PREDICTING THE PRICES OF TMT BARS USING AI MODELS

Ankita Samanta – M.Sc. in Data Science, Batch: 2024-2026, St. Xavier's College (Autonomous)Kolkata

Kathakali Sardar – M.Sc. in Data Science, Batch: 2024-2026, St. Xavier's College (Autonomous) Kolkata

Dipanjan Chakroborty – M.Sc. in Data Science, Batch: 2024-2026, St.Xavier's College (Autonomous) Kolkata

Shuvam Maity – M.Sc. in Data Science, Batch: 2024-2026, St. Xavier's College (Autonomous)Kolkata

Raunak Nandy – M.Sc. in Data Science, Batch: 2024-2026, St. Xavier's

College (Autonomous)Kolkata

Arkoprovo Mondal – M.Sc. in Data Science, Batch: 2024-2026, St. Xavier's College (Autonomous)Kolkata

Project Guide/Mentor Name:

Siddhartha Roychowdhury

Period of Internship: 19th May 2025 – 15th July 2025

Report submitted to: IDEAS – Institute of Data Engineering, Analytics and Science Foundation, ISI Kolkata

Abstract:

This project aims to predict steel prices of the IF and BF routes using AI models. We developed Random Forest, LSTM, and Prophet models using historical data to predict prices over the past 6 months and compared model accuracy. A correlation analysis was conducted between actual and predicted values to evaluate short-term performance. Using the trained models, recursive forecasting was applied to predict prices for the next 7, 30, and 60 days. We also performed monthly comparative analysis between IF and BF price trends. Additional variables provided during the internship were cleaned, merged, and imputed to form a common dataset. Highly correlated variables were selected for both IF and BF datasets. Models were retrained on these selected features to further enhance prediction accuracy. A final model comparison was conducted, and the best-performing model was used for the final future predictions. The project demonstrates a complete data-driven approach to time series prediction using both classical and deep learning techniques.

Introduction:

The steel industry experiences price fluctuations due to various economic and industrial factors. Accurate prediction of steel prices can aid in inventory planning and pricing strategies. This project focuses on predicting steel prices in two major routes: the **Induction Furnace (IF)** and **Blast Furnace (BF).** Technologies like Random Forest Regression, Prophet and LSTM (Long Short-Term Memory networks) were employed to build predictive models. The dataset included historical steel prices along with 9 newly introduced variables. We applied extensive preprocessing steps such as missing value imputation, lag feature creation, and correlation-based feature selection. The purpose was to compare the price behavior of IF and BF routes, select highly influential variables, and use them to built robust prediction model.

Training Topics Covered (First 2 Weeks):

- Python and Data Cleaning
- Data Visualization
- Feature Engineering
- Machine Learning Basics

The ultimate goal was to predict future prices for 7, 30, 60 days and determine which model performs best for real-time prediction.

Project Objective:

- To predict steel prices of the BF and IF routes using Random Forest, Prophet and LSTM models based on historical data.
- To evaluate and compare the prediction performance among models on past 6 months data using error metrics and correlation analysis.
- To generate future prediction (7, 30, 60 days) using recursive prediction from trained models.
- To analyze monthly price differences between IF and BF routes and observe underlying market trends.
- To integrate additional variables, perform data cleaning and imputation, and select highly correlated variables with steel prices.
- To determine the best-performing model based on prediction accuracy and use it for final future price forecasting and visualization.

Here's an explanation of each AI model used:

Random Forest

- Use: Classification and regression tasks, such as predicting customer churn or identifying fraudulent transactions.
- Example: Predicting whether a customer will churn based on features like usage patterns, demographic data, and billing information.

LSTM (Long Short-Term Memory)

- Use: Time-series forecasting, natural language processing, and speech recognition.
- Example: Predicting stock prices based on historical data, or generating text summaries of news articles.

. Prophet

- Use: Time-series forecasting, such as predicting stock prices, sales, or weather patterns.
- Example: Predicting daily sales based on historical data.

Model Used Justification Based on the Literature Review:

1.TMT Bar Price Depends on Multiple Complex Factors:

The literature discusses how TMT bars vary by grades (Fe 415, Fe 500, Fe 600) and are used in diverse structural applications. These uses impact market demand, which in turn affects price fluctuation over time. Thus, predicting TMT prices requires models that can handle non-linear relationships and time-based dependencies.

2. Why Random Forest?

TMT bar pricing may be influenced by multiple features (e.g., grade, usage, demand, raw material cost). Random Forest is effective for non-linear regression and handles high-dimensional structured data well. It also provides feature importance, helping understand what factors more affect TMT prices.

3. Why Prophet?

TMT bar price trends follow seasonality and long-term trends (e.g., during monsoon or high construction seasons). Prophet is excellent for capturing seasonal trends, holidays, and trend changepoints, making it a good fit for forecasting monthly/weekly prices. It's also interpretable, which helps in practical construction forecasting.

4. Why LSTM?

The process of TMT bar manufacturing and demand is dynamic and temporal**. LSTM (Long Short-Term Memory)** is powerful for learning patterns in sequential data, such as price time series. It captures long term dependencies and is useful when previous months' trend impact future prices.

Methodology:

Here, our first object is to predict for last 6 months of TMT 12 steel prices for both the BF and IF routes using Random Forest Regressor model, LSTM model and Prophet Model. The methodology includes data preprocessing, feature engineering, model training, prediction performance evaluation and correlation analysis between two production routes.

A. **Random Forest Regressor:**
   I. **Data Preprocessing and Splitting:** The dataset consists of daily TMT steel prices from both the BF and IF routes. For each dataset:
      - The last 6 months of available data were reserved for testing.
      - The remaining data was used for training and validation.

Additionally, for each date, a DayIndex feature was created to represent the number of days since the start of the dataset. This numerical representation allows the model to capture temporal trends. To capture short-term dependencies, a lag feature (Lag1) was engineered by shifting the price value by one day. Rows with missing lag values were dropped.

