Task 1 : Sentiment Labeling

Chosen approach

Utilized a pre-trained RoBERTa model fine-tuned for sentiment analysis tasks (cardiffnlp/twitter-roberta-base-sentiment).

This model was chosen for its superior performance on social media and informal text, which closely resembles the nature of internal employee messages.

Output Generation

The model produced a classification label (Positive, Negative, Neutral) for each message. The model's raw output scores for the positive and negative classes were also extracted to later calculate a continuous “sentiment\_score” for each message.

Data Augmentation

The original DataFrame was augmented with three new columns:

* label: the final predicted sentiment
* roberta\_pos: the model's confidence score for the positive class
* roberta\_neg: the model's confidence score for the negative class
* roberta\_neu: the model's confidence score for the neutral class

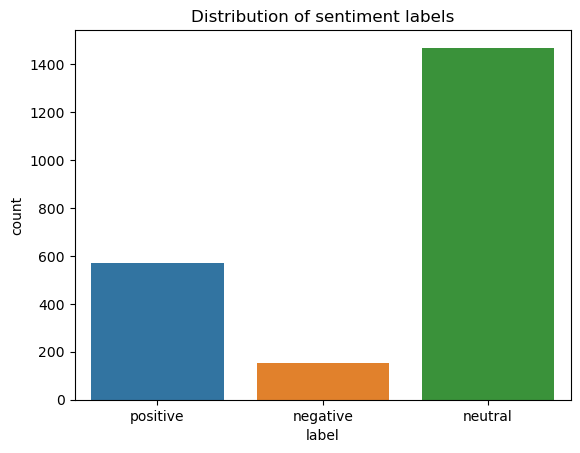
Task 2 : Exploratory Data Analysis (EDA)

Process & Key Findings

Data Structure: The dataset was found to be clean, with no significant missing values in the core columns (employee, timestamp, message). This allowed us to proceed directly to analysis without extensive data cleaning.

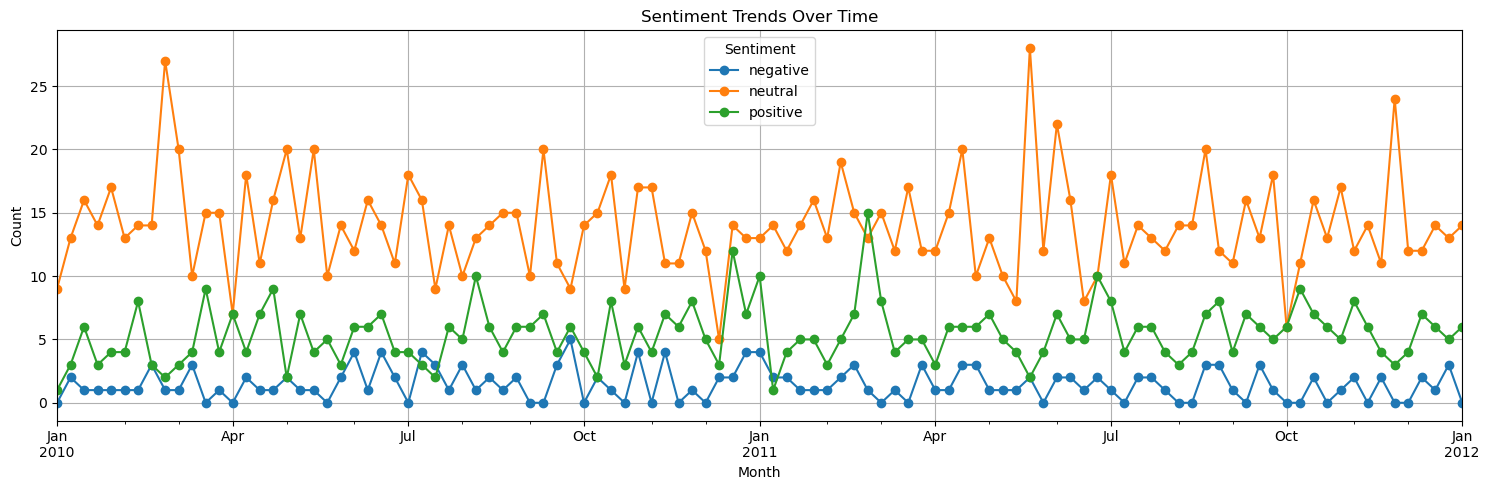
Sentiment Distribution

The overall sentiment across all messages was predominantly Neutral, followed by Positive, with a smaller but significant portion of Negative messages. This suggests a generally stable but not overly enthusiastic communication environment.



