

Project Report: AMS 559 EV Charging Forecasting

1. Project Overview

1.1 Objective

The project report aims to create and assess a model for predicting the hourly charging demand of electric vehicles (EVs). This includes understanding the data, preprocessing, feature engineering, model selection, and training. The report evaluates the model's performance and predicts hourly energy consumption for December.

1.2 Datasets

The training dataset, stored in "train.csv," spans from January 1, 2020, to December 1, 2020, covering 11 months of charging data. It includes features such as 'connectionTime', 'disconnectTime', 'doneChargingTime', 'kWhDelivered', 'sessionID', 'siteID', 'spaceID', 'stationID', and other relevant parameters necessary for forecasting hourly charging demand for electric vehicles (EVs).

2. Data Preprocessing

2.1 Data Cleaning

In the data preprocessing stage, NaN values were converted to 0 to ensure that the dataset was ready for analysis and modeling. We considered removing NaN values altogether. However, this approach would have led to data loss, which in turn would have decreased the size of our training data, potentially affecting the robustness of our training model. Additionally, removing NaN values did not seem appropriate as the data is not incorrect; rather, it tapers down after the first quarter due to the pandemic and subsequent lockdown.

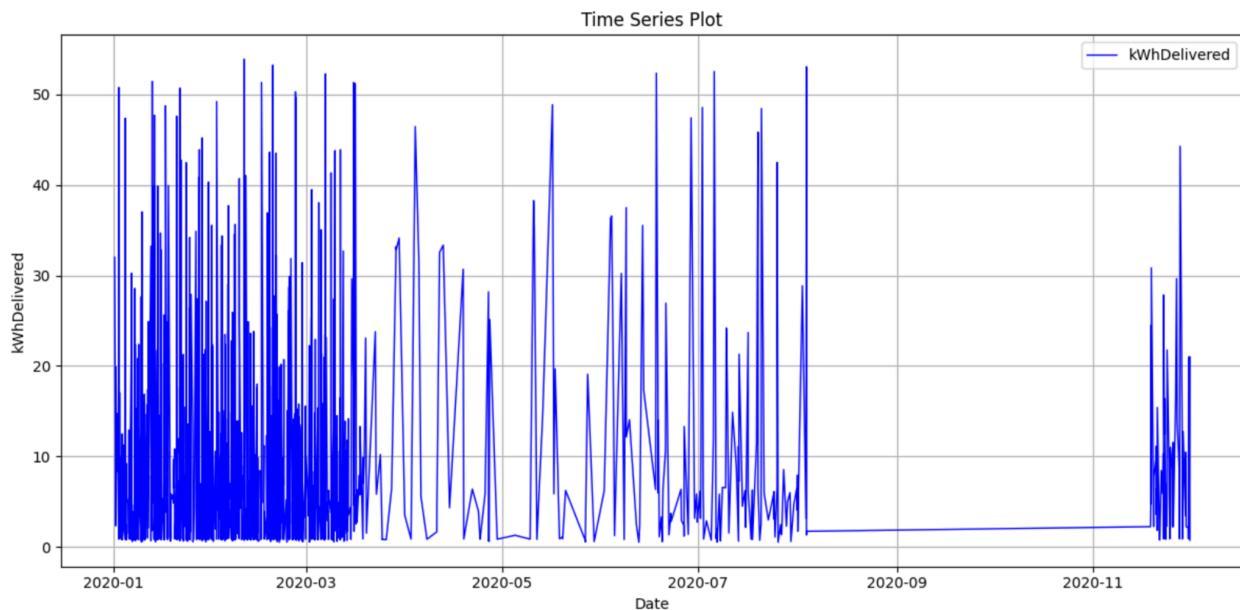


Fig 2.1.1 Time Series Plot of kWhDelivered vs connectionTime Date

The scatter plot (Fig 2.1.2) illustrates the distribution of 'kWhDelivered' data points, revealing potential outliers marked in red cross. These outliers represent instances where the kWh delivered significantly deviates from the typical range observed in the dataset. Such deviations might be indicative of errors in data collection, anomalies in energy consumption, or exceptional events affecting the charging process. Identifying and examining these outliers is crucial for ensuring data quality and understanding the underlying factors contributing to their occurrence. By scrutinizing these data points, we can gain valuable insights into the charging patterns, detect irregularities, and make informed decisions regarding data preprocessing or model adjustments to enhance the accuracy and reliability of our analysis.