   II. **Model Selection: Random Forest Regressor**
   The Random Forest Regressor was chosen due to its ability to handle:
      - Non-linear relationships,
      - Robustness to outliers and overfitting,
      - And effectiveness on datasets with limited feature sets.

Separate Random Forest models were trained for the BF and IF routes using the following features:

- DayIndex: Represents the time progression.
- Lag1: Previous day's price to incorporate short-term memory.

The model was configured with:

- n_estimators = 100 (number of decision trees),
- random_states=42 (for reproducibility).

III. **Model training , testing and evaluation:**

The model was first trained on the training portion of the dataset. Then the model was used to predict TMT prices on the test set (last 6 months) and performance was evaluated using three standard metrices:

- **Mean Absolute Error (MAE):** Measures average absolute difference between predicted and actual prices.
- **Root Mean Square Error (RMSE):** Penalizes larger errors more heavily.
- **R^2 Score:** Indicates the proportion of variance explained by the model.

These performance metrices were computed to assess real-world prediction ability and then line plots were generated for both BF and IF routes.

IV. **Cross-Route Correlation Analysis:**

To explore the relationship between IF and BF predicted prices:

- Predictions from both routes were merged on the Date column.
- Pearson correlation was computed between predicted BF and predicted IF

This analysis helped assess whether predicted trends in one route correlate with the other, which could indicate shared market drivers or similar demand patterns.

B. LSTM

1. **Data Collection and Preprocessing:**
   - **Dataset**: Daily TMT 12MM IF Route steel prices with dates were used.
   - **Date Parsing**: The Date column was converted to datetime format and sorted chronologically.
   - **Feature Engineering**:
     - A binary feature is_monsoon was added to indicate monsoon season (June to September).
     - To model **seasonality**, cyclical monthly patterns were encoded using sine and cosine transformations:

$$\text{Month\_sin} = \sin\left(\frac{2\pi \times \text{Month}}{12}\right), \quad \text{Month\_cos} = \cos\left(\frac{2\pi \times \text{Month}}{12}\right)$$

   - **Normalization:** The Price column was scaled to the [0,1] range using MinMaxScaler to ensure stable LSTM training.
2. **Sequence Creation:**
   - LSTM models require sequential data. Hence, time series windows of **30 consecutive days** (seq_length = 30) were created.
   - For each sequence, the input consisted of:

        - Scaled_price
        - Month_sin
        - Month_cos
    - The target was the price on the next day following the 30 day sequence.
3. **Train-Test split:**
    - The dataset was split such that the last 6 months of data were reserved for testing.
    - The LSTM was trained only on earlier historical data to simulate real-world prediction conditions.
4. **LSTM Model architecture:**
    - A univariate LSTM model was built with the following configuration:
        - 1 LSTM layer with 64 units
        - 1 Dense output layer with linear activation for regression
    - **Loss Function**: Mean Squared Error (MSE)
    - **Optimizer**: Adam
    - **Training:**
        - Epochs: 300
        - Batch size: 32
5. **Model Evaluation:**
    - The model's predictions on the last 6 months were compared to actual prices.
    - **Evaluation metrics:**
        - **RMSE**
        - **MAE**

          R² Score
    - A visual comparison plot of **actual vs predicted prices** was also generated.

C. **Prophet:**

a) **Data Preprocessing:**
- The dataset was first chronologically sorted and cleaned.
- Lag-based features were generated:
    - **Lag1**: BF price 1 day before.
    - **Lag2**: BF price 2 days before.
- **Rolling Mean**: 7-day moving average (Rolling7) was computed to incorporate local trends.
- **Calendar Features**:
    - Month and Weekday were derived from the date to model monthly and weekly seasonality.
    - A binary feature is_monsoon was created to denote whether a day falls in the Indian monsoon season (June to September).

b) **Prophet Model Configuration:**
- Prophet was configured with:
    - **Yearly seasonality** enabled (to model long-term patterns),

    - **Multiplicative seasonality** mode, suitable when seasonal effects scale with price level.
    - **Custom monsoon seasonality** was added with a Fourier order of 10.
  - **External Regressors**:
    Prophet was extended with the following additional regressors:
    - Lag1, Lag2
    - Rolling7 (trend smoothing)
    - Month, Weekday

c) **Forecasting Last 6 Months:**
  - Only the last 6 months of data were selected (prophet_df['ds'].max() - 6 months) for prediction.
  - A **future DataFrame** was created using actual lag and seasonal features available from historical records.
  - The model was used to reconstruct (i.e., predict) the actual prices for the same period to validate its accuracy.

d) **Evaluation:**
  The following metrics were used to assess the model performance:
  - **R^2 Score**: Measures the proportion of variance explained by the model.
  - **RMSE (Root Mean Squared Error)**: Indicates average prediction error magnitude.

The actual and predicted prices for the 6-month period were plotted to visually assess alignment.

RESULT:

