Load Prediction Model Using Gradient Boosting Regressor

Author: Hiran Prajaubphon (劉海洋) 11109338A

This report discusses the development and evaluation of a machine learning model aimed at predicting electricity load. The model utilizes the Gradient Boosting Regressor to forecast future load based on historical load data and weather-related features, such as maximum, minimum, and average temperature, along with the day of the week. Key features used in the model include two lagged features: n-1 (load from the previous day) and n-7 (load from the same day of the previous week). The dataset is split into training and testing sets with 670 samples for training and 330 samples for testing to assess the model's performance.

1. Data Preparation:

- The dataset was loaded from the file HW#6.csv, containing information about the year, month, day, week day, load, and temperature.
- The target variable, Load, was converted to numeric values. Any non-numeric entries were coerced into NaN, which were then dropped to ensure clean data.
- Two lagged features were created:

  Load_n-1: The load from the previous day (lag of 1).

  Load_n-7: The load from the same day in the previous week (lag of 7).
- Rows with missing values due to the lag operations were also removed to ensure that all data used in training and testing were valid.

2. Feature Selection:

The features used to train the model were selected as follows:

- **Input Features**:

  Load_n-1 (Load from the previous day)

  Load_n-7 (Load from the same day last week)

  Week Day (Day of the week, 1-7)

  Max Temp (Maximum temperature of the day)

  Min Temp (Minimum temperature of the day)

  Avg Temp (Average temperature of the day)
- **Target Variable**:

Load (Electricity load to predict)

3. Data Splitting:

- The dataset was split into **training** and **testing** sets with the following distribution:

  Training Set: 670 samples (Used to train the model)

  Test Set: 330 samples (Used to evaluate the model's performance)

- The split was performed using the train_test_split function from scikit-learn, ensuring randomness and a proper evaluation of the model's generalization ability.

4. Model Selection:

The model chosen for predicting Load was the **Gradient Boosting Regressor**. This model was selected due to its efficiency in handling various types of features and its ability to capture complex non-linear relationships. The model was configured with the following hyperparameters:

- learning_rate=0.1: Controls the contribution of each tree.
- max_depth=3: The maximum depth of each individual tree.
- n_estimators=200: The number of boosting stages (trees).
- subsample=0.8: The fraction of samples used for fitting each base learner.
- random_state=42: Ensures reproducibility of results.

5. Model Training:

- The Gradient Boosting Regressor was trained using the training data (670 samples), with the input features (Load_n-1, Load_n-7, Week Day, Max Temp, Min Temp, and Avg Temp) to predict the target variable Load.

6. Model Evaluation:

- The model was evaluated on the test set (330 samples), using the following performance metrics:

  Mean Absolute Error (MAE): This measures the average magnitude of the errors in predictions, without considering their direction.

  Root Mean Squared Error (RMSE): This is another measure of prediction accuracy, where larger errors are penalized more heavily.