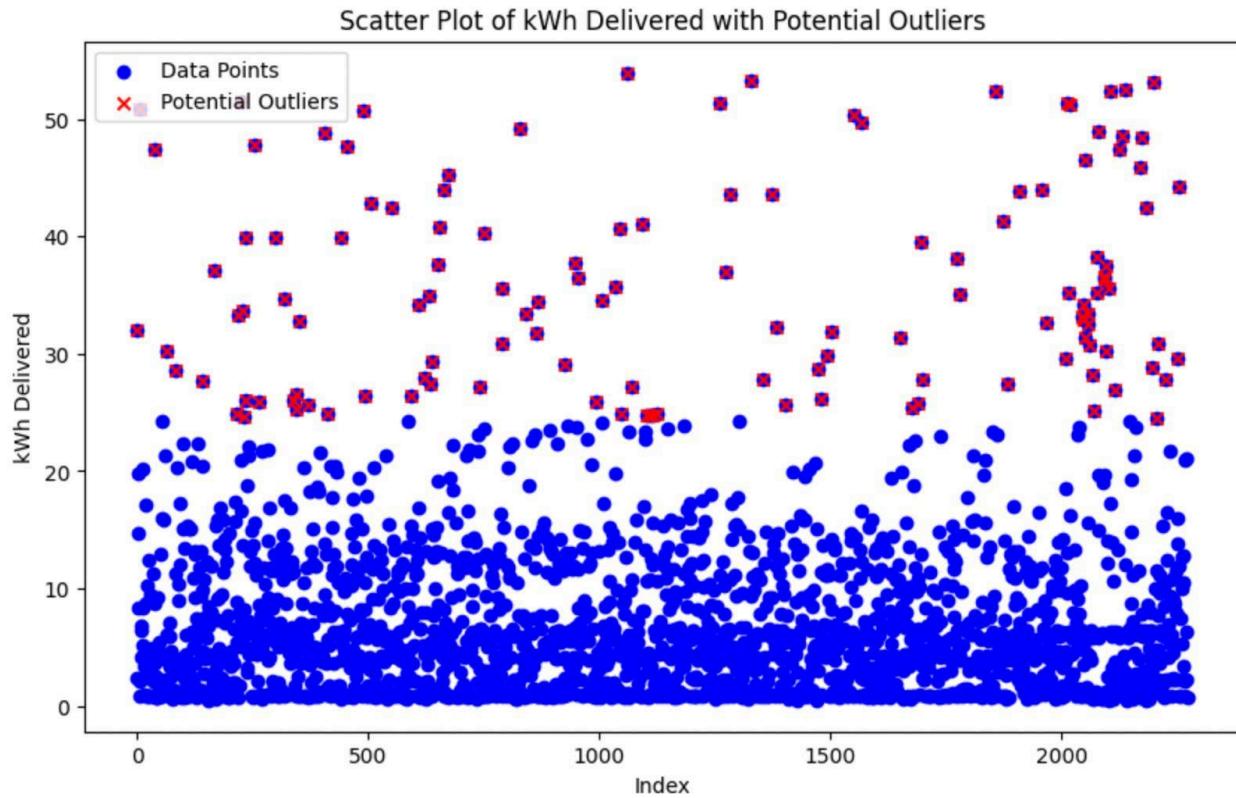


Fig 2.1.2 Scatter Plot of kWhDelivered with potential outliers

2.2 Feature Engineering

For the prediction of hourly EV power demand, we mainly focused on 3 columns from the original data - 'connectionTime', 'disconnectTime', and 'kWhdelivered'. The objective was to process the data to get hourly kWh delivered values, ensuring proportional distribution within each hour. Additionally, we addressed missing timestamps in the data and filled them with appropriate values.

