Fuel Consumption Prediction Models

Using Machine Learning and Mathematical Methods

Presented by Tejas Vilas Kondhalkar & Bulbul Kumari
University of Delhi – ECE Department
Guide: Dr. Juhi Jain

Ship Fuel Consumption Prediction Models Based on Machine Learning and Mathematical Methods

The goal of this research is to enhance the accuracy of ship fuel consumption predictions using machine learning (ML) and mathematical modeling. The framework incorporates both white-box (physics-based) and black-box (ML-based) models to provide reliable predictions while ensuring interpretability and robustness.

Project Objective

Predict fuel consumption of ships

Accurate estimation to optimise fuel use and costs

Compare machine learning and mathematical methods

Evaluate efficiency, accuracy, and interpretability

Reduce fuel expenses & CO_2 emissions

Contribute to sustainable shipping practices

Project Overview

Purpose

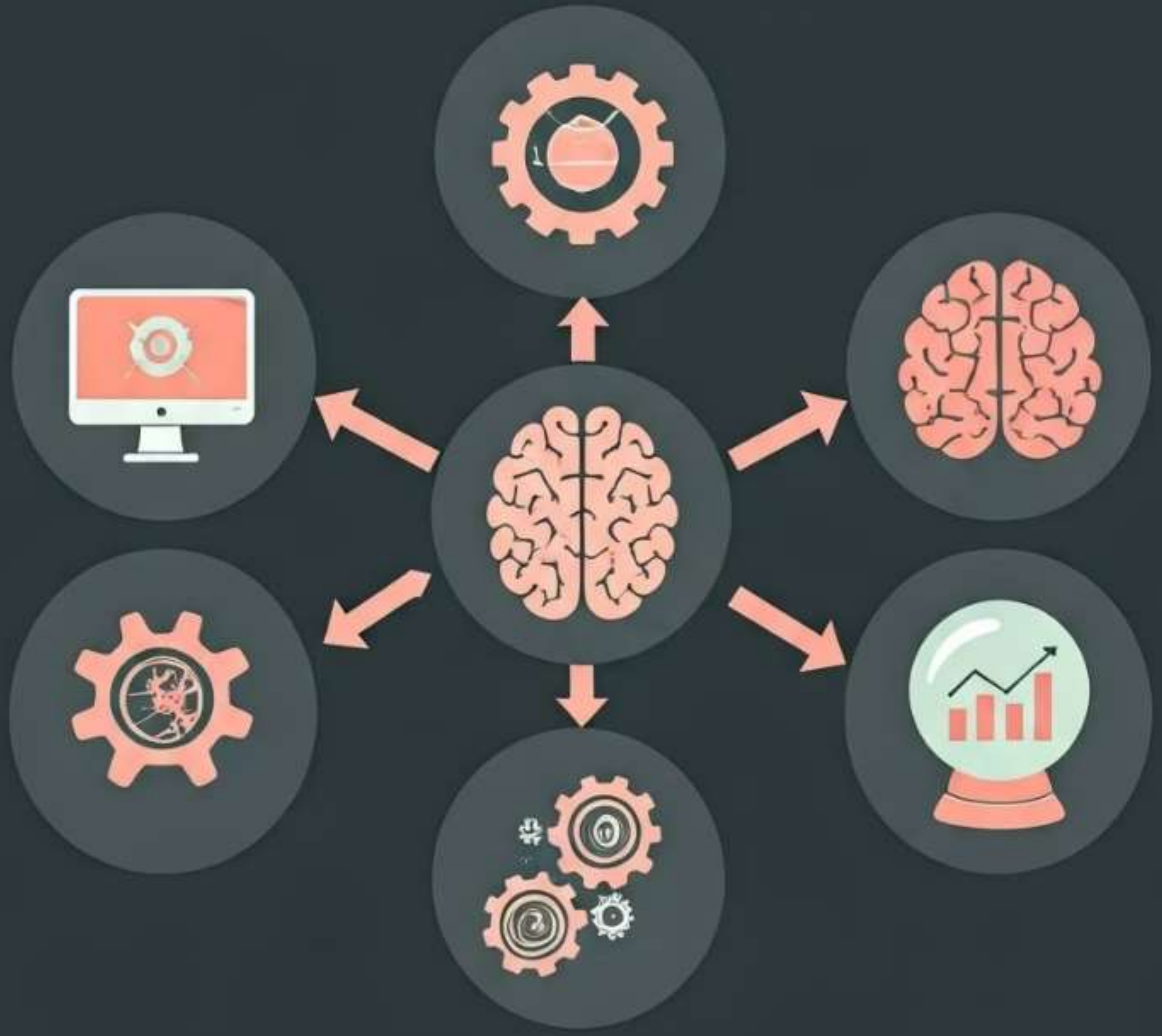
Predict ship fuel efficiency and CO_2 emissions using machine learning.

Dataset

ship_fuel_efficiency.csv with 1,440 entries and 10 features.(From Kaggle)

Methods

White Box and Black Box - Random Forest and XGBoost models.



Analysis Workflow

Data Loading & Inspection

Load dataset and examine basic statistics.