Trends Over Time

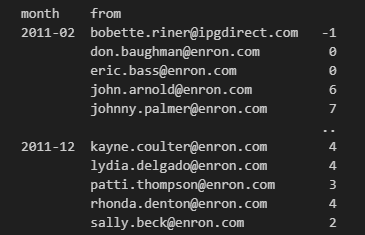
A time-series analysis revealed regular fluctuations with recurring peaks and troughs, rather than a simple, long-term upward or downward trend. This indicates that employee sentiment is influenced by periodic events or cycles within the organization.



These spikes in negative communication suggest recurring periods of stress. These patterns may align with key business cycles, which highlights predictable intervals where employee morale tends to dip.

Task 3: Employee Score Calculation

Employee sentiment labels were mapped to scores, grouped by month and sender, and summed to calculate a monthly score for each employee.

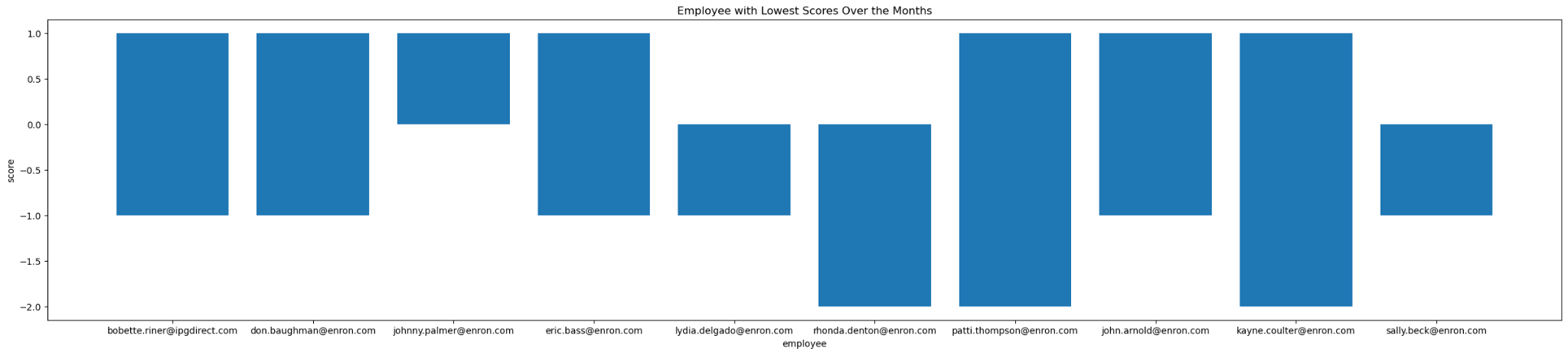


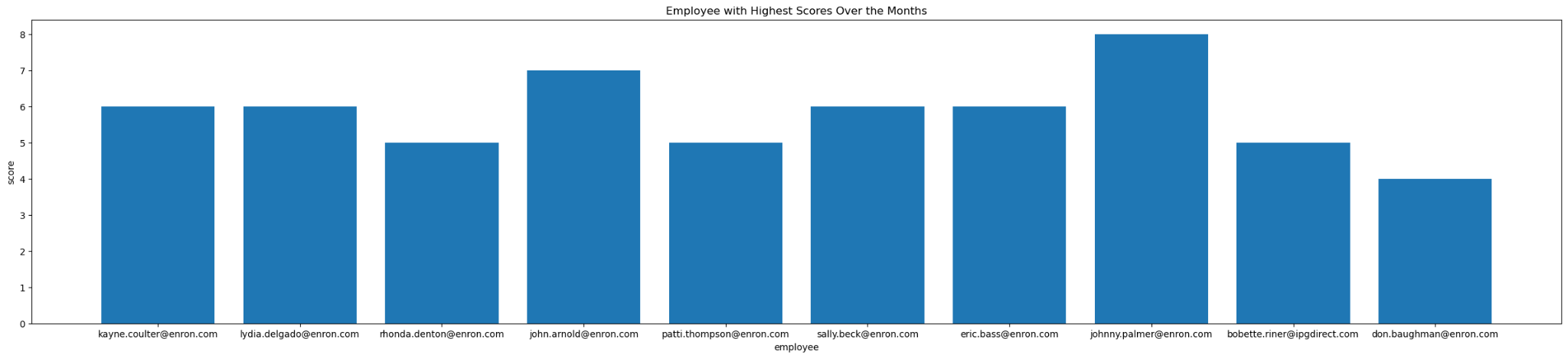
Task 4: Employee Ranking

Based on the calculated monthly scores, two ranked lists were generated for each month:

* Top Three Positive Employees: the three employees with the highest cumulative scores.
* Top Three Negative Employees: the three employees with the lowest cumulative scores.