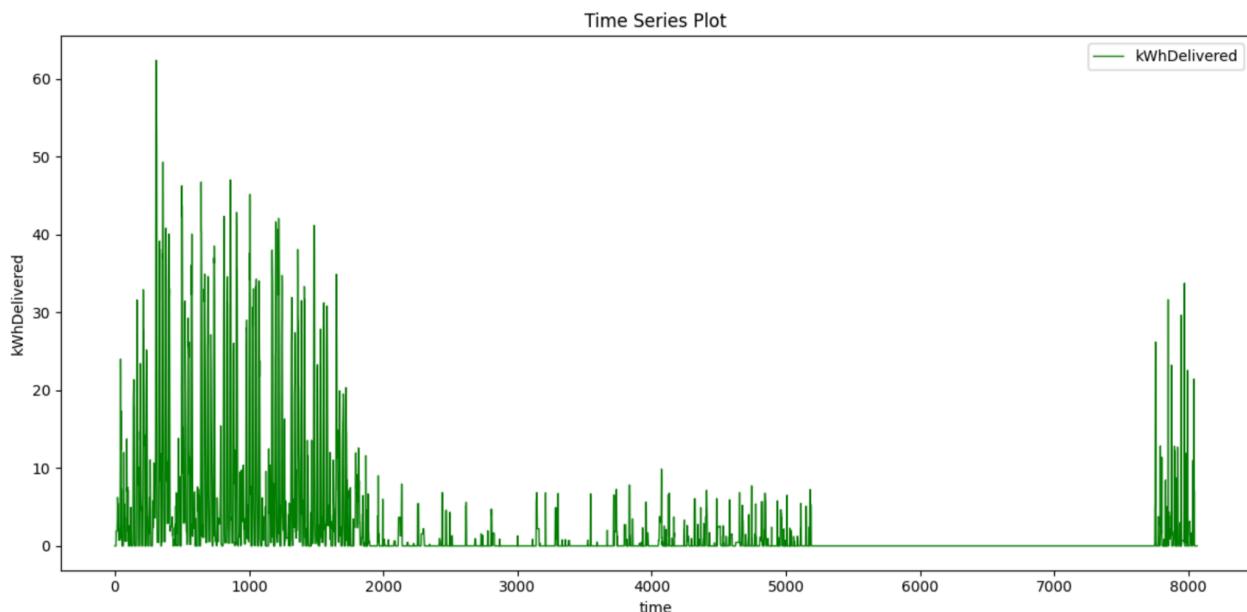
The first step involved dividing the charging data into hourly intervals while ensuring that the kWh delivered was proportionally distributed within each hour. This step was crucial as it accurately represented the charging behavior over time. Proportional distribution ensured that the kWh delivered was allocated appropriately, considering the duration of charging within each hour. For this, we first calculated the energy delivered per second. Using this quantity simplified the task of distributing the energy proportionally.

	time	kWhDelivered
0	2020-01-01 09:00:00+00:00	0.566942
1	2020-01-01 10:00:00+00:00	1.914626
2	2020-01-01 11:00:00+00:00	1.914626
3	2020-01-01 12:00:00+00:00	1.914626
4	2020-01-01 13:00:00+00:00	1.914626

Fig 2.2.1 First 5 data points

For example, if a vehicle was charged from 9:40 am to 10:15 am, to calculate the energy demand for the hour 9:00 am to 10:00 am, we first found the energy delivered per second and then multiplied it by the total number of seconds for which the vehicle was charged in that hour (e.g., 20*60 seconds). So, the energy demand would be `energy_delivered_per_sec * 20 * 60`.

Following the proportional division, the data was grouped to aggregate kWh delivered values for each hour. This step was crucial because there may have been multiple sessions for the same hour, and we needed the sum of those entries to get the actual final demand for each hour. After grouping the data, missing timestamps were identified and filled with appropriate values. Here, we filled the missing values with zeros, assuming that the demand for that period was zero.

**Fig 2.2.2** Time Series Plot for kWhDelivered vs time

Furthermore, we extracted several features from the timestamp data to enhance our understanding and analysis of the EV charging demand dataset. Here's an explanation of each feature:

- Date: Extracted from the timestamp to represent the date of each data entry. This allowed us to analyze trends and patterns based on the calendar date.
- Hour: Extracted from the timestamp to represent the hour of the day for each data entry. This allowed us to analyze charging behavior based on different times of the day.
- Day_of_week: Extracted from the timestamp to represent the day of the week (0 = Monday, 6 = Sunday) for each data entry. This allowed us to analyze charging patterns based on the day of the week.
- Day_of_year: Extracted from the timestamp to represent the day of the year (1-365) for each data entry. This allowed us to analyze charging behavior based on the time of year.
- Day_of_month: Extracted from the timestamp to represent the day of the month for each data entry. This allowed us to analyze charging patterns based on different days of the month.
- Month: Extracted from the timestamp to represent the month (1-12) for each data entry. This allowed us to analyze charging behavior based on different months of the year.
- Quarter: Extracted from the timestamp to represent the quarter (1-4) of the year for each data entry. This allowed us to analyze charging patterns based on different quarters of the year.
- Year: Extracted from the timestamp to represent the year for each data entry. This allowed us to analyze charging behavior over different years.

