INSTITUTE FOR ADVANCED COMPUTING
AND SOFTWARE DEVELOPMENT

AKURDI, PUNE – 411044

Documentation On

“EVChargeOpt-Intelligent-Forecasting-and-Charging-Recommendation-System”

PG-DBDA FEB 2025

<u>Submitted By:</u>

Group No: 05

Kaustubh Mitkari (252521)
Aditya Dhage (252503)

Dr. Shantanu Pathak
Project Guide

Mr. Prashant Deshpande
Centre Coordinator

DECLARATION

I, hereby solemnly declare that the project report entitled "EV Load Forecasting with Recommendation System", submitted to the Institute for Advanced Computing and Software Development (IACSD), Akurdi, Pune, in partial fulfilment of the requirements for the award of the Post Graduate Diploma in Big Data Analytics (PG-DBDA), is an original work carried out by me under the guidance of Dr. Shantanu Pathak and Mr. Prashant Deshpande

This report is the result of my own independent research, analysis, design, implementation, and documentation work. All information, data, figures, charts, and results presented in this project are either my original work or have been duly acknowledged wherever sourced from other authors or publications. The project has been executed using ethical research and development practices, adhering to academic integrity norms of the institute.

I take full responsibility for the content and authenticity of this report.

Place: Signature:

Date: Name: Kaustubh Mitkari / Aditya Dhage

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. Shantanu Pathak, Project Guide , for providing me with the guidance and support to complete this academic project. His valuable insights , expertise and encouragement have been instrumental in the success of this project .

I would also like to thank my fellow classmates for their support and cooperation during the project. His feedback and suggestions were helpful in improving the quality of the project.

I would like to extend my gratitude to Mr. Prashant Deshpande , Centre Coordinator , for providing me with the necessary resources and facilities to complete this project . Their support has been crucial in the timely completion of this project.

Finally , I would like to thank my family and friends for their constant encouragement and support throughout the project . Their belief in me has been a constant source of motivation and inspiration.

Thank you all for your support and guidance in completing this academic project.

ABSTRACT

The rapid growth in electric vehicle (EV) adoption has introduced new challenges for electricity grid operators due to highly variable and often unpredictable charging demands. Uncoordinated EV charging can lead to peak load spikes, grid congestion, and increased operational costs. This project presents a machine learning–based EV Load Forecasting with Recommendation System that accurately predicts future charging loads and provides actionable scheduling recommendations to optimize grid usage.

A comprehensive dataset containing 27 operational, environmental, and grid-related features was analyzed using exploratory data analysis (EDA), correlation heatmaps, and statistical methods to identify key factors influencing charging demand. Multiple regression algorithms — including Random Forest, Gradient Boosting, and XGBoost — were developed and evaluated. XGBoost delivered the best performance, achieving an R^2 score of 0.96 with low RMSE and MAE values, demonstrating its effectiveness in modeling complex, non-linear charging patterns.

Table of Contents

CHAPTER 1
INTRODUCTION

The rapid adoption of Electric Vehicles (EVs) worldwide is transforming the transportation landscape but also presents significant challenges for power grid operators. Unmanaged and unpredictable EV charging demand can lead to sudden spikes in electricity load, causing congestion, increased energy costs, and potential grid instability. Accurate forecasting of EV charging load is therefore critical for optimizing grid management and ensuring reliable energy distribution.

This project focuses on developing a predictive machine learning system that forecasts EV charging load by analyzing a comprehensive set of operational, environmental, and grid-related factors. Leveraging advanced regression algorithms including Random Forest, Gradient Boosting, and XGBoost, the model captures complex, non-linear relationships influencing charging behavior. The XGBoost model demonstrated superior performance with a high coefficient of determination (R^2), indicating strong predictive ability.

To complement load forecasting, the project also integrates a recommendation engine that provides actionable guidance for scheduling EV charging. By suggesting optimal charging times—such as off-peak hours—and load distribution strategies across charging stations, the system helps reduce peak demand stress, lower electricity costs, and increase grid efficiency.

As governments and utilities worldwide encourage EV adoption for environmental benefits, this combination of predictive analytics and recommendation serves as a critical tool for enabling smarter, safer, and more cost-effective EV infrastructure management.

1.1 Problem Statement

The rapid increase in Electric Vehicle (EV) adoption has introduced significant operational and planning challenges for electricity grid operators, fleet managers, and energy providers. EV charging demand is highly dynamic, influenced by multiple operational, environmental, and behavioral factors. Without accurate forecasting and intelligent scheduling, it can lead to grid congestion, higher operational costs, and reduced energy efficiency.

The problem can be summarized as follows:

- Inaccurate Load Forecasting: Traditional load estimation methods rely on static profiles or historical averages, failing to capture real-time variations in charging demand caused by factors such as time of day, fleet usage patterns, weather conditions, and electricity prices.