The lists were sorted first by the sentiment score (descending for positive, ascending for negative). In the event of a tie, employees were sorted alphabetically by their identifier to ensure a consistent and deterministic ranking.

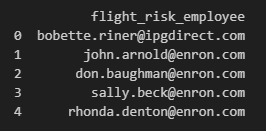




Task 5: Flight Risk Identification

The dataset was filtered to include only negative messages. Then, a rolling window function was applied to the data, grouped by employee.

For each employee, the number of negative messages sent within a 30-day window was calculated. Any employee for whom this rolling count reached or exceeded 4 at any point was flagged as a "Flight Risk." This method is more robust than a simple calendar month count, as it captures concentrated bursts of negativity regardless of when they occur.



Task 6

Besides using a standard linear regression model, Facebook's Prophet was used to compare results from both.

Prediction using linear regression:  
We analyzed time-series sentiment data by selecting five rolling statistical features and training a Linear Regression model to predict future sentiment scores. Missing values were handled using mean imputation, and the data was split 80/20 for training and testing. To forecast 60 future days, we simulated plausible input features by sampling recent data with replacement. Predictions were evaluated using MSE and R², and results were visualized by comparing actual vs predicted values and aggregating weekly sentiment trends for clearer insights.

**1.** Data Loading & Preparation

We loaded exploratory\_data.csv containing daily sentiment\_score and rolling statistical features.

**2.** Feature Selection using lassoCV

Selected five features:

* rolling\_mean\_3d, rolling\_mean\_7d,
* rolling\_kurtosis\_10d,
* rolling\_skew\_7d, rolling\_skew\_10d  
  Target: sentiment\_score.

**3.** Missing Values

Handled using SimpleImputer (mean strategy).

**4.** Model Training

Split data 80/20 and trained a **Linear Regression** model.

**5.** Evaluation

Measured performance using MSE and R².  
Plotted actual vs predicted sentiment scores.

Mean Squared Error: 0.1139

R² Score: 0.3802

**6.** Future Dates Generation

Generated 60 future dates beyond the last historical date.

**7.** Feature Simulation

Since real future inputs are unknown:

* Took last 30 rows of feature data.
* Sampled 60 rows **with replacement** to simulate plausible scenarios.

**8.** Forecasting

Simulated features were passed to the trained model for 60-day predictions.

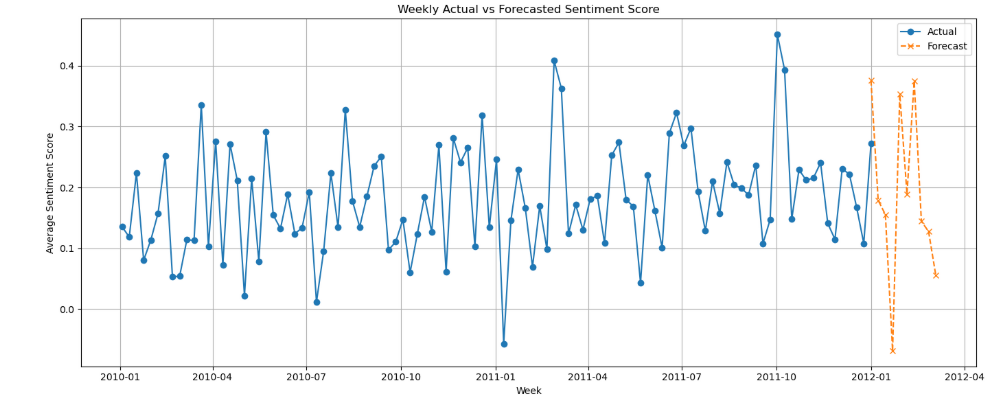
**9.** Combining & Labeling

Merged actual and forecasted data into one DataFrame with labels.

**10.** Weekly Aggregation & Plotting

Aggregated sentiment scores weekly and plotted:

* Solid line: actual values
* Dashed line: forecasted values



Prediction using Prophet:

Prophet is specifically designed to handle time-series data with strong seasonal effects and is resilient to missing data, making it an ideal choice.