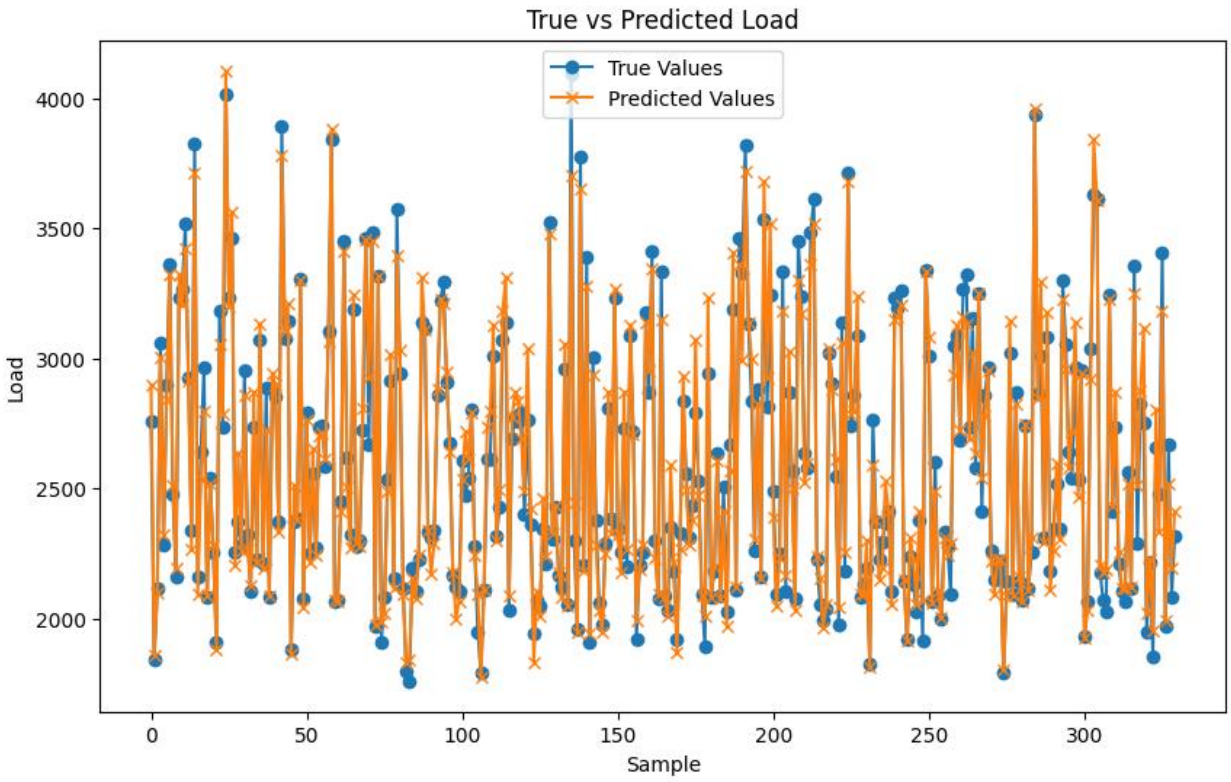
  R-squared (R^2 Score): This indicates how well the model explains the variance in the target variable. A higher R^2 means better predictive accuracy.

7. Results:

After training and testing the model, the following metrics were calculated:

- **Mean Absolute Error (MAE):** 70.64
- **Root Mean Squared Error (RMSE):** 96.60
- **R^2 Score (Test Accuracy):** 0.96

These results demonstrate that the model is highly effective at predicting the electricity load, with a high R^2 score indicating that a significant portion of the variance in load can be explained by the model.