These extracted features provide additional dimensions for analyzing and understanding the EV charging demand dataset, enabling us to derive insights that may not have been apparent from the original data alone.

3. Model Selection

When choosing a forecasting model, we need to consider the characteristics of the data, such as seasonality, trend, and noise level, as well as the computational resources available. Experimenting with multiple models is often beneficial for achieving the best performance so we focused on that and experimented on it.

- A. **Linear Regression** - We explored linear regression as a method for predicting EV charging demand based on various time-related features. We split the dataset into training and testing sets using an 80-20 split, preserving the temporal order of the data. The features used for prediction were the hour of the day, day of the week, day of the year, day of the month, month, quarter, and year. These features were used to train a linear regression model from the scikit-learn library. The model was trained on the training set and then used to make predictions on the testing set. The

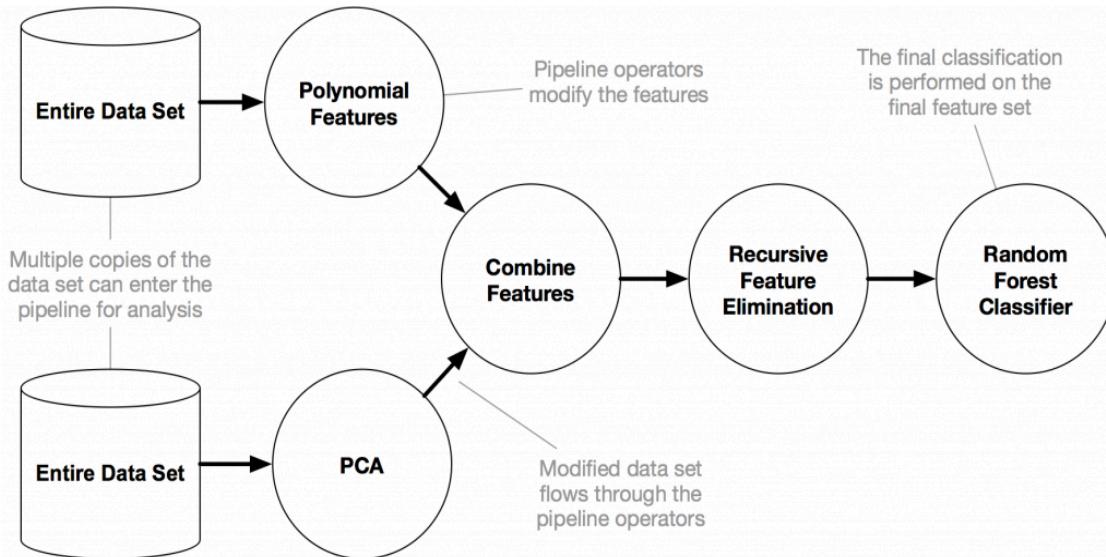
mean squared error (MSE) was calculated to evaluate the performance of the model. The MSE measures the average squared difference between the actual and predicted values, with lower values indicating better performance. Overall, linear regression provided a baseline for predicting EV charging demand based on time-related features, but more complex models may be needed for more accurate predictions.

- B. **LightGBM** - It employs a leaf-wise tree construction strategy, optimizing loss reduction and reducing memory usage. Utilizing gradient-based sampling and regularization techniques like L1 and L2, LightGBM prevents overfitting. It is known for its efficiency and scalability, LightGBM is ideal for large-scale datasets and diverse machine learning tasks, effectively tackling complex predictive modeling challenges.

We demonstrate the implementation of LightGBM for a regression task on our dataset. Initially, the dataset is split into features (X) and the target variable (y), with further partitioning into training and testing sets using the `train_test_split` function. Subsequently, the data is converted into the LightGBM dataset format, setting up the necessary data structures for training. Hyperparameters are defined, including boosting type, objective, and metric, along with parameters like the number of leaves, learning rate, and early stopping rounds. The model is then trained using the `lgb.train` function, and predictions are made on the testing set. Finally, the mean squared error is calculated to evaluate the model's performance.

Preparation of the dataset for training a LightGBM model involves splitting the dataset into features (X) and the target variable (y), including temporal features such as 'hour', 'day_of_week', 'day_of_year', 'day_of_month', 'month', 'quarter', and 'year'. The dataset is divided into training and testing sets using the `train_test_split` function from scikit-learn, with 20% of the data reserved for testing and a random state set for reproducibility. Subsequently, the data is converted into LightGBM dataset format using the `lgb.Dataset` function, facilitating training. This encapsulates the foundational steps for preparing the dataset and initializing the LightGBM model for subsequent training and evaluation.