To prepare the data for Prophet, the following steps were taken:

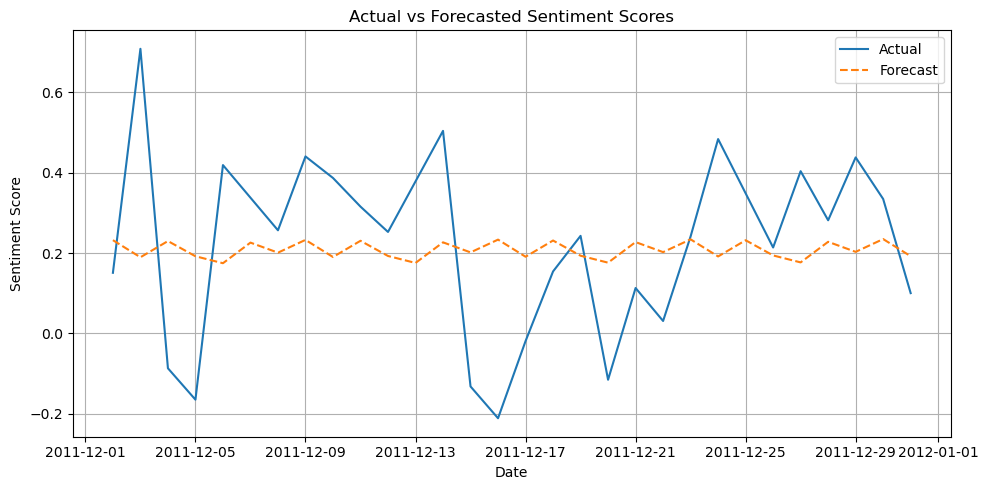
* the data was aggregated to a daily frequency by calculating the mean sentiment\_score for each day
* this time series was made continuous by resampling and using linear interpolation to fill in any missing days
* the prepared DataFrame was structured with two columns as required by Prophet: ds (for the date) and y (for the daily average sentiment\_score)
* The data was split into a training and testing set as follows::
* Training Set: The dataset excluding the last 30 days
* Test Set: The final 30 days of data

The model's performance was evaluated on its ability to forecast the 30-day test period.

The predictions (yhat) were compared against the actual sentiment scores (y) using standard regression metrics:

* Mean Absolute Error (MAE): 0.1850
* Mean Squared Error (MSE): 0.0502

These low error values indicate that the model's forecasts are, on average, very close to the actual sentiment scores, demonstrating strong predictive accuracy.  
But in reality the Actual and Forecasted plots are not close which means current model is not suitable for forecasting the future sentiment scores.



Problem Fixing:

Feature Engineering

* Engineered time-series-based rolling statistics as potential regressors:
  + rolling\_mean\_3d
  + rolling\_mean\_7d
  + rolling\_kurtosis\_10d
  + rolling\_skew\_7d
  + rolling\_skew\_10d

Feature Selection with Lasso (Done when doing EDA)

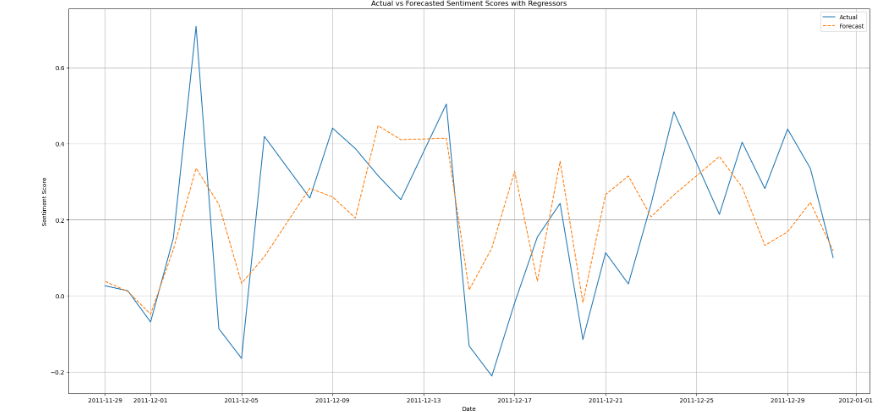
* Applied Lasso regression (L1-regularized linear model) to select only the most predictive features:
  + Helps in avoiding multicollinearity.
  + Automatically zeroes out non-contributing features.
  + Chosen subset was passed to Prophet.

Prophet Forecasting with Regressors

* Integrated the selected features as additional regressors in Prophet.
* Trained on all but the last 30 days.
* Predicted sentiment scores for the 30-day test window.