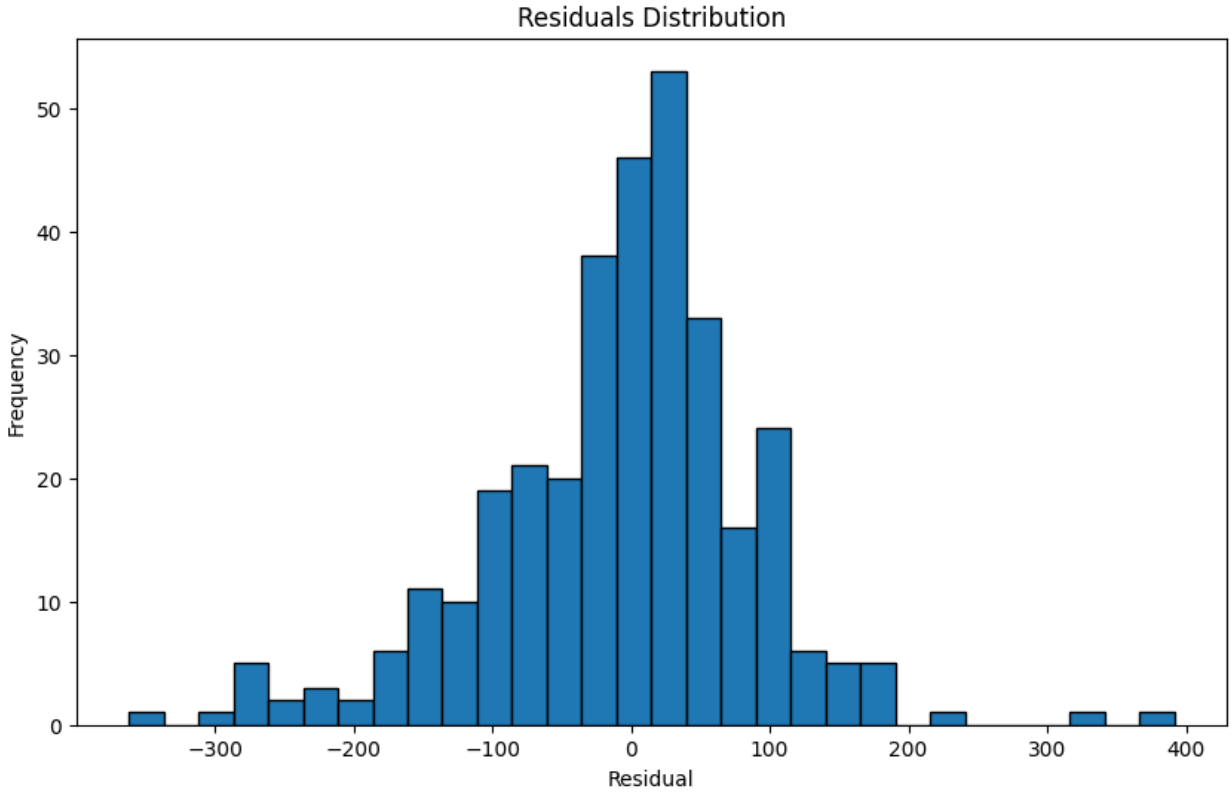


The graph suggests that the model performs well in predicting the load values. However, there is room for improvement, particularly in instances where the predictions deviate from the actual values. Further model refinement, such as adjusting parameters or incorporating additional features, could enhance its accuracy.

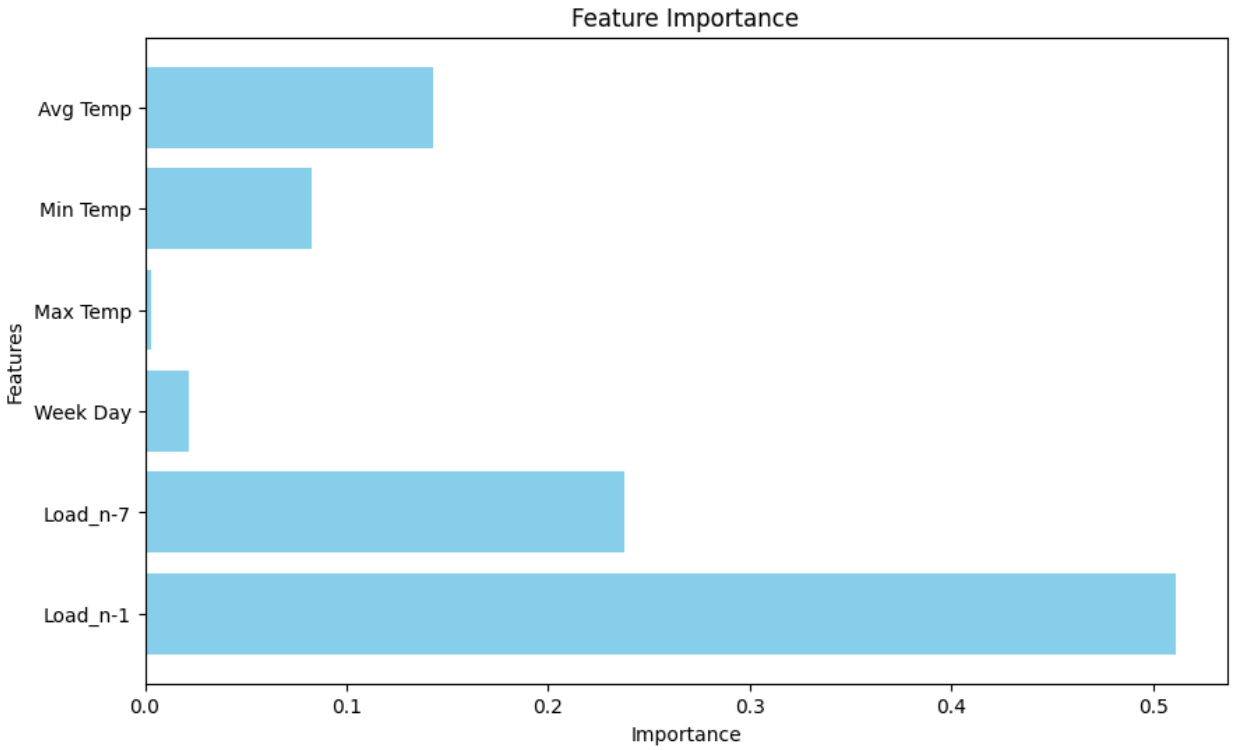
Close Alignment: The blue line (true values) and the orange line (predicted values) closely follow each other, suggesting that the model is generally accurate in predicting the load values.