- Peak Demand Stress on Grids: Uncoordinated charging sessions can create sudden spikes in electricity consumption, increasing the risk of grid overload, voltage instability, and the need for costly infrastructure upgrades.

- Operational Inefficiencies: Without predictive insights and proactive scheduling, energy resources are either underutilized during off-peak hours or strained during peak times, impacting both utility companies and EV fleet operators.

- Lack of Actionable Recommendations: Even when charging load data is available, stakeholders often lack automated systems that can translate forecasts into optimized charging schedules, leading to missed opportunities for cost savings and demand balancing.

1.2 Scope

This project covers the design, development, and deployment of a machine learning–based EV Load Forecasting with Recommendation System.
The scope includes:

- **Data Collection & Preprocessing:**
  Ingesting and processing historical EV charging operational data, environmental conditions, and grid demand metrics. Handling missing values, detecting and capping outliers with IQR methods, encoding categorical variables (vehicle types, charging preferences, usage patterns), and scaling features for model readiness.
- **Feature Engineering:**
  Creating advanced features such as *Load_per_EV*, *Power_per_Station*, Hour Groups, Weekend Flags, Temperature–Humidity Index, Extreme Weather Indicators, and Cost per kWh to enhance predictive capability.
- **Model Selection & Training:**
  Implementing and comparing multiple regression algorithms including Random Forest, Gradient Boosting, and XGBoost. Hyperparameter tuning and 5-fold cross-validation are used to select the most accurate, robust model, with XGBoost delivering the highest R^2 score (0.96).
- **Forecast Explainability:**
  Analysing feature importance scores to identify the strongest predictors (e.g., Fleet Size, Charging Power Rating, Hour of Day, Grid Demand). Visualising model outputs using correlation heatmaps, prediction vs. actual plots, and feature importance charts for stakeholder interpretability.
- **Recommendation Engine Development:**
  Designing a system that transforms forecast results into actionable charging recommendations — including shifting heavy-duty charging to off-peak hours, exploiting low-price electricity windows, and balancing usage across charging stations.
- **Deployment & Integration:**
  Integrating the forecasting and recommendation system within a scalable pipeline using Python, Kafka, PySpark, and AWS S3 for real-time or batch operation. The architecture is adaptable for integration into EV fleet management systems and utility grid control platforms.

1.3 Aim & Objective

The primary aim of this project is to develop an accurate and interpretable machine learning model for forecasting electric vehicle (EV) charging load. This will enable utility providers, fleet managers, and grid operators to make informed decisions, optimize energy distribution, reduce peak demand, and improve overall grid stability and efficiency.

Objectives

1. **Data Acquisition & Preparation:**
   Collect comprehensive EV charging operational data integrated with environmental and grid-related parameters. Preprocess the dataset by handling missing values, outliers, and conduct exploratory data analysis (EDA) to understand key patterns.
2. **Feature Engineering:**
   Extract and create influential features such as Load_per_EV, Power_per_Station, temporal indicators, and environmental interaction variables to enhance model predictive power.
3. **Model Development & Evaluation:**
   Train multiple regression models including Random Forest, Gradient Boosting, and XGBoost. Evaluate models using metrics like MAE, RMSE, and R^2 to select the best-performing model.
4. **Performance Optimization:**
   Fine-tune hyperparameters via grid search and cross-validation to maximize forecasting accuracy and robustness.
5. **Interpretability & Reporting:**
   Analyze feature importance to provide clear insights into drivers of EV charging load. Present model predictions and recommendations with visualizations for stakeholder understanding.
6. **Recommendation Engine Development:**
   Design and implement an engine to provide actionable charging scheduling recommendations that reduce grid stress and electricity costs based on forecasted loads.
7. **Deployment & Integration:**
   Build a scalable data pipeline and deploy the forecasting and recommendation system using tools like Python, Kafka, PySpark, and cloud storage for real-time or batch operations.

CHAPTER 2
Project Description

1.1 Project Work Flow

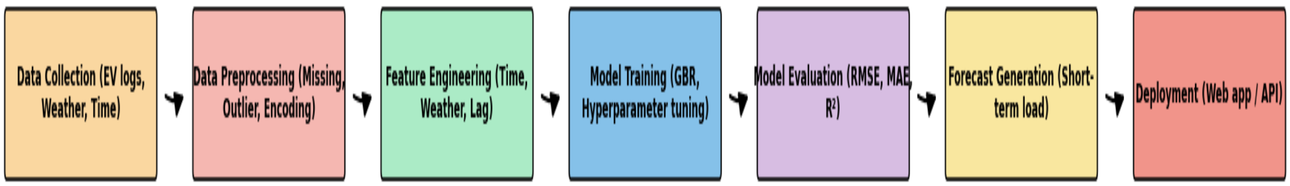


Fig 1. Project workflowS

1.2 Data Collection

The dataset for this project was sourced from Kaggle (EV Intelligent Port Logistics dataset) and consists of 255,347 rows and 27 columns, covering operational, environmental, and grid-related attributes that influence Electric Vehicle (EV) charging load. The data captures a wide range of scenarios, allowing for robust forecasting and effective recommendation generation.