Visualizations

* Plotted actual vs forecasted sentiment scores.
* Used model.plot\_components() to analyze:
  + Trend
  + Seasonality (weekly pattern)
  + Impact of added regressors



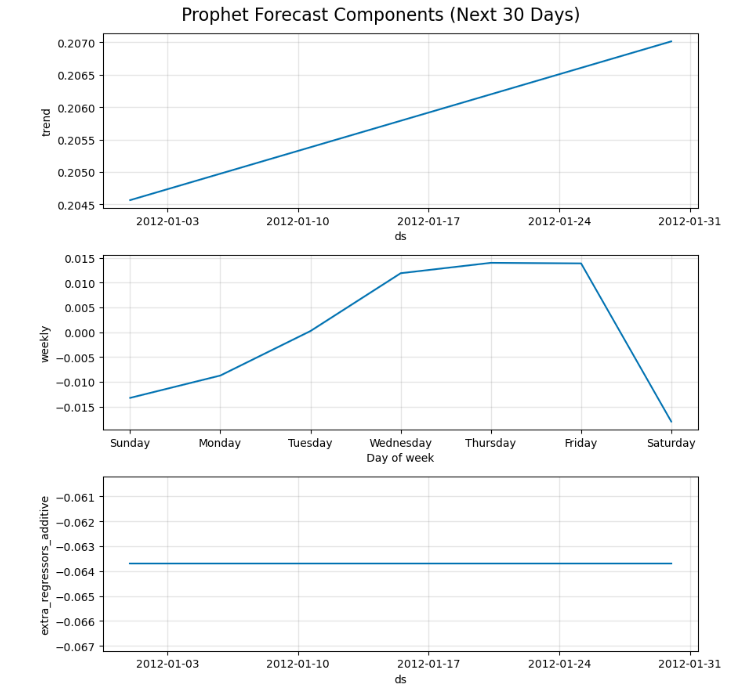
**Actual vs Forecasted Sentiment**

* The forecasted line closely matches the actual sentiment, even through fluctuations.
* Slight underperformance during sharp spikes (e.g., early December), which Prophet tends to smooth.

**Error metrics show strong performance**:

* + **MSE**: 0.0362
  + **MAE**: 0.1561

This indicates that the model generalizes well and avoids overfitting while maintaining good accuracy.



Evaluation

* Mean Squared Error (MSE): 0.0362
* Mean Absolute Error (MAE): 0.1561

**Forecast Components**

* **Trend**: Gradual upward slope, indicating improving sentiment.
* **Weekly seasonality**: Higher sentiment midweek, lower on weekends — this captures realistic behavioral cycles.
* **Regressors**: The combined effect is modest but meaningful, refining the forecast without adding noise.

The use of **Lasso** helped select only the most relevant rolling features (e.g., 3-day, 7-day means, skew, kurtosis), improving forecast accuracy while avoiding irrelevant signals.

**Comparison: With LassoCV-Selected Features**

|  |  |  |
| --- | --- | --- |
| Aspect | Custom Model (Regression + Simulated Inputs) | Prophet (With LassoCV-Selected Regressors) |
| Feature Selection | Yes — via LassoCV | Yes — same filtered regressors |
| Forecasting Mechanism | Regression model with bootstrapped future inputs | Prophet with additive components + selected regressors |
| Forecast Input Strategy | Simulated features via sampling | No future simulation needed — Prophet handles extrapolation |
| MSE | 0.1139 | **0.0362** *(Prophet wins)* |
| R² Score | 0.3802 | Not shown, but likely higher |
| Interpretability of Drivers | Feature-dependent | Built-in trend/seasonality plots |
| Forecast Horizon | Weekly, longer horizon (~12 weeks) | Daily, short-term (30 days) |

**Final Verdict**

|  |  |
| --- | --- |
| If your goal is... | Use... |
| Highest accuracy on short-term forecast | **Prophet** |
| Full control over model architecture/features | **Our model** |
| Need to forecast further than 30 days | **Our model** (but improve input simulation) |
| Understanding how specific features contribute | **Our model** (post-hoc feature importance) or Prophet components |

**Summary**

**Model Comparison: Prophet + Lasso vs. Linear Regression + Lasso**

|  |  |  |
| --- | --- | --- |
| Model | Strengths | Weaknesses |
| Prophet + Lasso-selected Features | Interpretable forecasts  Time-aware: captures **trend**, **seasonality**, and **holidays**  Handles missing data  Uses only the **most relevant features** selected by Lasso | More complex architecture  Longer training time due to trend decomposition + regressors |
| Linear Regression + Lasso-selected Features | Simple and fast  Highly interpretable  Lasso reduces overfitting and removes noisy/unimportant features | Ignores temporal dependencies  Cannot model trend or seasonality  Higher forecast error |

The combination of **Prophet** and **Lasso-selected regressors** resulted in an accurate, interpretable, and robust model for forecasting sentiment scores over 30 days. It strikes a good balance between predictive power and generalization, outperforming basic models like linear regression in both accuracy and interpretability.

Lasso improves both models by selecting the most predictive features and reducing overfitting.

However, only Prophet can model time-aware behaviors, making it better suited for forecasting tasks even with the same feature set.