Data Cleaning (Kwon Method)

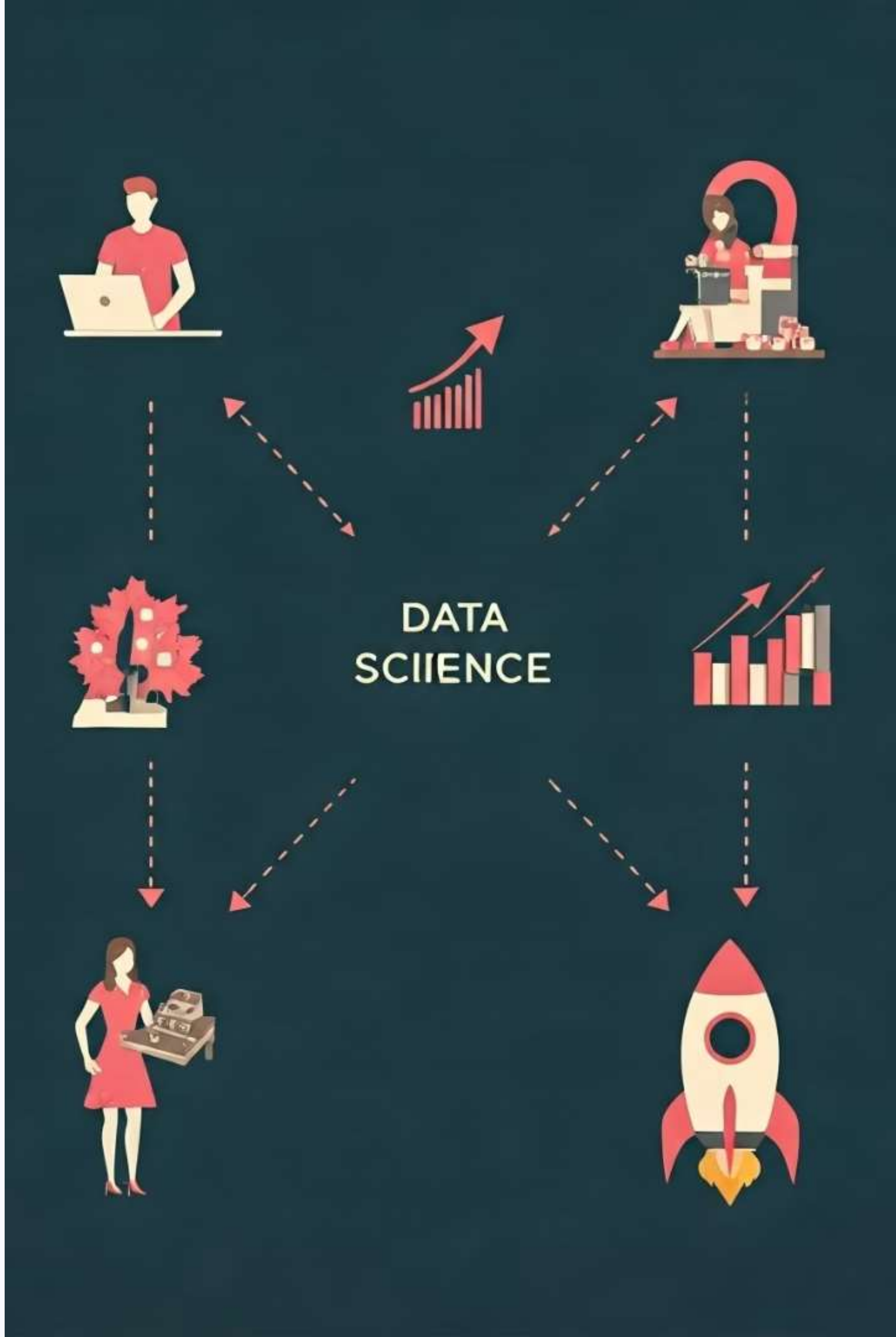
Remove outliers using Z-scores.

Feature Engineering

Create new variables to improve model performance.

Model Training & Evaluation

Train models and assess their accuracy.