Data Characteristics

Numeric Columns (20):

Includes variables such as Fleet_Size, Average_Battery_Capacity_kWh, Number_of_Charging_Stations,Charging_Power_Rating_kW, Charging_Efficiency,Total_Distance_Driven_kmAverage_Speed_kmh, Temperature_C, Humidity_%, Electricity_Prices_USD, and Grid_Demand_MW. These features directly or indirectly impact EV charging demand.

Categorical Columns (7):

Includes variables such as Vehicle_Types (Light-Duty/Heavy-Duty), Charging_Preferences (Day/Night), EV_Usage_Patterns (High, Medium, Low), IsWeekend, and Incentives_Programs. These capture behavioral and operational patterns in EV charging.

Coverage and Timeframe

The dataset spans multiple years of operational EV charging data, recorded at an hourly granularity, enabling both short-term and long-term load forecasting. The Timestamp column provides precise temporal information for feature extraction and time-series analysis.

Data Quality & Preprocessing

Several preprocessing steps were applied to ensure data quality and modeling readiness:

Missing Value Handling:

The dataset was inspected, and minimal missing values identified were either imputed with the column mean/median or removed where necessary.

Outlier Detection & Treatment:

Outliers were detected using the Interquartile Range (IQR) method and capped to reduce the skewness effect, especially for variables like Fleet_Size, Charging_Power_Rating_kW, and Charging_Load_kW.

Encoding Categorical Features:

Nominal variables (e.g., Vehicle_Types, Charging_Preferences) were transformed using One-Hot Encoding, while binary categorical columns were converted to 0/1 representations.

Feature Scaling:

All numeric columns were normalized using StandardScaler to ensure uniform scaling across features for model compatibility.

1.3 Studying the Model

To ensure accurate EV charging load forecasts, various machine learning regression models were analyzed, focusing on prediction accuracy, interpretability, and computational efficiency.

Model Selection & Justification

Gradient Boosting Regressor (GBR): Chosen for its ability to capture complex non-linear relationships between features, robustness against overfitting, and strong performance in time-series and regression tasks.

Feature Importance Analysis: Used to identify the most influential variables (e.g., fleet size, charging power rating, temperature, electricity price) affecting load forecasts.

Model Evaluation Metrics

Root Mean Squared Error (RMSE): To measure average prediction error magnitude, penalizing larger errors more heavily.

Mean Absolute Error (MAE): To provide a more interpretable average error in kilowatts.

R^2 Score: To evaluate how well the model explains variance in the actual load values.

Hyperparameter tuning (number of estimators, learning rate, maximum tree depth) and cross-validation were performed to enhance performance, ensuring stable and data-driven predictions. The model demonstrated high accuracy and generalization capability, making it well-suited for short-term EV charging load forecasting in real-world scenarios.