<u>FOR IF ROUTE:</u>

Prophet model R^2:0.9702

LSTM model R^2:0.9524

Random Forest model R^2: 0.9165

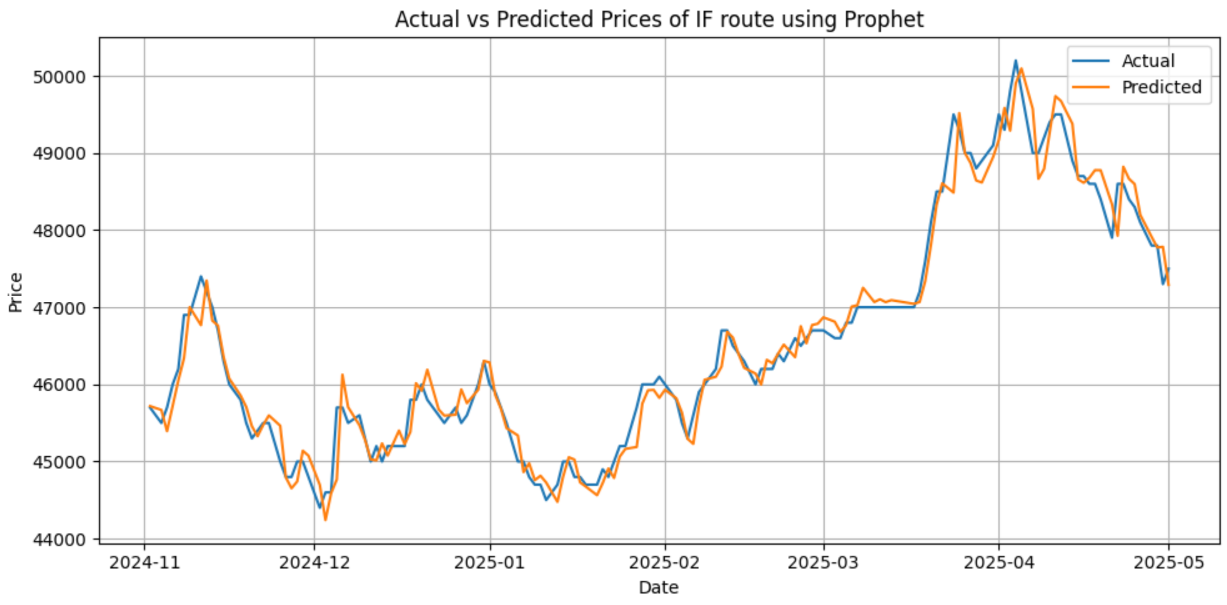
<u>FOR BF ROUTE:</u>

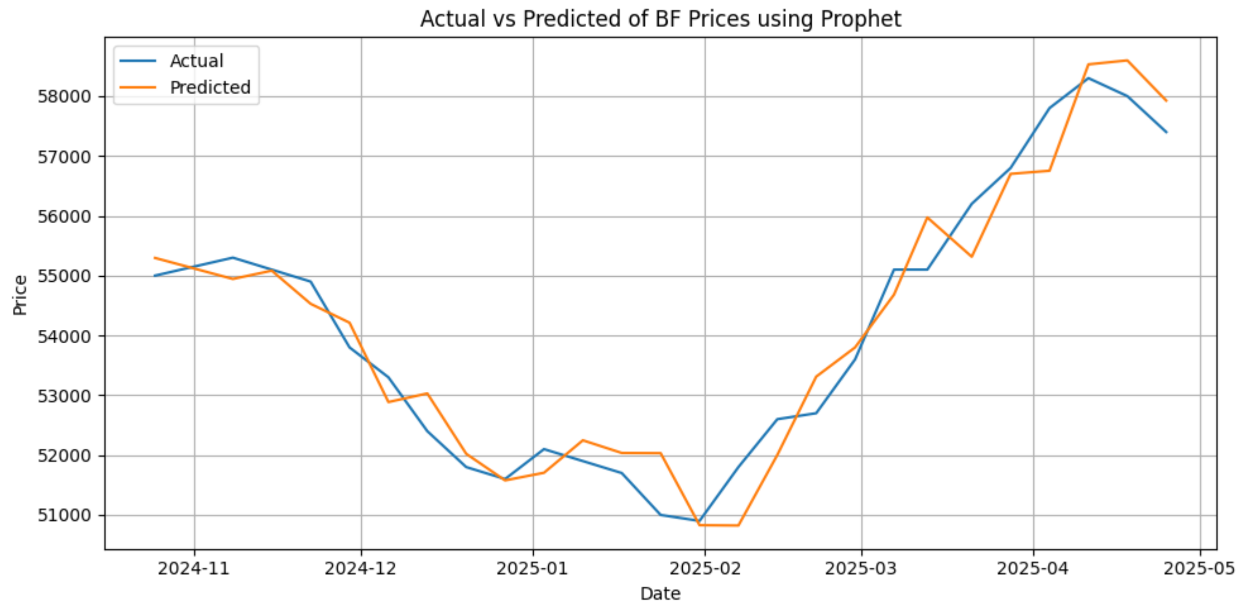
Prophet model R^2:0.9418

LSTM model R^2: -0.0121

Random Forest model R^2: 0.8947

So, **Prophet** performed best for both IF and BF routes. It gave the highest R^2 scores, especially on the smaller BF dataset where LSTM failed. This shows Prophet handles seasonality and small datasets better than LSTM and Random Forest.





Correlation between IF and BF route predicted prices is 0.8355.

Future predicted dataset using Prophet: last 6 months' predicted dataset using Prophet

Now, our goal is to predict for last 7 days, 30 days, 60 days using those 6 months' prediction.

METHODOLOGY

A. **Random Forest Regressor:**

- **Data Preparation and Feature Engineering:**
    - The historical price dataset was processed to remove irrelevant columns and ensure consistent formatting.
    - Dates were converted to datetime format and sorted chronologically.
    - New features were created to capture temporal dependencies:
        - **Lag1**: Previous day's price.
        - **Day Index**: Number of days from the starting date of training data.

    These features served as predictors for the Random Forest model.

- **Model Training:**
    - A Random Forest Regressor was trained separately on BF and IF price data.
    - The model learned the relationship between:
        - The **Day Index**, representing the time elapsed from the start of the training data.
        - The **Lag1 price**, representing the most recent known or predicted value
    - The target variable was the actual steel price on the current day.
- **Future Price Prediction:**
    - The next 7 days of prices were forecasted recursively:
        - Starting from the last known prediction in the test set,
        - The predicted price of one day was used as the input for the next day.
        - For each future day:
            - The corresponding **Day Index** was calculated.
            - A feature vector [Day Index, Lag1] was formed.
            - This was fed into the trained Random Forest model to obtain the predicted price.
    - This process was repeated for each day in the 7-day horizon.
- **Output and Visualization:**
    - The predicted prices were stored in separate DataFrames for BF and IF routes.
    - These predictions were printed for inspection and later used for comparative analysis and visualization with other models.