Data Preprocessing Steps

1 Outlier Removal

Applied Z-score method with threshold |z| > 3

2 Handling Missing Values

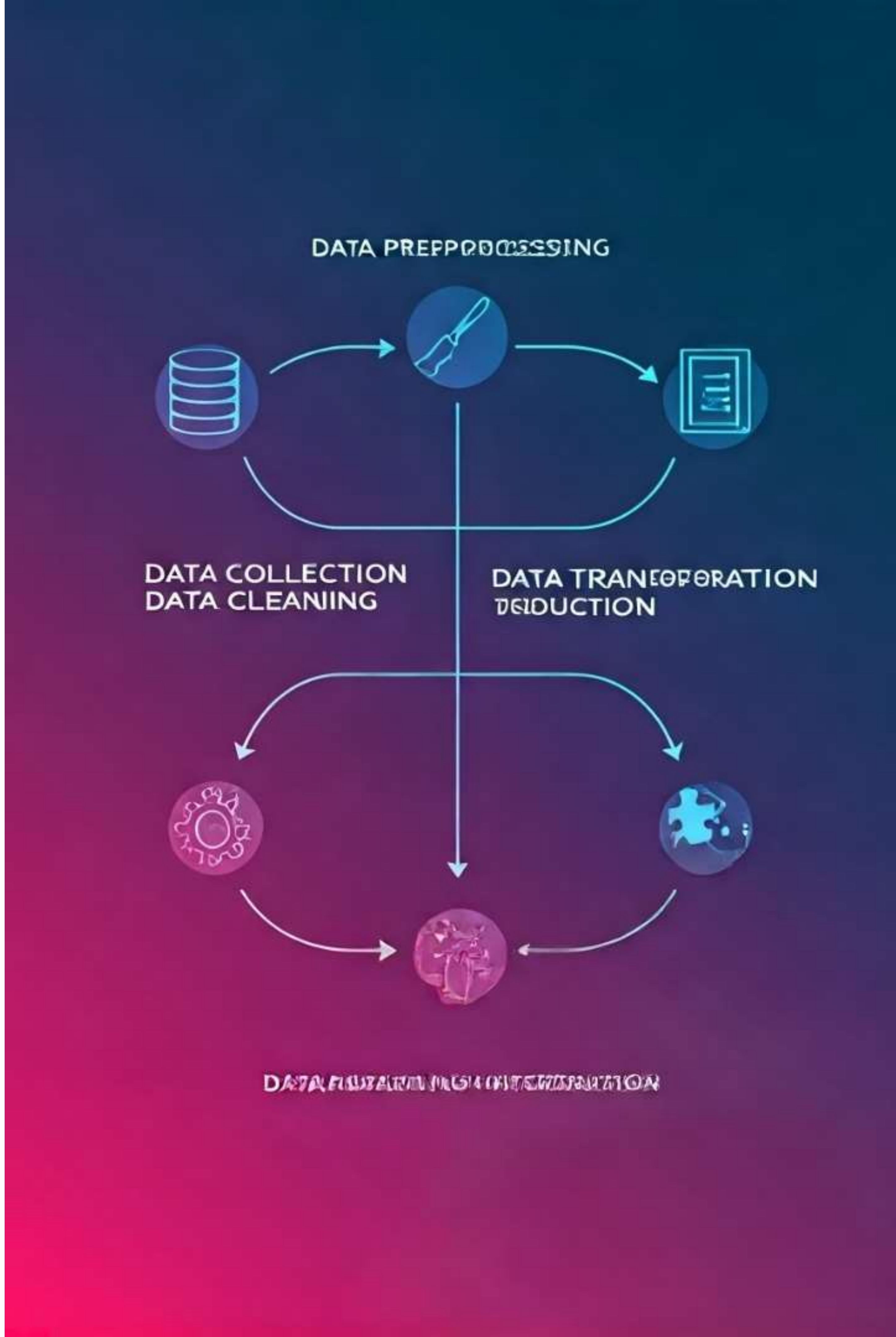
Imputation or removal to ensure data quality