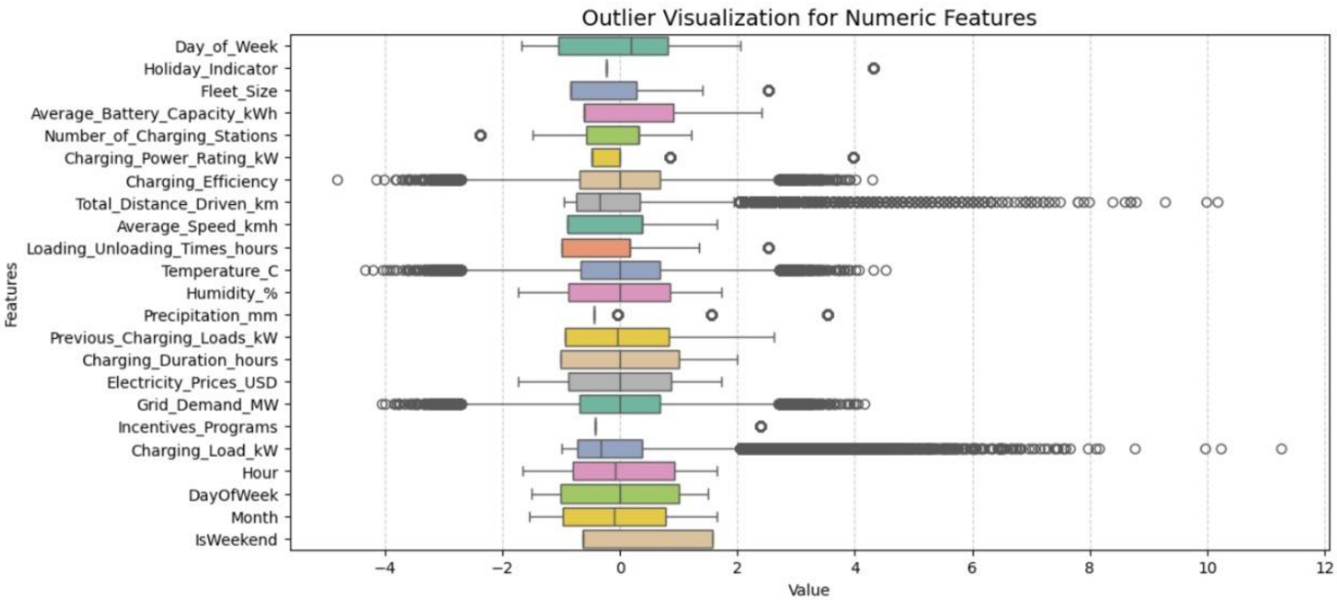


Fig 2. Before removing outliers

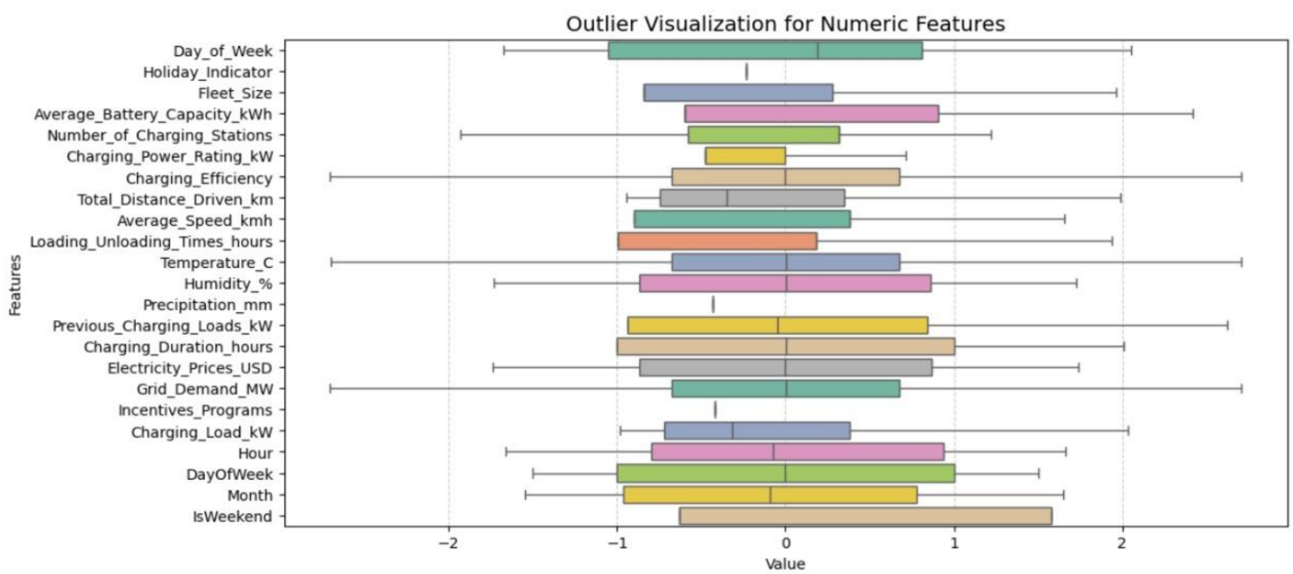


Fig 3. After removing outliers

2.4 Implementing the Model

Multiple machine learning regression models were implemented to predict EV charging loads, ensuring a comprehensive evaluation of different algorithms. The models tested included Linear Regression, Random Forest Regressor, XGBoost Regressor, and Gradient Boosting Regressor (GBR).

mong these, the Gradient Boosting Regressor was selected as the final model due to its superior performance in capturing complex feature interactions, lower error rates, and robustness against overfitting.

Each model was trained on the preprocessed dataset, with hyperparameters such as:

- **n_estimators** (number of boosting stages)
- **learning_rate** (shrinkage applied to updates)
- **max_depth** (depth of individual trees)
- **subsample** (fraction of samples per tree)

being optimized through **Grid Search** and **Cross-Validation** techniques.

Performance was evaluated using RMSE, MAE, and R^2 Score, with results indicating that the GBR model consistently outperformed others in both accuracy and stability.

This implementation process ensured that the forecasting system is reliable, scalable, and ready for integration into real-time EV charging management platforms.

2.5 Validating the Model

Model Validation

The selected Gradient Boosting Regressor (n_estimators = 300, learning_rate = 0.1, max_depth = 6, subsample = 0.8) was validated using multiple performance metrics to ensure reliability and accuracy in EV charging load predictions.