B. **LSTM:**

After training the LSTM model on historical steel price data enriched with seasonal features (such as monthly sin/cos encoding and monsoon indicator), we generate prediction for the next 100 days using an auto-regressive approach. This step-by-step strategy is outlined below:

- **Starting Point: Last Known Sequence:**

- We begin prediction using the most recent 30-day sequence from the historical data.
- Each time step in the sequence contains:
  - Scaled Price
  - Month_sin(to capture cyclical trend)
  - Month_cos(to encode seasonal cycle)

- **Iterative Prediction (Day-by-Day- Loop):**
  - For each of the next 100 days, we perform the following steps:
    - **Data Generation:** The next date is calculated by incrementing the last known date.
    - **Seasonality Feature Creation:**
      1. Month_sin = sin(2 π * month / 12)
      2. Month_cos= cos(2 π* month/12)
    - **Next Input Sequence Formation:**
      1. The new day's input replaces the oldest in the rolling 30-day window.
      2. The most recent predicted price is used (auto-regressive input).

    - **Prediction:**
      1. The updated sequence is fed into the trained LSTM model.
      2. The model outputs the next day's scaled price, which is then:
         - Stored for future use.
         - Inverse-transformed using the original scaler to obtain the actual predicted price.
- **Building Forecast Output:**
  - The 100 predicted prices are stored alongside their corresponding forecast dates.
  - A new Dataframe is created containing:
    1. Date
    2. Predicted_Price (actual scale)
    3. Month
    4. Is_Rainy_Season
- **Seasonal Awareness:**
  - The future prediction highlights seasonal dips during the monsoon (June to September):
    1. Inclusion of monthly cyclical features in the input.
    2. Model's learning of past seasonal price behaviour.
- **Outcome:**
  - This iterative, self-feeding loop enables the LSTM to generate robust forward-looking predictions.

- The method simulates how the model would perform in a real-world predicting scenario where only past data is known and future inputs must be inferred.

C. **Prophet:**

- **Initialization:**
    1. The prediction starts from the **last known observation** in the training dataset.
    2. A 60-day future date range was created starting from the day after the last date in the training set.
- **Recursive Prediction Loop**:
  For each date in the 60-day future range:
    1. We computed the lag-based features using the most recent values from historical data or previously predicted values:
        - Lag1, Lag2, Lag3, Lag4 (last 4 days)
        - Rolling7: 7-day moving average of recent prices.
    2. Time-based features such as Month, Weekday, and is_monsoon (June to September) were generated directly from the date.
    3. Missing values (which can occur in early steps) were imputed with 0 for stability.
    4. The Prophet model was used to generate a forecast for the current date using the constructed feature set.
    5. The predicted price (yhat) was stored and then **added back to the historical series**, so it could be used to compute future lags and rolling averages in subsequent iterations.