- C. **TPOT Regressor** - We employed the TPOT library for automated machine learning (AutoML) to enhance our predictive model for EV charging demand. TPOT is designed to explore a wide range of machine learning pipelines and hyperparameters to identify the best-performing model for a given dataset. In our implementation, we used `TPOTRegressor` to predict kWh delivered based on features such as the hour, day of the week, day of the year, day of the month, month, quarter, and year. We split the dataset into training and testing sets, trained the `TPOTRegressor` on the training data, and then evaluated its performance using the mean squared error (MSE) metric on the test data. TPOT's automated approach saves time and effort by intelligently searching through the model space, potentially uncovering more effective models than those manually selected. Additionally, TPOT allows us to export the best pipeline as a Python script, enabling reproducibility and future use in similar projects.

**Fig 3.1** TPOT pipeline [4]

4. Performance Evaluation and Diagnostics

For performance evaluation, we utilized the mean squared error (MSE), a metric that quantifies the average squared difference between predicted and actual values. Additionally, we used a graph to visualize the model's performance. The graph includes a black dashed line representing perfect predictions. Points above this line indicate overestimation by the model, while points below indicate underestimation. The proximity of points to the black line reflects the accuracy of the model's predictions, with closer points indicating better performance.

A. Linear Regression -

The mean squared error (MSE) from the linear regression model was 25.16, indicating a significant discrepancy between the predicted and actual kWh delivered values. This high MSE suggests that the model's predictions were far from the actual values, making it a poor fit for the data. The graph's points are scattered widely and do not align with the black dashed line representing perfect predictions. This lack of alignment indicates that the model's predictions were inconsistent and did not capture the true pattern of the data.

The high MSE and the scattered points in the graph suggest that the linear regression model is underfitting the data. Underfitting occurs when the model is too simple to capture the underlying patterns in the data, resulting in poor performance. In this case, the linear regression model is unable to accurately predict EV charging demand based on the given features, indicating a need for a more complex model.