3 Feature Engineering

Calculated speed, displacement, power, and draft

4 Encoding & Scaling

MinMaxScaler used; data split 80% train, 20% test



Data Loading & Initial Inspection

Dataset used : [Kaggle - Ship Fuel Consumption and CO2 Emissions Analysis]

```
df = pd.read_csv('ship_fuel_efficiency.csv')
print(df.info())
print(df.describe())
```

- We load the dataset using Pandas and examine data types, non-null counts, and summary statistics.
- There are 10 columns in the dataset namely – ship_id , ship_type, route_id, month, distance, fuel_type ,fuel_consumption , CO2_emissions , weather_conditions , engine_efficiency.
- Key Features :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1440 entries, 0 to 1439
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   ship_id            1440 non-null   object 
 1   ship_type          1440 non-null   object 
 2   route_id           1440 non-null   object 
 3   month              1440 non-null   object 
 4   distance           1440 non-null   float64
 5   fuel_type          1440 non-null   object 
 6   fuel_consumption   1440 non-null   float64
 7   CO2_emissions      1440 non-null   float64
 8   weather_conditions 1440 non-null   object 
 9   engine_efficiency  1440 non-null   float64
dtypes: float64(4), object(6)
memory usage: 112.6+ KB
None
          distance  fuel_consumption  CO2_emissions  engine_efficiency
count  1440.000000       1440.000000    1440.000000        1440.000000
mean    151.753354       4844.246535   13365.454882          82.582924
std     108.472230       4892.352813   13567.650118           7.158289
min      20.080000        237.880000     615.680000          70.010000
25%      79.002500       1837.962500    4991.485000          76.255000
50%     123.465000       3060.880000    8423.255000          82.775000
75%     180.780000       4870.675000   13447.120000          88.862500
max     498.550000      24648.520000   71871.210000          94.980000
```

The output reveals our dataset structure and initial statistical properties.

Data Cleaning (Kwon Method)

Calculate Z-scores

Standardize numeric values to identify outliers.

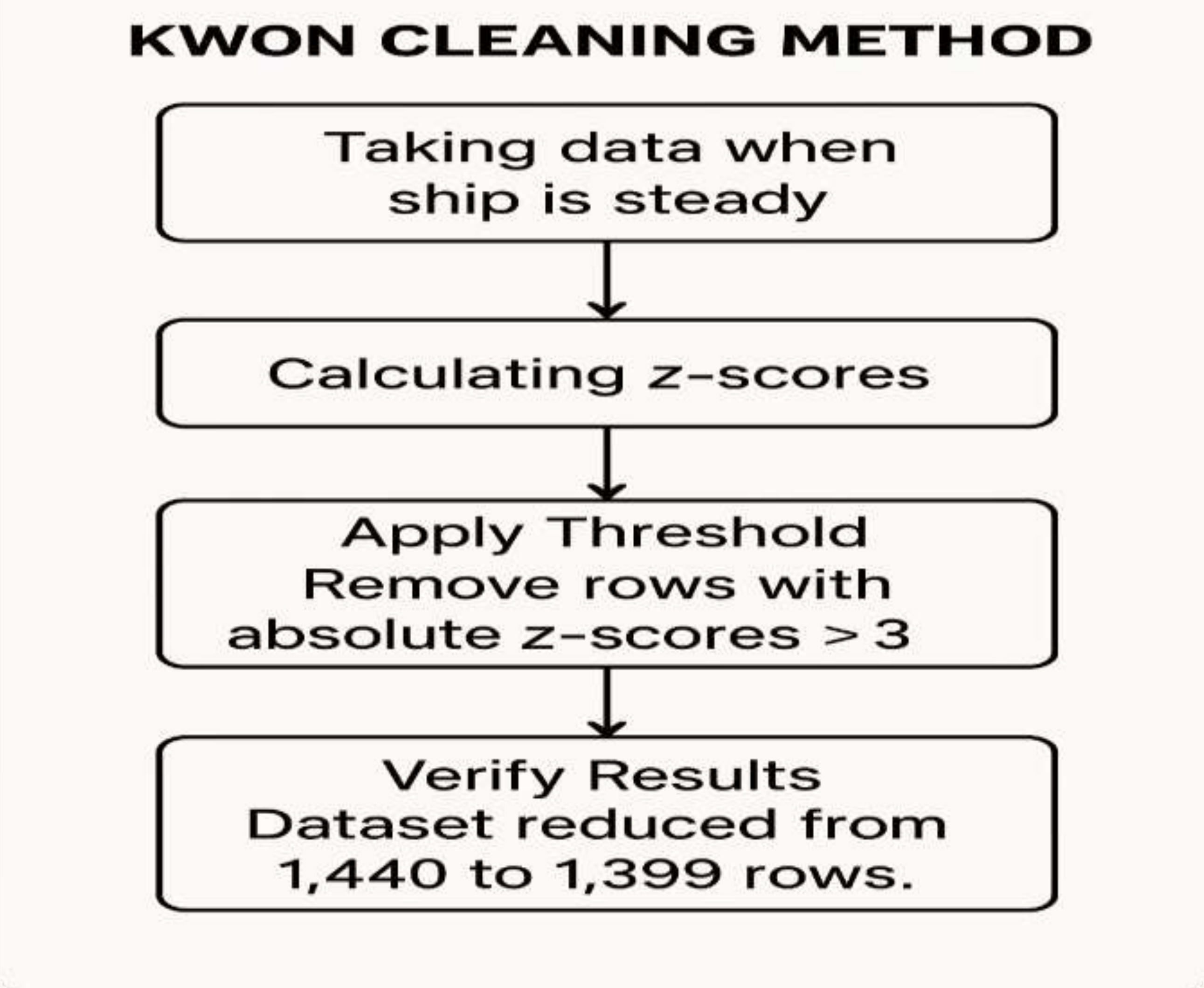
Apply Threshold

Remove rows with absolute Z-scores > 3.

Verify Results

Dataset reduced from 1,440 to 1,399 rows.