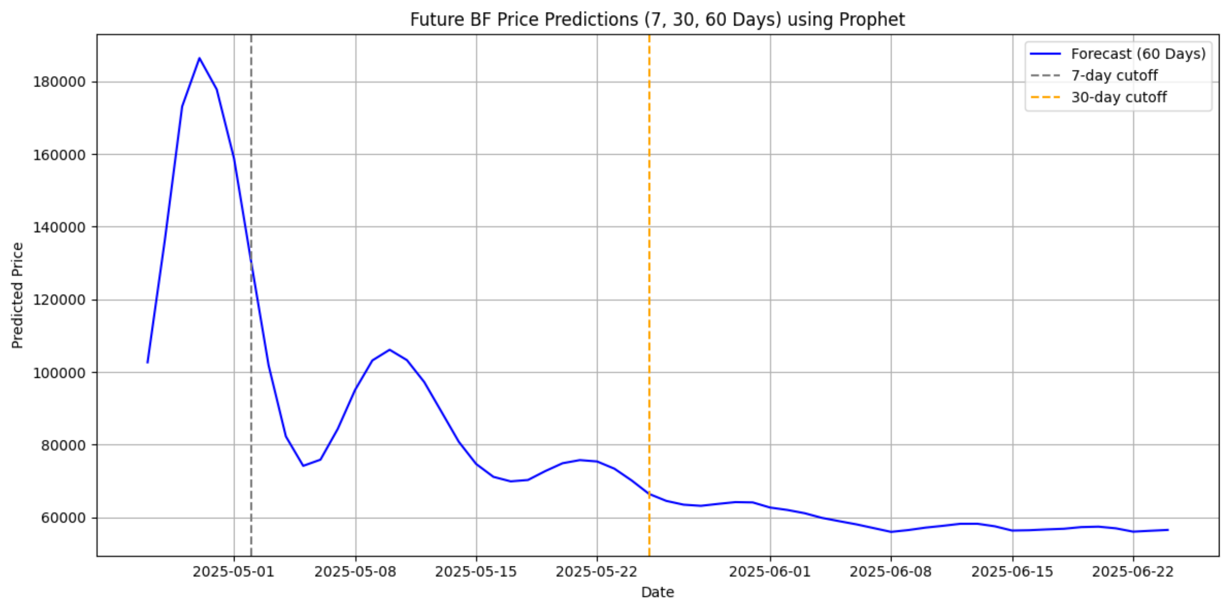
▪ **Output:**

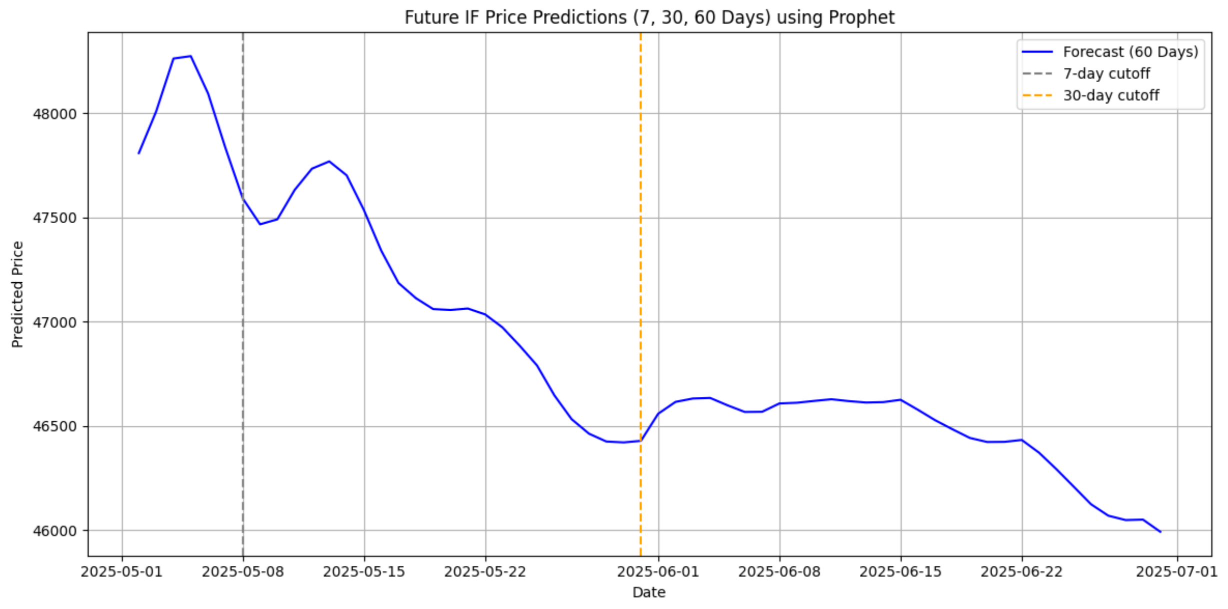
- All predictions were stored in a DataFrame and saved to a CSV file.
- A line plot was generated to visualize predicted prices for the next:
    - ➢ **7 days** (short-term),
    - ➢ **30 days** (mid-term), and
    - ➢ **60 days** (long-term).

▪ Vertical lines were drawn at the 7-day and 30-day marks to help distinguish zones of increasing forecast uncertainty.

RESULT:

Since Prophet performed the best, we are giving the next prediction of Prophet.





Predicted Dataset for IF and BF route: future prediction using Prophet

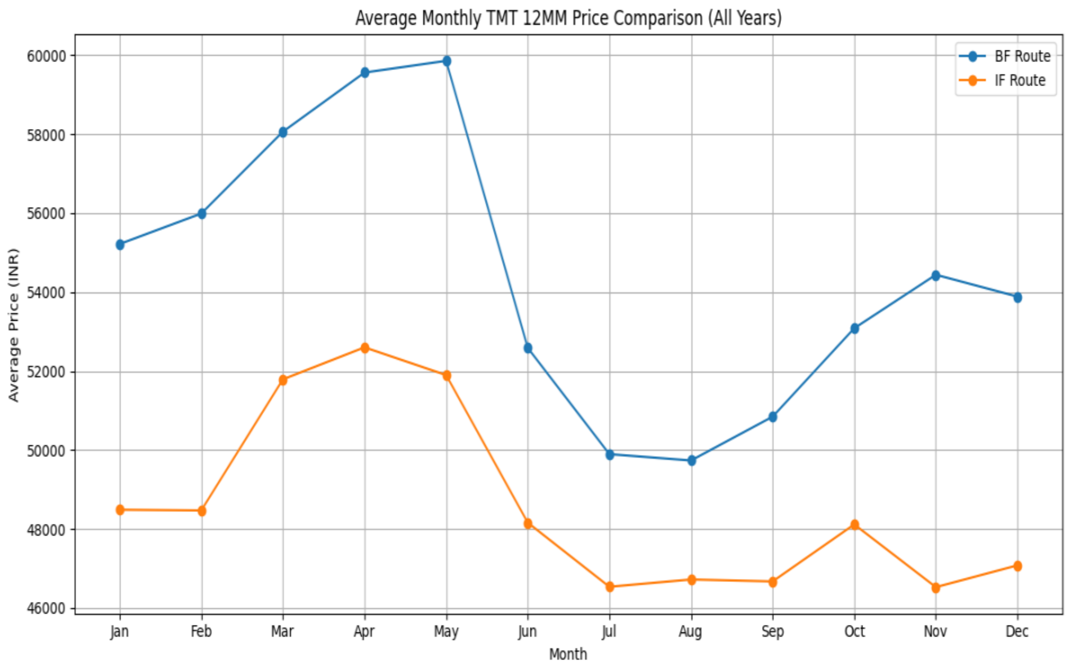
Now, we have performed monthly comparison between IF and BF route.

METHODOLOGY:

- **Data Preparation**
- Combined datasets of BF and IF routes into a single DataFrame.
- Added a new column Route to distinguish between BF and IF.
- **Feature Engineering**
- Extracted Year and Month from the Date column to enable time-based grouping.
- **Monthly Average Calculation**
- Calculated the average price for each route, month, and year.

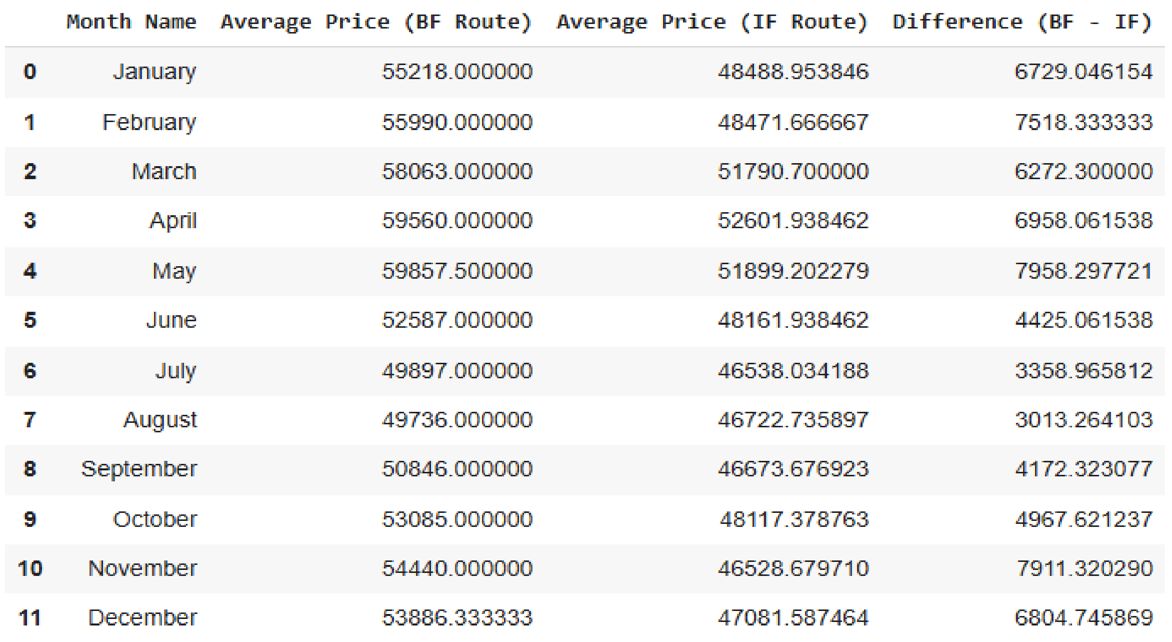
- Then computed the overall monthly average price (e.g., average of all January, February etc.) across all available years.
- **Pivot for Visualization**