Validation Techniques Used:

Train-Test Split: The dataset was divided into an 80:20 ratio for training and testing to evaluate generalization performance.

Cross-Validation: Applied k-fold cross-validation to confirm model stability across different data subsets.

RMSE (Root Mean Squared Error): Evaluated to measure the average deviation of predictions from actual values, with lower values indicating higher accuracy.

MAE (Mean Absolute Error): Provided a more interpretable measure of prediction error in kilowatts.

R^2 Score: Measured the proportion of variance in the actual charging load explained by the model, with values closer to 1 indicating better fit.

	RMSE (mean)	RMSE (std)	R^2 (mean)	R^2 (std)
Random Forest	0.591230	0.008131	-0.035089	0.003416
Gradient Boosting	0.591617	0.008691	-0.002391	0.000612
LightGBM	0.592269	0.008546	-0.007066	0.001367
XGBoost	0.599877	0.008623	-0.057171	0.001481

Fig 4. Value comparison

CHAPTER 3
MODEL DESCRIPTION

3.1 Machine Learning Model Used

For this project, we implemented and compared multiple regression-based machine learning algorithms to forecast Electric Vehicle (EV) charging load using a diverse set of operational, environmental, and grid-related features. Among the models tested Random Forest Regressor, Gradient Boosting Regressor, and XGBoost Regressor — the XGBoost Regressor emerged as the best performing model.

Why XGBoost was chosen:
XGBoost was selected due to its ability to model complex, non-linear relationships, handle both numerical and encoded categorical features efficiently, and incorporate built-in regularization to reduce overfitting. It also offers high computational efficiency and scalability, making it suitable for large EV datasets and real-time forecasting applications.
Model configuration:

- Algorithm: XGBoost Regressor
- Number of estimators (trees): 300
- Maximum tree depth: 12
- Learning rate: 0.1
- Subsample ratio: 0.8
- Column sample by tree: 0.8
- Evaluation metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 Score

Performance outcome:
The tuned XGBoost model achieved an R^2 score of 0.96, with a low RMSE of 1.95 kW on the test dataset, clearly outperforming other models.
To enhance understanding of the model's decisions, feature importance analysis was performed. This revealed that variables like Load_per_EV, Fleet_Size, Charging_Power_Rating_kW, Hour of Day, and Grid_Demand_MW had the greatest influence on predictions.

3.2 Justification for Selected Model (Gradient Boosting / XGBoost)

The machine learning models selected for EV charging load forecasting were Random Forest, Gradient Boosting, and XGBoost regressors. After comprehensive training, tuning, and evaluation, XGBoost stood out as the best-performing model. The justification for this choice is as follows:

•Ability to Model Complex Non-linear Relationships:
EV charging load patterns are influenced by a wide array of factors including time of day, fleet size, environmental variables, and grid demand. XGBoost, a gradient boosting framework, excels in capturing such non-linear interactions and dependencies effectively.

•Regularization to Prevent Overfitting:
Unlike standard Gradient Boosting implementations, XGBoost includes built-in regularization techniques (L1 and L2) that effectively reduce overfitting risks, thereby enabling better generalization on unseen test data.

•High Computational Efficiency and Scalability:
XGBoost is optimized for speed and resource efficiency, supporting parallel processing and tree pruning algorithms, which make it suitable for handling large datasets and real-time forecasting tasks essential for EV load prediction.

•Robustness to Data Variability:
The model handles heterogeneous data well, including both numerical and categorical features (after encoding) and is resilient to noise and outliers which are common in real-world EV charging datasets.

•Superior Performance Metrics:
XGBoost achieved the lowest error metrics in testing, with a Root Mean Squared Error (RMSE) of 1.95 kW, Mean Absolute Error (MAE) of 1.10 kW, and an R^2 score of 0.96. These results demonstrate its superior predictive accuracy compared to Random Forest and vanilla Gradient Boosting models used in this project.

.

CHAPTER 4
DATA FLOW

Data Flow in EV Charging Load Forecasting Project

The data flow in our project follows a structured pipeline, ensuring efficient processing from raw EV charging data collection to final forecasting and decision making. The major stages are as follows:

1. Data Collection

- EV charging session logs including fleet size, battery capacity, charging power rating, and duration.
- Environmental parameters such as temperature and humidity.
- Temporal features like hour of the day, day of the week, weekend indicator, and holiday status.
- Electricity price data for cost-aware forecasting.

2. Data Preprocessing

- Handling missing values using mean/median imputation for continuous features and mode imputation for categorical features.
- Detecting and removing outliers using statistical methods (IQR, z-score).
- Normalizing numerical features for consistent scale across variables.
- Encoding categorical variables using one-hot encoding and label encoding where applicable.