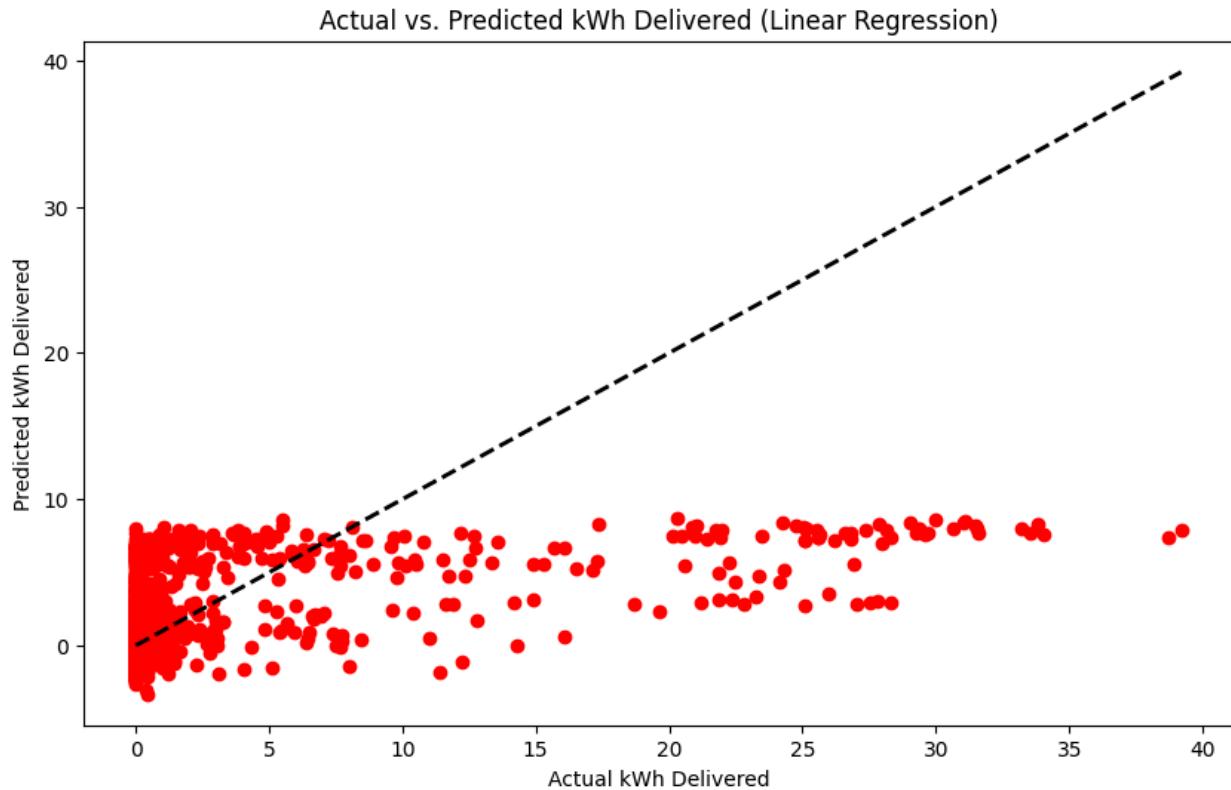


Fig 4.1 Actual vs Predicted kWh Delivered Using Linear Regression

B. LightGBM -

The mean squared error (MSE) for the LightGBM model was 3.12, which is significantly lower than the MSE of 25.16 for the linear regression model. This indicates that the LightGBM model outperformed the linear regression model in predicting EV charging demand. The superior performance of LightGBM can be attributed to its ability to handle complex relationships in the data. Unlike linear regression, which assumes a linear relationship between the features and the target variable, LightGBM can capture non-linear relationships and interactions between features. This flexibility allows LightGBM to model the complexities of EV charging demand better, resulting in more accurate predictions.

In the graph for the LightGBM model, most of the points lie closer to the black dashed line representing perfect predictions. This suggests that the model's predictions are more aligned with the actual values, indicating a better fit to the data. Since most points are correctly predicted, it is unlikely that the model is underfitting. However, we need to be cautious about overfitting, which occurs when the model learns the noise in the training data rather than the underlying patterns.