- Converted the result into a pivot table where each row represents a month and columns show average prices for BF and IF routes.



Results:

- **Price Difference:** The **BF Route consistently shows higher average monthly prices** than the IF Route throughout the year, indicating a premium likely due to production or quality differences.
- **Seasonal Trend**: Both routes experience a **price peak between March and May**, followed by a **notable drop from June to August**.
- **Volatility**: The **BF Route shows greater price fluctuations** compared to the IF Route, which remains relatively stable except for a mild rise in March–May.
- **Lowest Prices**: Both routes hit their **lowest average prices around July–August**, possibly due to monsoon season or market slowdown.
- **Implication**: Buyers might prefer the **IF Route for cost-effectiveness**, whereas the **BF Route might be chosen for quality or specification needs**, despite its higher cost.



	Month Name	Average Price (BF Route)	Average Price (IF Route)	Difference (BF - IF)
0	January	55218.000000	48488.953846	6729.046154
1	February	55990.000000	48471.666667	7518.333333
2	March	58063.000000	51790.700000	6272.300000
3	April	59560.000000	52601.938462	6958.061538
4	May	59857.500000	51899.202279	7958.297721
5	June	52587.000000	48161.938462	4425.061538
6	July	49897.000000	46538.034188	3358.965812
7	August	49736.000000	46722.735897	3013.264103
8	September	50846.000000	46673.676923	4172.323077
9	October	53085.000000	48117.378763	4967.621237
10	November	54440.000000	46528.679710	7911.320290
11	December	53886.333333	47081.587464	6804.745869

Now, we have taken few variables to check whether the variables are useful for improving prediction or not and for this first we have selected highly correlated variables and check model performance among these three models.

METHODOLOGY:

- Merged 9 newly provided variables into IF and BF datasets.
- Handled missing values using forward and backward fill.
- Calculated correlation of each variable with steel prices.
- Selected top correlated features for model input.

RESULTS:

- High Correlated variables for IF ROUTE:
    - i. Scrap Mandi (Correlation:0.965501)
    - ii. Sponge Iron PDRI (Correlation:0.932645)
    - iii. Sponge Iron DRCLO (Correlation: 0.950870)
- High Correlated variables for BF ROUTE:
    - i. Scrap Mandi (Correlation: 0.885420)
    - ii. Sponge Iron PDRI (Correlation: 0.882816)
    - iii. Sponge Iron DRCLO (Correlation: 0.861774)

Now, with these high correlated variables we want to check whether they are helping to improve prediction or not with these three models.

METHODOLOGY:

A. Random Forest Regressor:

- **Data Preparation:** We began by importing and cleaning the dataset. The Date column was converted to datetime format, and the data was sorted chronologically to preserve the time series nature. Missing values caused by feature engineering were handled appropriately by dropping rows with insufficient lag/rolling data.
- **Feature Engineering**: To enhance the model's predictive power, we generated several time-dependent features:
  - **Lag Features**: Lag1 and Lag2 representing the price 1 and 2 days prior.
  - **Rolling Mean**: 7-day rolling average (Rolling7) to capture short-term trends.
  - **Date Features**: Extracted Month and Weekday from the Date to model seasonal and weekly effects.
  - **External Factors**: Included prices of related materials such as Scrap_Mandi_Price, Sponge_Iron_DRCLO_Price, and Sponge_Iron_PDRI_Price.
- **Model Selection with Cross-Validation:**
  To ensure temporal consistency and avoid data leakage, we used TimeSeriesSplit (5 folds) for cross-validation. We performed a grid search over the following hyperparameters:
    - n_estimators: [100, 200]
    - max_depth: [5, 10, None]

The model was evaluated on each fold using **negative Mean Absolute Error (MAE)** as the scoring metric. The best model was selected based on the grid search results.

- **Train-Test Split:**
  The dataset was split chronologically into training (first 70%) and testing (last 30%) sets to preserve the time series order. This mimics a real-world forecasting scenario where future values are predicted based on past data.
- **Model Training and Forecasting:**
  The best Random Forest model (from GridSearchCV) was retrained on the training set and used to generate predictions on the test set.
- **Model Evaluation:**
  The model's performance was evaluated using RMSE, MAE, R^2 Score.

These metrics gave a quantitative understanding of the model's accuracy on unseen data.

B. LSTM:

- **Data Preprocessing:**
  - We began by loading and sorting the dataset chronologically based on the Date column.
  - The target column was renamed as Price for clarity.
  - A binary indicator variable is_monsoon was added to account for potential effects of monsoon months (June to September).

- To encode **monthly seasonality**, we added two features:
  - Month_sin = sin(2π * month / 12)
  - Month_cos = cos(2π * month / 12)

  This allows the model to capture cyclical behavior across the year.
- The price values were normalized using **MinMaxScaler** to speed up training and stabilize gradients.

▪ **Sequence Generation for LSTM:**
To train an LSTM, we transformed the data into supervised learning sequences:
- A sliding window approach was used to generate sequences of **30 consecutive days** (seq_length = 30), each followed by the next day's price as the target.
- Each sequence contains the past 30 days' Price_scaled, Month_sin, and Month_cos.