```python
# Applying Kwon Cleaning Method with numeric conversion
def kwon_cleaning_method(df, threshold=3):
    df_numeric = df.select_dtypes(include=[np.number])  # Select only numeric columns
    z_scores = (df_numeric - df_numeric.mean()) / df_numeric.std()  # Compute Z-scores
    cleaned_df = df[(z_scores.abs() <= threshold).all(axis=1)]
    return cleaned_df


df = kwon_cleaning_method(df)


# Drop NaN values
df = df.dropna()
```



Feature Engineering

Speed Calculation

df['speed'] = df['distance'] / 10

Distance divided by a constant (hypothetical time).

Displacement Derivation

df['displacement'] = df['CO2_emissions'] * 0.1

Derived from CO_2 emissions.

Power Computation

df['power'] = df['engine_efficiency'] * df['fuel_consumption']

Engine efficiency multiplied by fuel consumption.

	engine_efficiency	speed	displacement	power	draft
0	92.14	13.226	1062.576	348268.0078	53.12880
1	92.98	12.852	1277.973	414824.6912	63.89865
2	87.61	6.730	535.301	163631.8253	26.76505
3	87.42	7.168	650.652	209240.6442	32.53260
4	85.61	13.432	1161.703	365314.1359	58.08515
...	...	...	...	...	...
1435	75.88	6.384	485.228	123976.5380	24.26140
1436	78.00	6.143	357.113	98551.4400	17.85565
1437	79.67	19.309	1226.713	371392.0621	61.33565
1438	92.87	16.650	1229.771	399155.2600	61.48855
1439	90.82	12.766	1064.190	322402.8262	53.20950

Machine Learning Models Employed

White Box

Baseline model; interpretable results

Random Forest

Ensemble technique; handles complex nonlinearities

XGBoost

Boosting model; highest accuracy and robustness

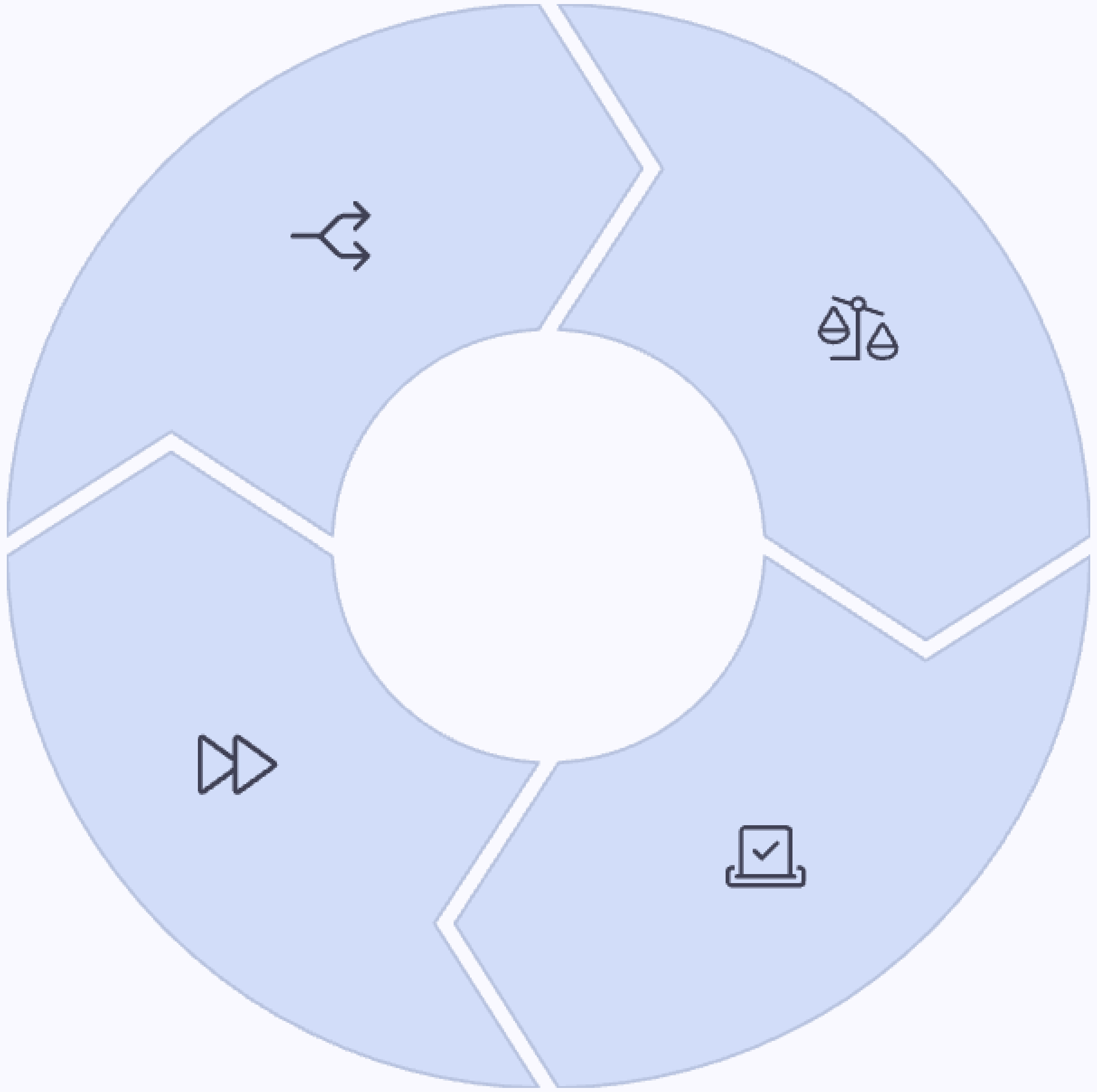
Mathematical Model: White Box Formula

This model standardises input parameters to estimate fuel usage.

Highly interpretable and helpful for initial analysis.