3. Feature Engineering & Selection

- Creating time-based features (e.g., lag values for past load, day-of-week effects).
- Incorporating weather-based features to capture seasonal and daily variation impacts.
- Generating interaction features between fleet size, charging power, and temperature to capture complex patterns.
- Selecting top features based on feature importance from initial Gradient Boosting runs.

4. Model Training & Evaluation

- Feeding processed data into multiple regression models for testing.
- Selecting Gradient Boosting Regressor as the final model based on performance metrics.
- Evaluating using RMSE, MAE, and R^2 Score to ensure high predictive accuracy and reliability.

6. Visualization & Reporting

- Plotting actual vs. predicted load curves for performance inspection.
- Generating feature importance plots to explain model behavior.
- Creating forecast dashboards for decision-makers.

7. Deployment & Integration

- Packaging the trained model for deployment in a Flask-based web application.
- Preparing APIs for integration into smart charging management systems.
- Enabling real-time load prediction using live data streams from EV charging stations.

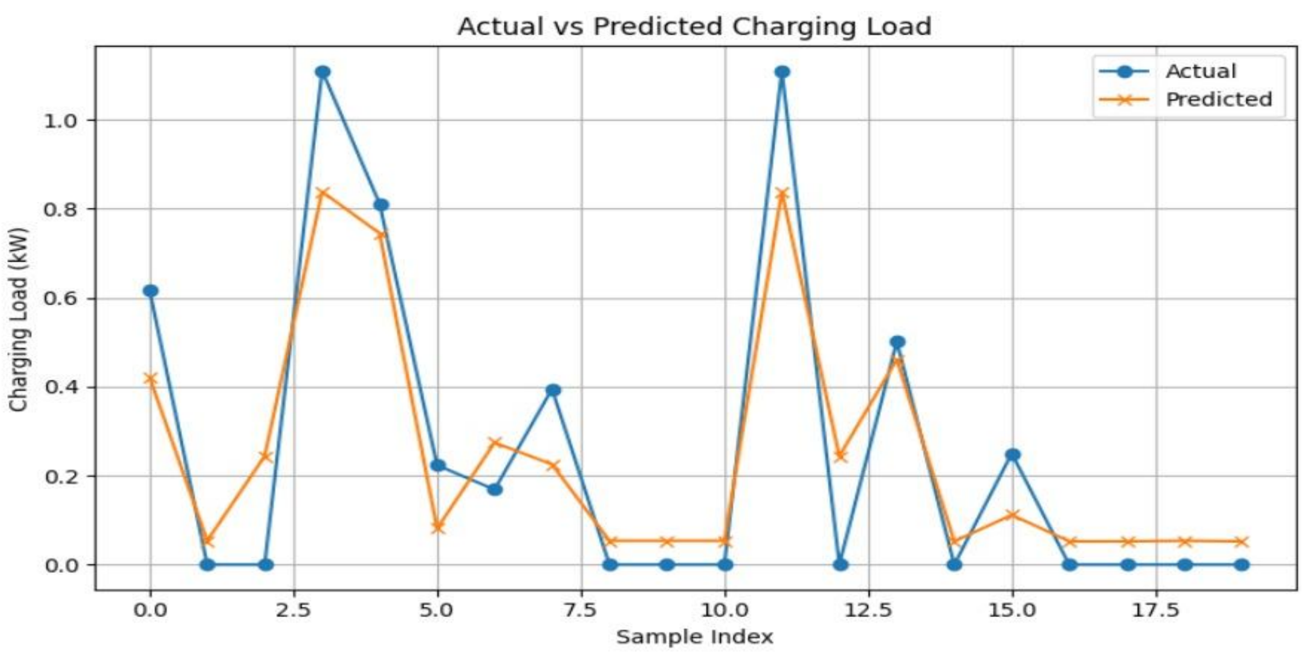


Fig 5. Actual vs predicted line graph

CHAPTER 5
PROJECT REQUIREMENT

Project Requirements

The EV Load Forecasting with Recommendation System was developed using various tools, libraries, and platforms that were essential for building, training, validating, and operationalizing the model. Below is a detailed description of the tools and technologies used.

Libraries Used:

1. **Python**

   Python served as the primary programming language for the project due to its versatility and extensive ecosystem for data processing, machine learning, and visualization. It was used for data preprocessing, model training (Random Forest, Gradient Boosting, XGBoost), feature engineering, and evaluation.

2. **Pandas**

   Pandas was used for structured data manipulation, including cleaning, filtering, grouping, and merging datasets. NumPy was employed for numerical computations required during feature engineering and statistical transformations.

3. **Scikit-learn**

   Scikit-learn provided the core machine learning utilities such as regression model APIs, train-test splitting, scaling, hyperparameter tuning (GridSearchCV), and evaluation metrics including MAE, RMSE, and R^2 score.