The resulting dataset was split into training (70%) and testing (30%) subsets in chronological order to simulate a realistic forecasting setup.

▪ **Model Architecture:**
We implemented an LSTM model using TensorFlow/Keras:
- **Input Layer**: Accepts sequences of shape (30 timesteps, 3 features).
- **LSTM Layer**: 64 hidden units, without return sequences.
- **Dense Layer**: Outputs the predicted scaled price for the next day.
- **Loss Function**: Mean Squared Error (MSE)
- **Optimizer**: Adam

The model was trained for 300 epochs with a batch size of 32 and 10% validation split.

▪ **Evaluation:**
- After training, the model was used to generate predictions on the test set.
- The predicted values were inverse-transformed back to the original price scale using the scaler.
- Performance was evaluated using standard regression metrics such as RMSE, MAE, R^2 Score.

These metrics helped assess the model's predictive accuracy on unseen data.

C. Prophet:

▪ **Data Preprocessing:**
- The dataset was first cleaned by dropping irrelevant or redundant columns, followed by formatting the Date column to datetime and sorting the data chronologically.
- Missing values from lag or rolling calculations were dropped.

▪ **Feature Engineering:**
Several time-based and lag-based features were created to improve the model:
- **Lag1 & Lag2**: Previous 1-day and 2-day prices.
- **Rolling7**: 7-day moving average to capture short-term trends.

    - **Month** and **Weekday**: To reflect monthly and weekday trends.
    - **is_monsoon**: A boolean indicator identifying monsoon months (June to September), potentially affecting steel production or demand.
- **Prophet Model Configuration:**
    - Prophet was configured with multiplicative seasonality, suitable for data where seasonal effects scale with trend.
    - **Yearly seasonality** was enabled to capture annual price cycles.
    - **Custom seasonality** (monsoon_season) was added with a Fourier order of 10 to explicitly model the potential influence of monsoon months.
    - **External regressors** such as Scrap_Mandi_Price, Sponge_Iron_DRCLO_Price, and lag-based features were included to improve predictive accuracy.
- **Model Training and Prediction:**
    - The dataset was split into 70% training and 30% testing based on chronological order.
    - The model was trained on the training set with all specified seasonal and regressor inputs.
    - Predictions were made across the **entire historical range**, and performance was evaluated specifically on the test portion.
- **Evaluation:**

  To evaluate model performance:

    - **R^2 Score** was calculated to determine the proportion of price variance explained by the model.
    - **Root Mean Squared Error (RMSE)** was computed to assess the average prediction error.

A plot was also generated to visually compare actual vs. predicted BF prices during the test period.

RESULTS:

<u>IF ROUTE:</u>

LSTM Model R^2:0.9584

Prophet Model R^2: 0.95

Random Forest Model R^2:0.9655

<u>BF ROUTE:</u>

LSTM Model R^2:0.9844

Random Forest Model R^2:0.9057

Prophet Model R^2: 0.89

Initially, the Prophet model showed promising results when trained on the univariate time series data, especially due to its ability to capture strong seasonality effectively. Given the consistent seasonal patterns in the dataset, Prophet was a reliable starting point for initial forecasting. However, after analyzing feature correlations, several highly correlated external variables were identified that influence the target variable. These variables were incorporated into a multivariate LSTM model, allowing it to learn from additional signals beyond just historical prices. The LSTM model, with its capability to model non-linear dependencies and long-term memory, outperformed Prophet once these correlated inputs were introduced. It captured both direct and delayed effects of external variables, resulting in significantly lower prediction errors (MAE, RMSE).

Thus, while Prophet worked well in the univariate setting, the LSTM model proved more effective in a multivariate setup, especially when enriched with meaningful, high-correlation features.

So, now our next object is to predict price for next 60 days' with the best model.

METHODOLOGY:

- **Data Preparation:** The analysis begins with a cleaned and merged dataset containing BF_Price and its highly correlated variables:
    - Scrap_Mandi_Price
    - Sponge_Iron_DRCLO_Price
    - Sponge_Iron_PDRI_Price

The dataset was sorted chronologically, and the 'Date' column was set as the index for time series processing.

- **Data Normalization:** All feature columns were scaled using the MinMaxScaler to transform values into the [0, 1] range. This ensures that the LSTM model treats all variables uniformly without bias toward higher-magnitude features.
- **Sequence Generation:**
  To train the model for time series prediction, a sliding window approach was applied**:**
    - A window of the past 30 days (window_size = 30) was used to predict the next day's values.
    - Input X consisted of sequences of 30 consecutive days.
    - Output y corresponded to the value immediately following each window.
- **LSTM Model Architecture:**
  A sequential LSTM model was constructed with the following architecture:
    - **LSTM Layer**: 64 units with ReLU activation.
    - **Dense Output Layer**: One neuron per target variable (4 total).