```python
# White-box model: Fuel Consumption Formula-Based Model (from research paper)
def fuel_consumption_model(X):
    speed, draft, displacement, power = X.T
    return (0.6 * speed**1.2 + 0.25 * draft**0.8 +
            0.15 * displacement**0.5 + 0.5 * power**1.1)


y_pred_whitebox = fuel_consumption_model(X_test)
```

Train-Test Split & Scaling



Split Data

80% training, 20% testing

Normalize Features

MinMaxScaler to range [0,1]

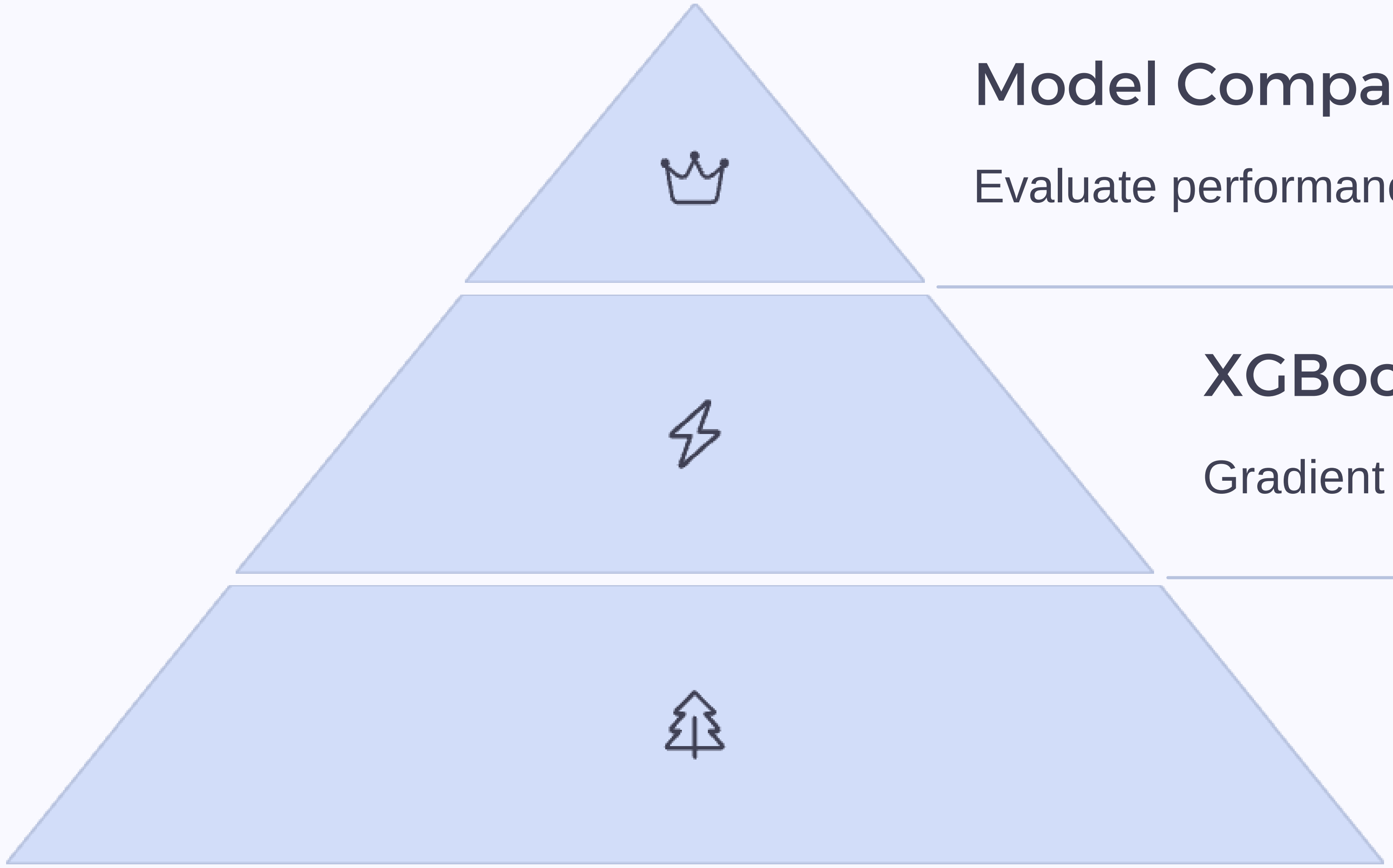
Verify Distribution

Examine target variable

Prepare for Modeling

Data ready for algorithms

Model Training



Model Comparison

Evaluate performance metrics

XGBoost

Gradient boosting with regularization

Random Forest

Ensemble of decision trees

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \in [0,+\infty)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \in [0,+\infty)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \in [0,+\infty)$$