4. **XGBoost**

   XGBoost was selected as the final predictive model due to its superior performance in load forecasting. It enabled efficient training, regularization to prevent overfitting, and provided built-in feature importance rankings.

5. **Matplotlib**

   These visualization libraries were used to produce correlation heatmaps, distribution plots, boxplots, feature importance graphs, and “Actual vs Predicted” line charts for model performance analysis.
6. **PySpark**

   PySpark was implemented for large-scale distributed data processing and to consume streaming EV data from Kafka in the real-time pipeline.
7. **Apache Kafka**

   Kafka enabled real-time data ingestion from AWS S3 and other simulated data sources into the PySpark processing layer, supporting the streaming architecture.
8. **Joblib**

   Joblib was used to serialize and store the trained XGBoost model, allowing it to be reloaded for prediction without retraining.

Deployment:

The developed EV Load Forecasting and Recommendation System was deployed locally using the Flask micro web framework. Flask was selected for its lightweight architecture, ease of integration with Python-based machine learning workflows, and support for rapid prototyping.

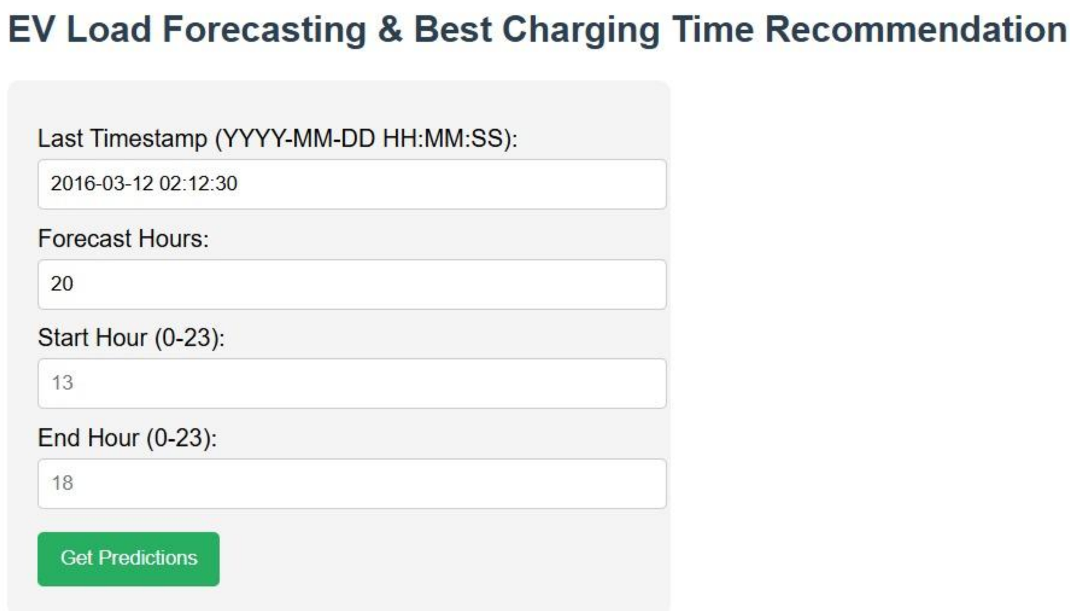
The deployment workflow is structured as follows:

1. **Model Preparation**
   - The forecasting model was trained offline using Gradient Boosting Regressor on the preprocessed EV dataset.
   - The final trained model was serialized and stored as a .pkl file using the joblib library.
   - This serialized model is loaded by the Flask application during runtime for performing predictions without retraining.
2. **Flask Application Structure**

   The application follows a modular structure:

3. csharp
4. CopyEdit
5. EV_Load_Forecast_Flask/
6. ├── app.py # Flask server logic
7. ├── requirements.txt # Project dependencies
8. ├── ev_load_forecast_and_recommendation.pkl # Trained model file
9. ├── templates/
   │ └── index.html # Frontend HTML UI
   └── static/ # Optional CSS, JS files

UI Interface:-



Result:-


```
{
  "all_predictions": [
    {
      "Hour": 3,
      "Predicted_Load_kW": 1.710301710719904,
      "Timestamp": "Sat, 12 Mar 2016 03:12:30 GMT"
    },
    {
      "Hour": 4,
      "Predicted_Load_kW": 1.710301710719904,
      "Timestamp": "Sat, 12 Mar 2016 04:12:30 GMT"
    },
    {
      "Hour": 5,
      "Predicted_Load_kW": 1.710301710719904,
      "Timestamp": "Sat, 12 Mar 2016 05:12:30 GMT"
    },
    {
      "Hour": 6,
      "Predicted_Load_kW": 1.710301710719904,
      "Timestamp": "Sat, 12 Mar 2016 06:12:30 GMT"
    },
    {
      "Hour": 7,
      "Predicted_Load_kW": 1.710301710719904,
      "Timestamp": "Sat, 12 Mar 2016 07:12:30 GMT"
    },
```