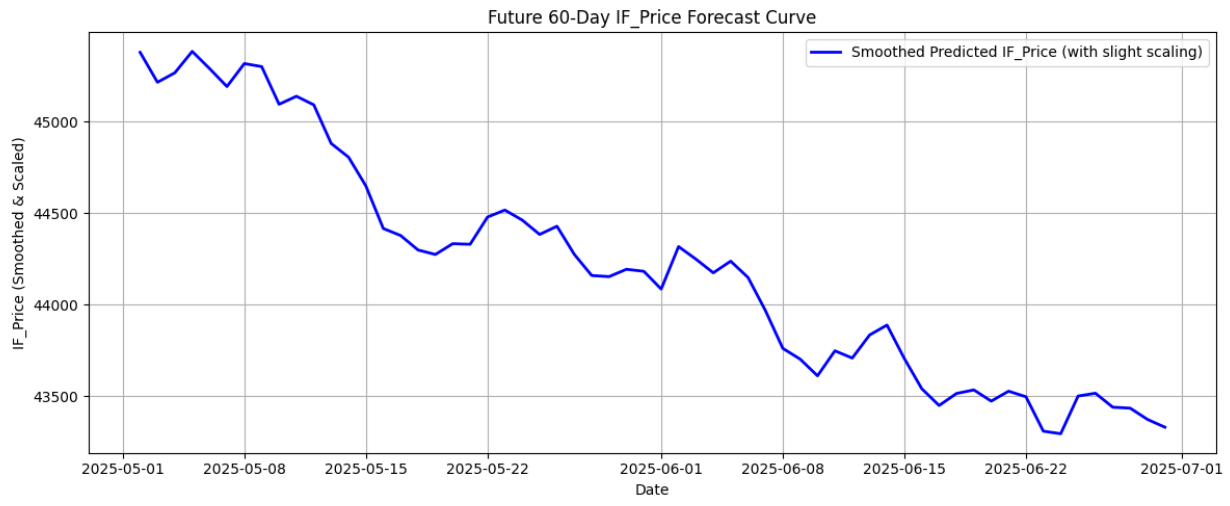
The model was compiled using the **Adam optimizer** and **Mean Squared Error (MSE)** as the loss function.

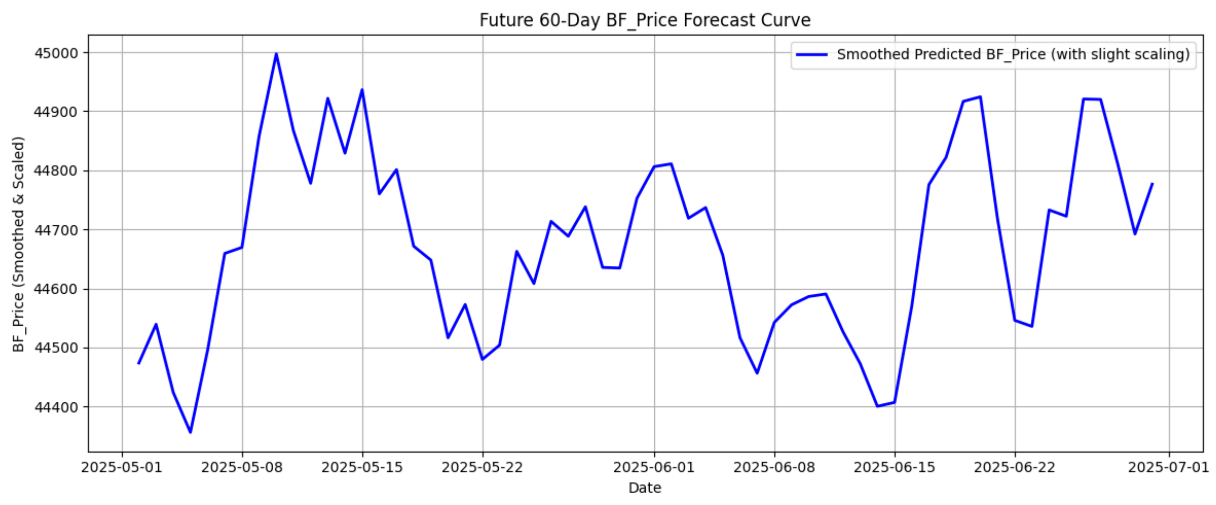
- **Model Training:**

  The model was trained for 50 epochs with a batch size of 16. To ensure reproducibility, a fixed random seed was set for Python, NumPy, and TensorFlow operations.

- **Forecasting Future Values:**
  After training, the last 30 days of input data were used to forecast the next 60 days step-by-step:
  - Each day's forecast was appended to the input and used to predict the following day (recursive forecasting).
  - The predictions were then inverse-transformed back to their original scale.
- **Regression-Based Adjustment of BF_Price:**
  To enhance BF_Price prediction accuracy, a post-model regression step was introduced:
  - A linear regression model was trained on historical data using the three input variables (Scrap, DRCLO, PDRI) to predict BF_Price.
  - This model was then used to re-calculate the future BF_Price from the LSTM-predicted values of the three inputs.
- **Forecast Smoothing and Visualization:**
  To improve visualization:
  - A small random noise (1% scale) was applied to the predicted BF_Price to avoid an artificial flat curve.
  - A rolling average (window size = 5) was applied to smooth the forecast.

The smoothed 60-day forecast was plotted to clearly visualize the price trend.





Future 60-Day Predicted Price source: FUTURE 60 DAYS PREDICTED PRICES USING LSTM

CONCLUSION:

Initial analysis showed that the Prophet model performed well when only univariate time series data was used. However, upon incorporating highly correlated variables, the LSTM model produced better predicting results. This suggests that while Prophet is suitable for capturing seasonality in univariate data, LSTM is more effective in a multivariate setting where inter-variable relationships matter.

Future work can include exploring models like SARIMA , RNN etc. which may offer even more accurate predictions. Most importantly, for improving the overall performance of any forecasting model, the availability of authentic, consistent and high-quality data is essential.

Challenges Faced and Solutions Implemented

- **Handling Missing Values:**
  The time series data contained several missing values, which could have impacted model accuracy. We addressed this by using **interpolation**, **backward fill**, and **forward fill** techniques to impute the missing entries, ensuring continuity in the time series.
- **Capturing Seasonality:**
  Initially, our models failed to capture **seasonal patterns** effectively, leading to underperformance in periods with known seasonal effects. To resolve this:
  - We **engineered a custom feature** called "is_monsoon" to explicitly indicate the monsoon period in the data.
  - We also **added a seasonality component** in our time series forecasting model (e.g., Prophet), allowing it to learn and adjust to recurring seasonal trends.

REFERENCES:

- **Mentor-provided dataset** (2025).
  This dataset was collected and shared by our mentor which includes raw and processed data relevant to the project domain.
- **Wikipedia** (n.d.).
  Used for background research and conceptual understanding of key topics.
- **Google Search** (n.d.).
  Utilized to discover supplementary information, validate data trends and academic sources relevant to the project topic.

GitHub : https://github.com/Shuvam-Maity/IDEAS_Internship_TMT_Price_Prediction