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \in [0,+\infty)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \in [0,1]$$

Performance Metrics

1. MAE

   Mean Absolute Error; average of absolute differences

2. MSE

   Mean Squared Error; average squared differences

3. RMSE

   Root Mean Squared Error; square root of MSE

4. R^2

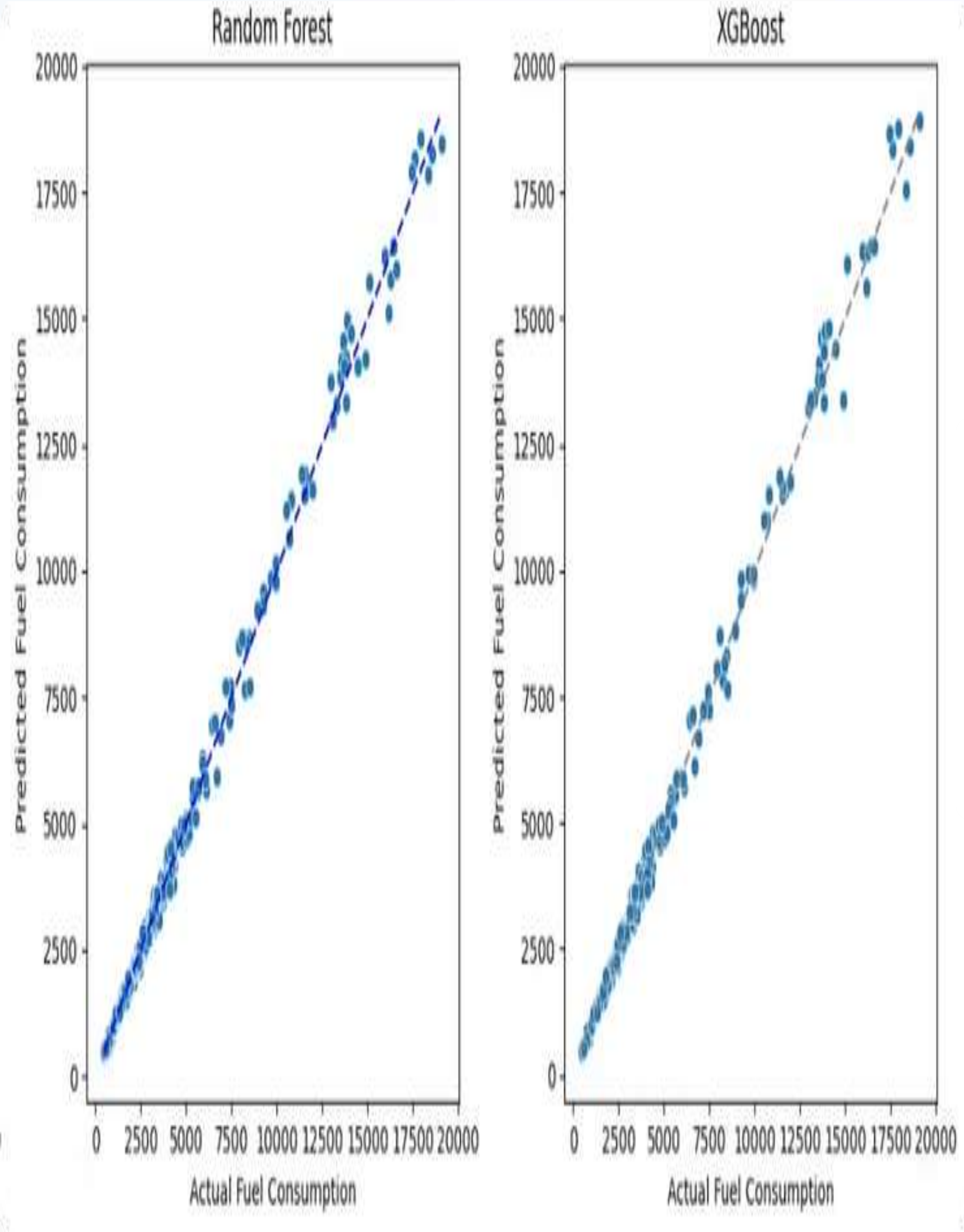
   Coefficient of Determination; variance explained by model

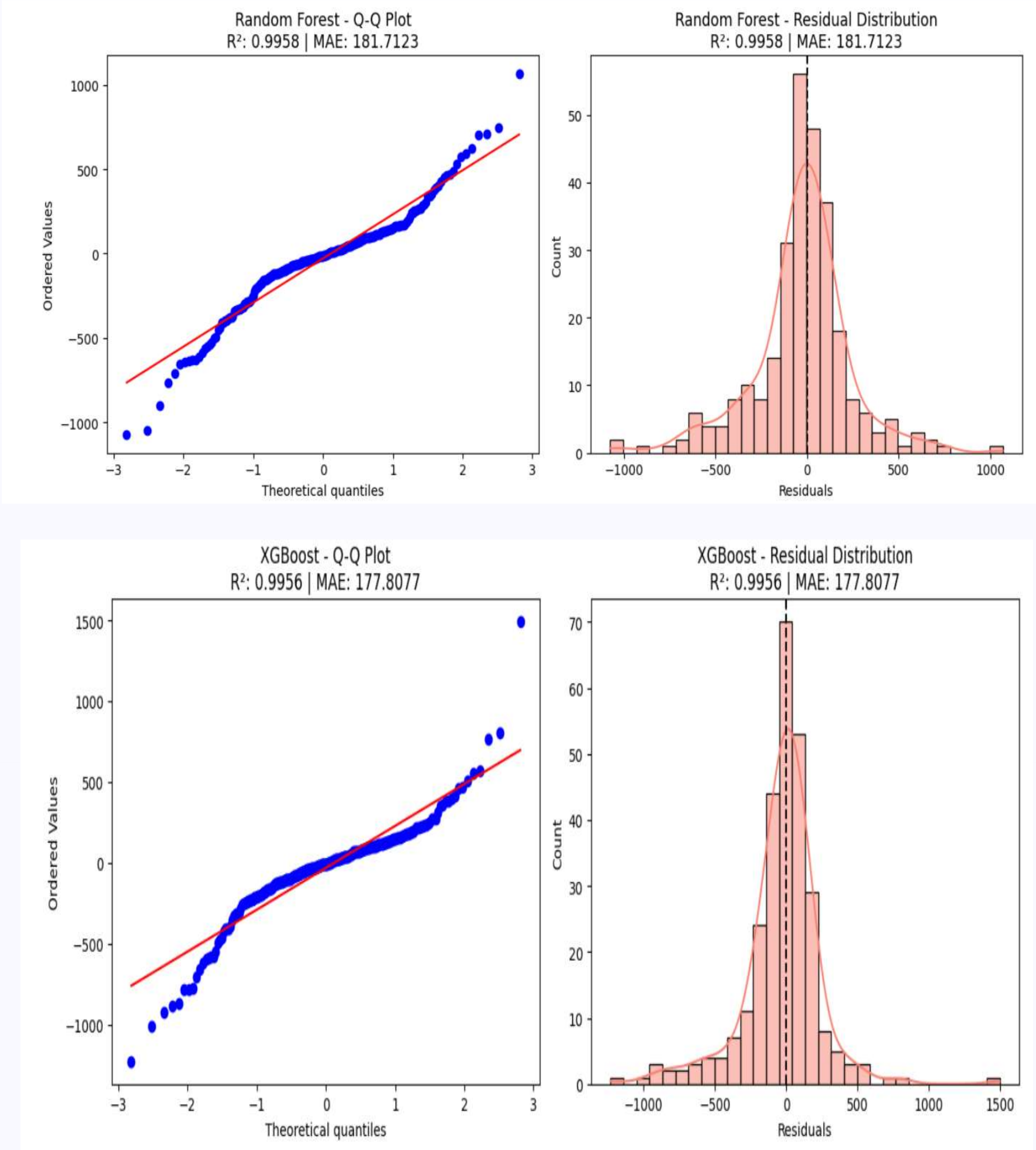
```python
# Evaluating the models
def evaluate_model(y_test, y_pred, model_name):
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
    print(f'\nModel: {model_name}')
    print(f'MAE: {mae}')
    print(f'MSE: {mse}')
    print(f'RMSE: {rmse}')
    print(f'R^2 Score: {r2}')