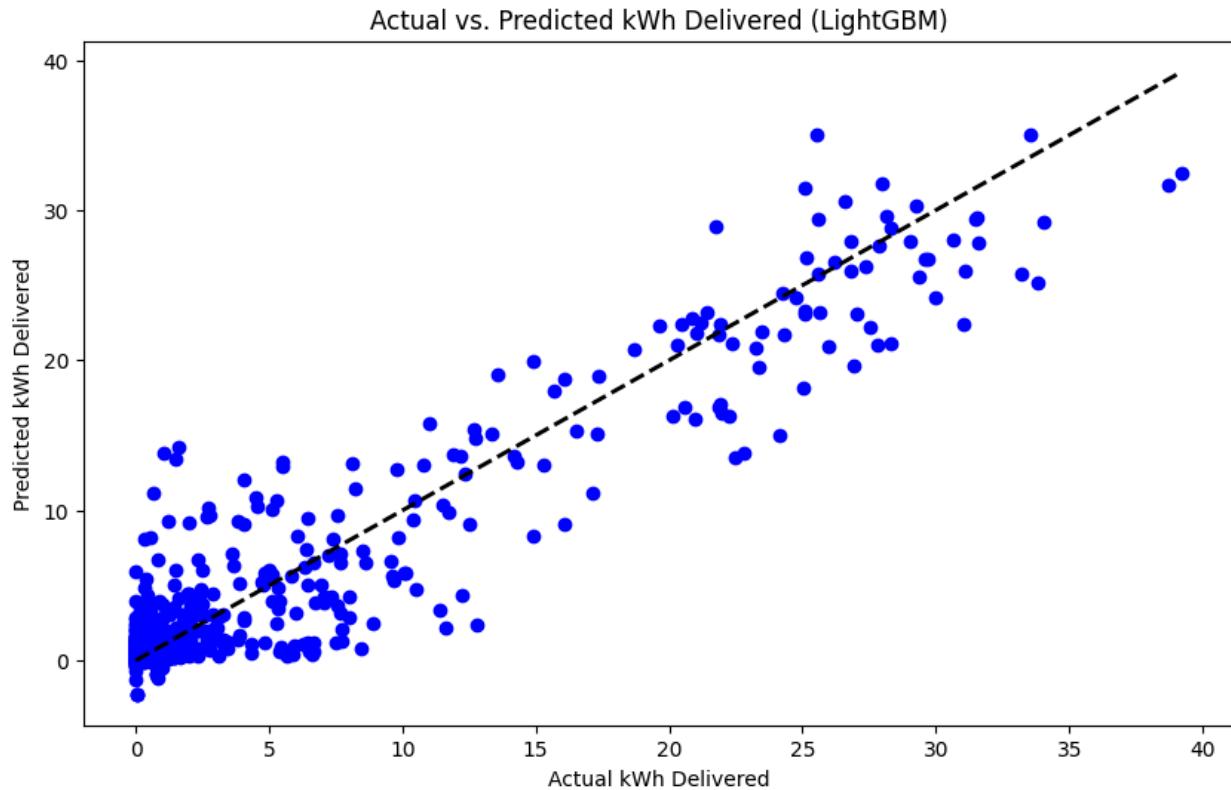


Fig 4.2 Actual vs Predicted kWh Delivered Using LightGBM

C. TPOTRegressor -

In the TPOTRegressor model, after exploring various machine learning pipelines over 15 generations, TPOT selected the RandomForestRegressor as the best-performing model for predicting EV charging demand. This selection was based on the RandomForestRegressor's ability to minimize the Mean Squared Error (MSE) and achieve the most accurate predictions compared to other models evaluated by TPOT.

The RandomForestRegressor, with its ensemble of decision trees, proved to be a suitable choice for this task due to its ability to capture complex relationships in the data and handle non-linearities effectively. This flexibility allowed the model to achieve a lower MSE of 2.0, indicating superior performance compared to the linear regression and LightGBM models.

In the graph for the RandomForestRegressor model, the majority of points cluster closely around the black dashed line, indicating that the model's predictions align well with the actual values. This clustering suggests that the RandomForestRegressor model has achieved a good balance between capturing the underlying patterns in the data and avoiding overfitting, making it a reliable choice for predicting EV charging demand.