Occasional Discrepancies: There are instances where the predicted values deviate significantly from the actual values, indicating that the model may struggle with certain data points or scenarios.

Similar Patterns: Both lines exhibit similar patterns, demonstrating that the model has effectively captured the underlying trends and relationships within the data.



The graph shows the distribution of residuals. Ideally, residuals should be normally distributed with zero mean. The graph shows a roughly bell-shaped distribution, suggesting that the model's errors are random and normally distributed, which is generally good. However, there are some outliers that might require further investigation.



The graph shows the importance of different features in predicting load.

- **Load_n-1** is the most important, indicating that past load values strongly influence the current load.
- **Load_n-7** is also significant, suggesting weekly patterns in load.
- **Avg Temp** has a notable impact, with temperature affecting energy demand.
- **Min Temp** is somewhat important, while **Max Temp** is the least influential.
- **Week Day** has a lower importance, indicating daily patterns are less significant.

In short, historical load data and temperature are key factors in predicting load.

8. Conclusion:

The Gradient Boosting Regressor effectively predicted electricity load based on the historical load and weather features. The inclusion of lag features (Load_n-1 and Load_n-7) allowed the model to account for temporal patterns in the data. The model's performance, as indicated by the evaluation metrics (MAE, RMSE, and R^2 score), suggests that it can be used reliably for load prediction tasks.

Overall, the model provides a solid foundation for load prediction and can be extended to incorporate more complex features or data sources for improved accuracy.

9. Code

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
from google.colab import drive
drive.mount('/content/drive')
data = pd.read_csv('/content/drive/MyDrive/AI/Hw6/HW#6.csv')
names = ['Year','Month','Day','Week Day','Load','Max Temp','Min
Temp','Avg Temp']
data.head()
# Load data
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/AI/Hw6/HW#6.csv',
header=None, names=['Year', 'Month', 'Day', 'Week Day', 'Load',
'Max Temp', 'Min Temp', 'Avg Temp'])
data['Load'] = pd.to_numeric(data['Load'], errors='coerce')

data = data.dropna(subset=['Load'])
data['Load_n-1'] = data['Load'].shift(1)
data['Load_n-7'] = data['Load'].shift(7)
data = data.dropna()

features = ['Load_n-1', 'Load_n-7', 'Week Day', 'Max Temp', 'Min
Temp', 'Avg Temp']
X = data[features]
y = data['Load']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=330, random_state=42)

from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor(
    learning_rate=0.1,
    max_depth=3,
    n_estimators=200,
    subsample=0.8,
    random_state=42
)
model.fit(X_train, y_train)
```

```
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R2 Score (Test Accuracy): {r2:.2f}")


plt.figure(figsize=(10, 6))
plt.plot(y_test.values, label="True Values", marker='o')
plt.plot(y_pred, label="Predicted Values", marker='x')
plt.legend()
plt.title("True vs Predicted Load")
plt.xlabel("Sample")
plt.ylabel("Load")
plt.show()

residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
plt.hist(residuals, bins=30, edgecolor='k')
plt.title("Residuals Distribution")
plt.xlabel("Residual")
plt.ylabel("Frequency")
plt.show()

feature_importances = model.feature_importances_
plt.figure(figsize=(10, 6))
plt.barh(features, feature_importances, color='skyblue')
plt.title("Feature Importance")
plt.xlabel("Importance")
plt.ylabel("Features")
plt.show()

comparison_df = pd.DataFrame({
    'Actual': y_test.values,
    'Predicted': y_pred
})
print(comparison_df)
```