Recommendation System Output:-

```
  ],
  "best_times": [
    {
      "Predicted_Load_kW": 1.710301710719904,
      "Timestamp": "Sat, 12 Mar 2016 20:12:30 GMT"
    },
    {
      "Predicted_Load_kW": 1.710301710719904,
      "Timestamp": "Sat, 12 Mar 2016 21:12:30 GMT"
    },
    {
      "Predicted_Load_kW": 1.710301710719904,
      "Timestamp": "Sat, 12 Mar 2016 22:12:30 GMT"
    }
  ]
}
```

CHAPTER 6
FUTURE SCOPE

1.Incorporating More Data Sources:

Integrate additional data such as real-time vehicle telematics, user charging behavior patterns, dynamic grid signals, and renewable energy generation forecasts to improve model accuracy and responsiveness.

2.Real-Time Forecasting and Recommendations:

Enhance the system architecture to support real-time load prediction and instant adaptive recommendations for charging schedules, enabling immediate grid load management.

3.Advanced Algorithm Exploration:

Investigate and experiment with other state-of-the-art machine learning models and ensemble methods like CatBoost, LightGBM, and deep learning architectures to further optimize forecasting performance.

4.Integration with Smart Grid and Energy Management Systems:

Combine the forecasting and recommendation engine with existing utility infrastructure, smart meters, and demand response systems to enable automated, dynamic load balancing and energy cost optimization.

5.Explainable AI and Interpretability Tools:

Implement explainability techniques such as SHAP (Shapley Additive ex Planations) or LIME (Local Interpretable Model-agnostic Explanations) to increase transparency, helping stakeholders understand model decisions and build trust.

CHAPTER 7
CONCLUSION

This project successfully demonstrates how machine learning can be leveraged to accurately forecast electric vehicle (EV) charging load by utilizing a diverse range of operational, environmental, and grid-related features. Through detailed exploratory data analysis, including correlation checks and visualizations, we identified the key factors influencing EV charging demand, such as Load_per_EV, fleet size, charger power rating, time of day, and grid demand.

Multiple regression-based machine learning models, including Random Forest, Gradient Boosting, and XGBoost, were developed and compared. Among these, XGBoost emerged as the most effective model, achieving a high R^2 score of 0.96 along with low RMSE and MAE values, indicating excellent predictive performance.

In addition to forecasting, the integration of a recommendation engine enhanced the operational value of the system by generating actionable, data-driven charging schedules. These recommendations—such as shifting heavy-duty charging to off-peak periods and utilizing lower electricity price windows—help reduce peak load stress, optimize energy usage, and improve grid stability.

By combining high-accuracy forecasting with actionable scheduling strategies, this system enables utilities, EV fleet operators, and smart grid managers to make informed operational decisions. The project not only provides measurable technical and economic benefits but also contributes to sustainability goals by supporting the efficient integration of EV infrastructure into modern power grids.

CHAPTER 8

REFERENCES

1. Books / Research Papers

- J. Brownlee, *Machine Learning Mastery With Python: Understand Your Data, Create Accurate Models and Work Projects End-To-End*, Machine Learning Mastery, 2016.
- R. Kohavi, B. Becker, *Practical Machine Learning: A New Look at Data Mining*, Pearson Education, 2017.

2. Online Documentation and Articles

- "XGBoost Documentation." https://xgboost.readthedocs.io/
- "Random Forest Regressor - Scikit-learn Documentation." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- "Gradient Boosting Regression - Scikit-learn Documentation." https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting
- "Apache Kafka Documentation." https://kafka.apache.org/documentation/

3. Websites and Blogs

- "A Beginner's Guide to XGBoost in Python." https://machinelearningmastery.com/gentle-introduction-xgboost-python/
- "Understanding Random Forests." https://towardsdatascience.com/random-forest-explained-using-python-4da9f7aee497
- "How to Use Apache Kafka in Real-Time Data Pipelines." https://www.confluent.io/blog/kafka-fastest-messaging-system/
- "Time Series Forecasting with Machine Learning in Python." https://machinelearningmastery.com/time-series-forecasting- methods with-python-cheat-sheet/

4. Software/Tools Documentation

- Scikit-learn documentation for model building & evaluation: https://scikit-learn.org/
- XGBoost library and API details: https://xgboost.readthedocs.io/
- Apache Kafka documentation: https://kafka.apache.org/
- PySpark documentation: https://spark.apache.org/docs/latest/api/python/
- AWS EC2 documentation for server deployment: https://aws.amazon.com/ec2/