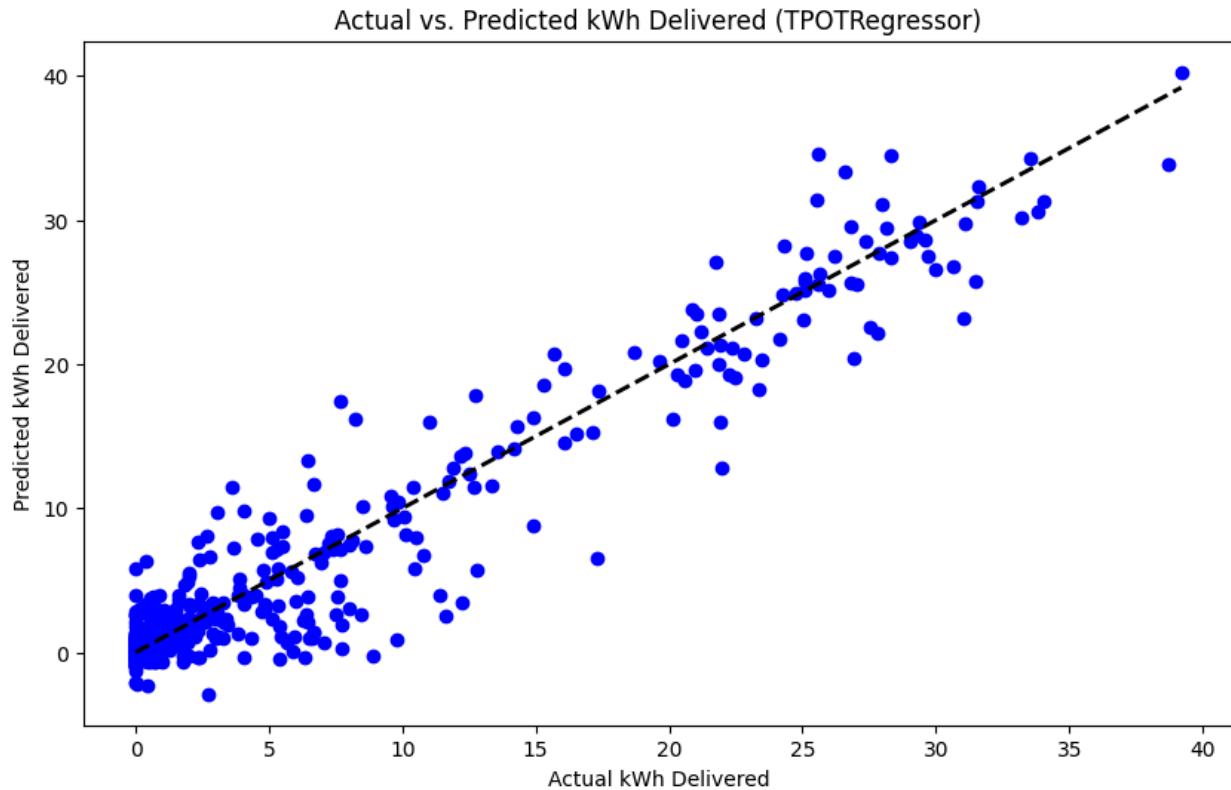


Fig 4.3 Actual vs kWh Delivered Using TPOTRegressor

After testing with 80-20 data, we achieved a good Mean Squared Error (MSE) using TPOT. Consequently, we opted to use TPOT for predicting the values of December for 744 hours, covering the period from 1st December at 12 pm to 1st January 2021 at 12 pm. On Kaggle, our model obtained an MSE of 3.13, indicating its effectiveness in forecasting EV charging demand for the specified time frame. This MSE value signifies that the model's predictions were, on average, within approximately 1.77 kWh of the actual values, demonstrating its accuracy in estimating hourly charging demand for electric vehicles.

Models	Without Outlier Detection (MSE)	With Outlier Detection (MSE)
Linear Regression	35.17	25.16
LightGBM	5.2	3.12
TPOTRegressor	3.02	2.0
Kaggle Score	3.13	3.15

Fig 4.4 Output table with the MSE values

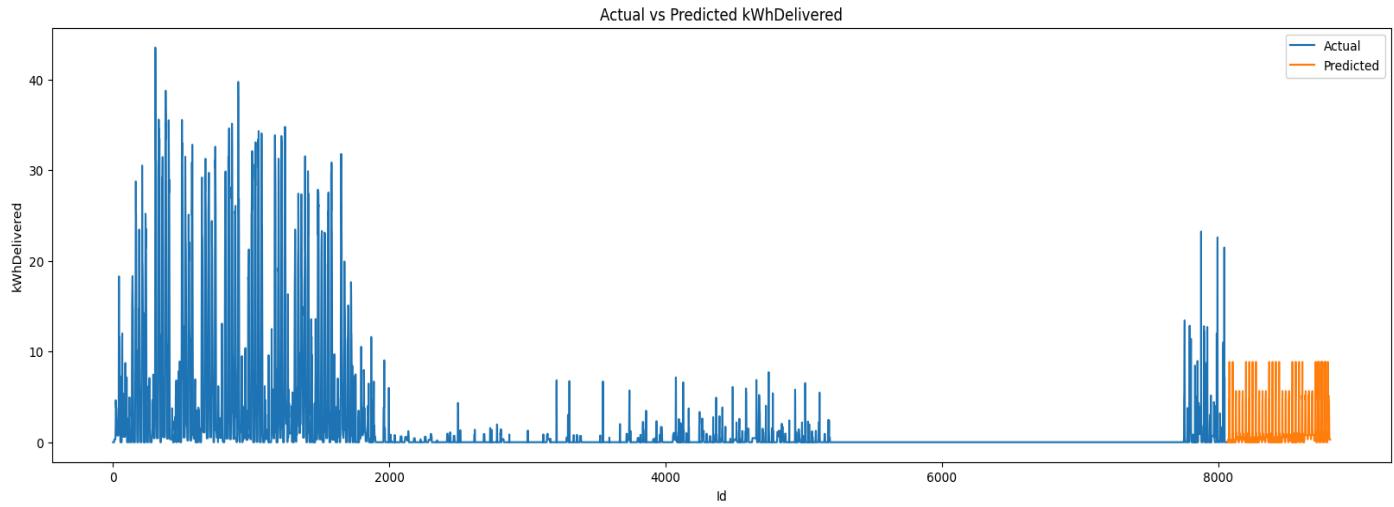


Fig 4.5 Actual vs Predicted kWhDelivered from Jan 1, 2020 to Jan 1, 2021, on an hourly basis

The application of outlier detection significantly enhanced model performance during training. However, this improvement did not translate to similar advancements in tested data. This discrepancy suggests that the outliers identified in the training data may not have been representative of the test set or that the model's reliance on these outliers led to overfitting.

6. Conclusion

In conclusion, our project on forecasting hourly EV charging demand using machine learning models yielded valuable insights and outcomes. We began by exploring various models, including linear regression, LightGBM, RandomForestRegressor, and TPOTRegressor, to predict EV charging demand accurately. Through data preprocessing, feature engineering, and model evaluation, we identified the TPOTRegressor model as the most effective on the Kaggle competition dataset.

7. Supporting Materials

1. [Kaggle](#)
2. [Data Cleaning](#)
3. [Models for Time series](#)
4. [AutoML](#)
5. [Time series forecast](#)
6. [ChatGPT](#)