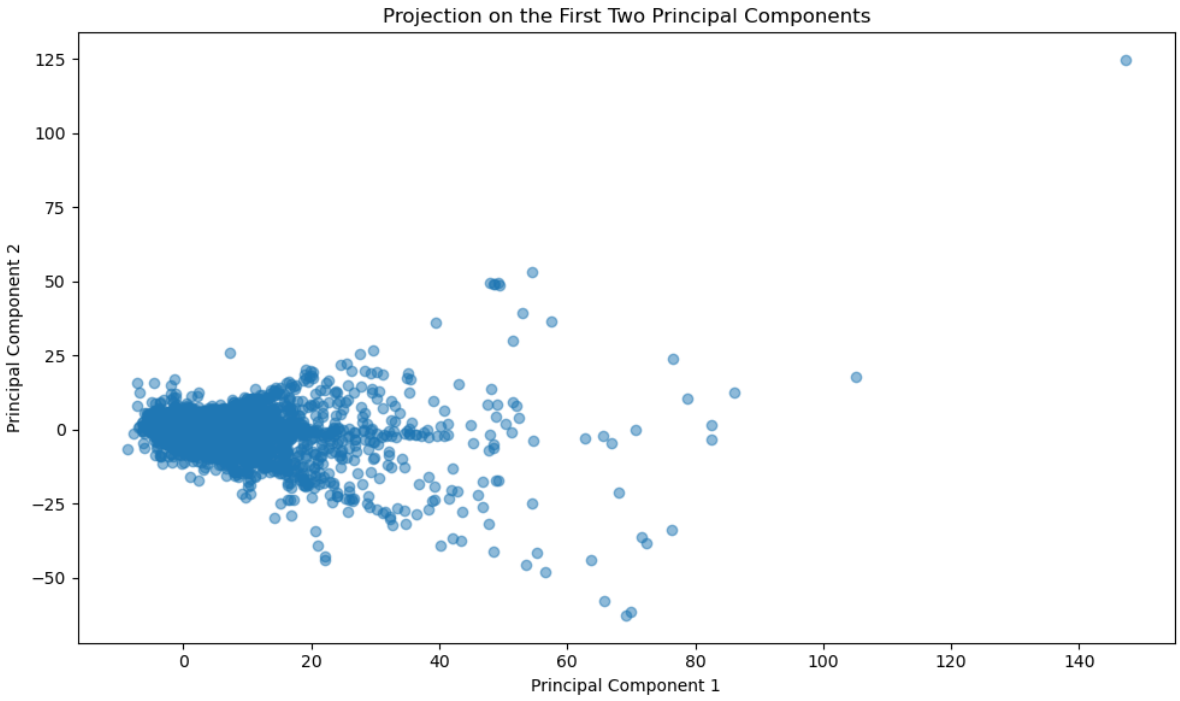

Figure 1.2: PCA scatter plot on the first 2 components

Chapter 2

Modeling Approaches and Hyperparameter Optimization

In this chapter, I will describe the modeling techniques I employed to forecast the target variable. We explored several predictive algorithms, including Ridge Regression, LSTM networks, and ensemble methods such as XGBoost and CatBoost. For each model, cross-validation (CV) was used alongside grid search to optimize hyperparameters and achieve robust performance. Below, we detail the approaches and the hyperparameter tuning strategies implemented. Please refer to each .py file for each part.

2.1 Ridge Regression

Ridge Regression is a linear model that incorporates an L_2 regularization term to mitigate overfitting. In our workflow:

- **Data Preprocessing:** Input features were standardized using the `StandardScaler` to ensure uniform scaling.
- **Hyperparameter Tuning:** We performed a grid search with k-fold cross-validation to tune the regularization parameter α, which controls the strength of the L_2 penalty.
- **Model Evaluation:** The optimal α was selected based on the lowest Mean Absolute Error observed during CV.

2.2 LSTM Networks

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) well-suited for sequential data. Our approach involved:

- **Sequence Creation:** Time series data were transformed into sequences using a sliding window technique, ensuring the model captured temporal dependencies.
- **Feature Engineering:** Both process features and derived time features (e.g., year, month, day, day of week, week of year) were included.

- **Model Architecture:** The network comprised an LSTM layer (with a tuned number of units), followed by a dropout layer for regularization, and a dense output layer.
- **Hyperparameter Tuning:** A grid search in conjunction with CV was performed to identify the best values for parameters such as the number of LSTM units, learning rate, batch size, and number of epochs. Early stopping was employed to prevent overfitting.

The final LSTM model was chosen based on its performance on the validation set, as measured by metrics like MSE.

2.3 XGBoost and CatBoost

Ensemble methods like XGBoost and CatBoost build strong predictive models by combining multiple decision trees. Our strategy for these models included:

- **XGBoost:** We fixed parameters such as `n_estimators` and `subsample`, and tuned hyperparameters including `max_depth`, `min_child_weight`, `colsample_bytree`, `gamma`, and `learning_rate` using grid search with 5-fold CV and also Bayesian search (but the results was less good). The evaluation metric used was Mean Absolute Error (MAE), and the best parameter set was selected based on CV performance.
- **CatBoost:** Similar tuning was performed for CatBoost. The model was optimized over parameters such as learning rate and iterations, also leveraging grid search and CV to minimize MAE.

For both methods, hyperparameter optimization ensured that the models generalized well to unseen data, with CV providing robust estimates of model performance.

Chapter 3

Summary and Results

In our experiments, the performance of the Ridge Regression and LSTM models did not reach the level of our benchmark, indicating that their predictive capabilities were limited in our context. In contrast, the ensemble methods, namely XGBoost and CatBoost, consistently achieved superior results. These models demonstrated robustness across various stages of data cleaning and preprocessing, thereby validating their effectiveness for our forecasting task.

The table below summarizes the performance scores for each model across four different types of data preprocessing: Raw data and three distinct cleaned versions. Where Outlier 1 corresponds to the clean data without the outliers detected by the K-Means on the PCA. Outlier 2 is where we delete all trains that have at least one outlier. And Winsorize is done by cutting 1% on each edge based on each quantitative feature.

Model	**Raw**	**Outlier 1**	**Outlier 2**	**Winsorize**
Ridge	0.98	0.85	1.14	1.10
LSTM	5.53	5.35	535.23	4.47
XGBoost	0.75	0.65	0.69	0.70
CatBoost	0.72	0.69	0.69	0.69

Table 3.1: Performance scores for each model across different data preprocessing stages.

These results underscore that while the Ridge Regression and LSTM models lag behind the benchmark with signs of overfitting, the advanced tree-based algorithms XGBoost and CatBoost emerge as the preferred choices for our predictive modeling efforts.