evaluate_model(y_test, y_pred_whitebox, "Fuel Consumption Formula (White-box)")
evaluate_model(y_test, y_pred_rf, "Random Forest (Black-box)")
evaluate_model(y_test, y_pred_xgb, "XGBoost (Black-box)")
```

Model Performance Summary

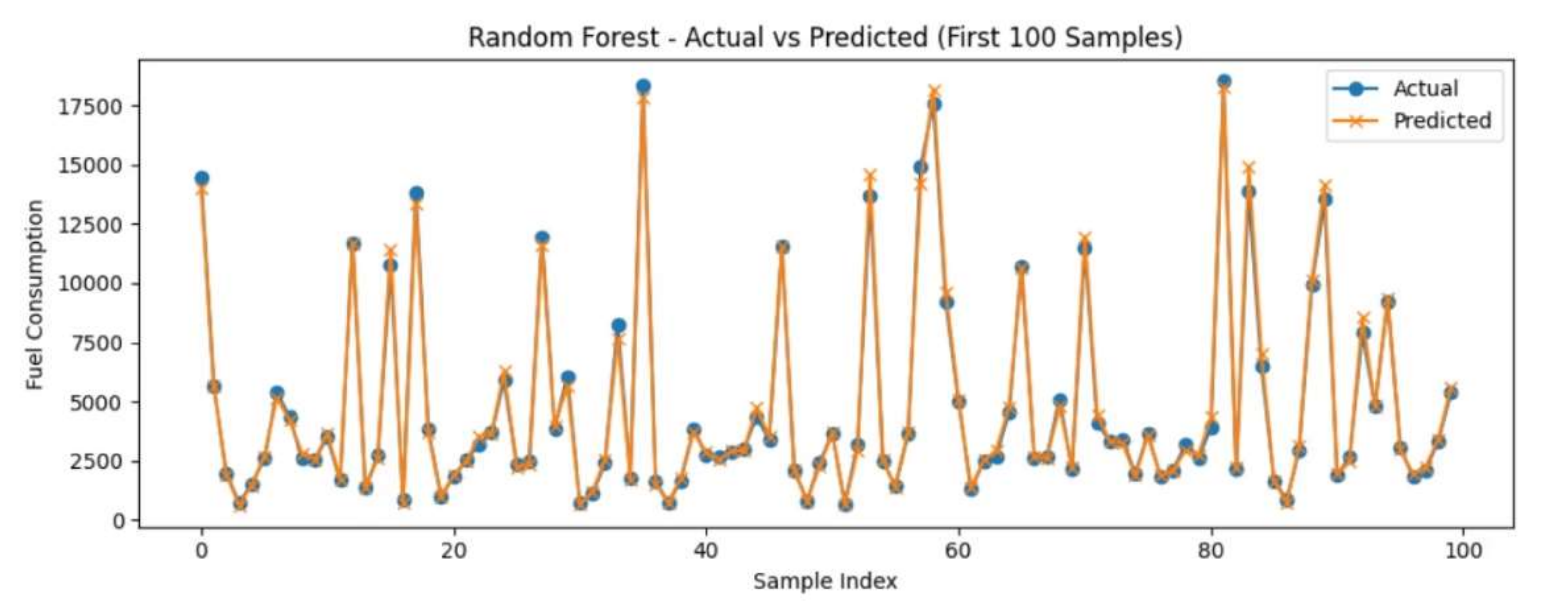
Model	MAE	MSE	RMSE	R^2
White BOX	4576.28	38214004.79	6181.74	-1.21
Random Forest	181.71	73171.67	270.50	0.9955
XGBoost	117.80	76177.09	276.00	0.9957

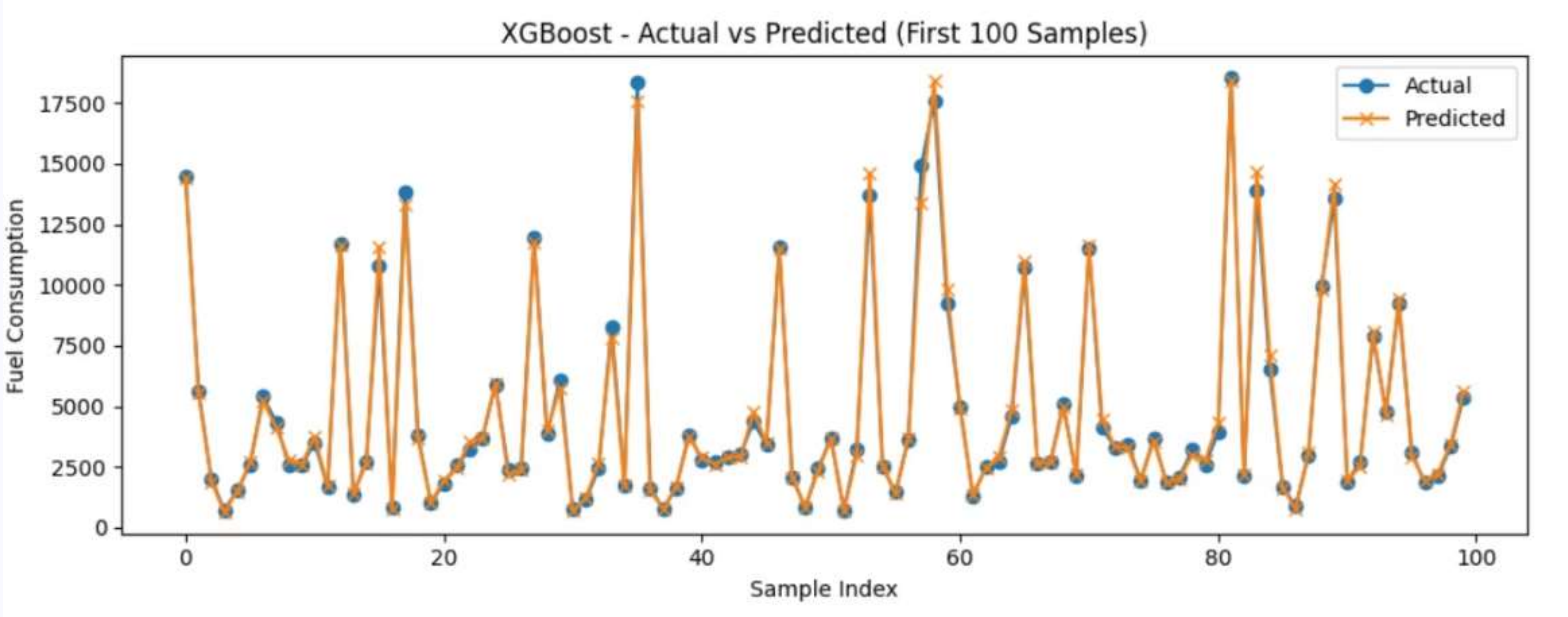
XGBoost outperforms all models in every key metric.


Random Forest
XGBoost
Predicted Fuel Consumption
Actual Fuel Consumption
0
2500
5000
7500
10000
12500
15000
17500
20000

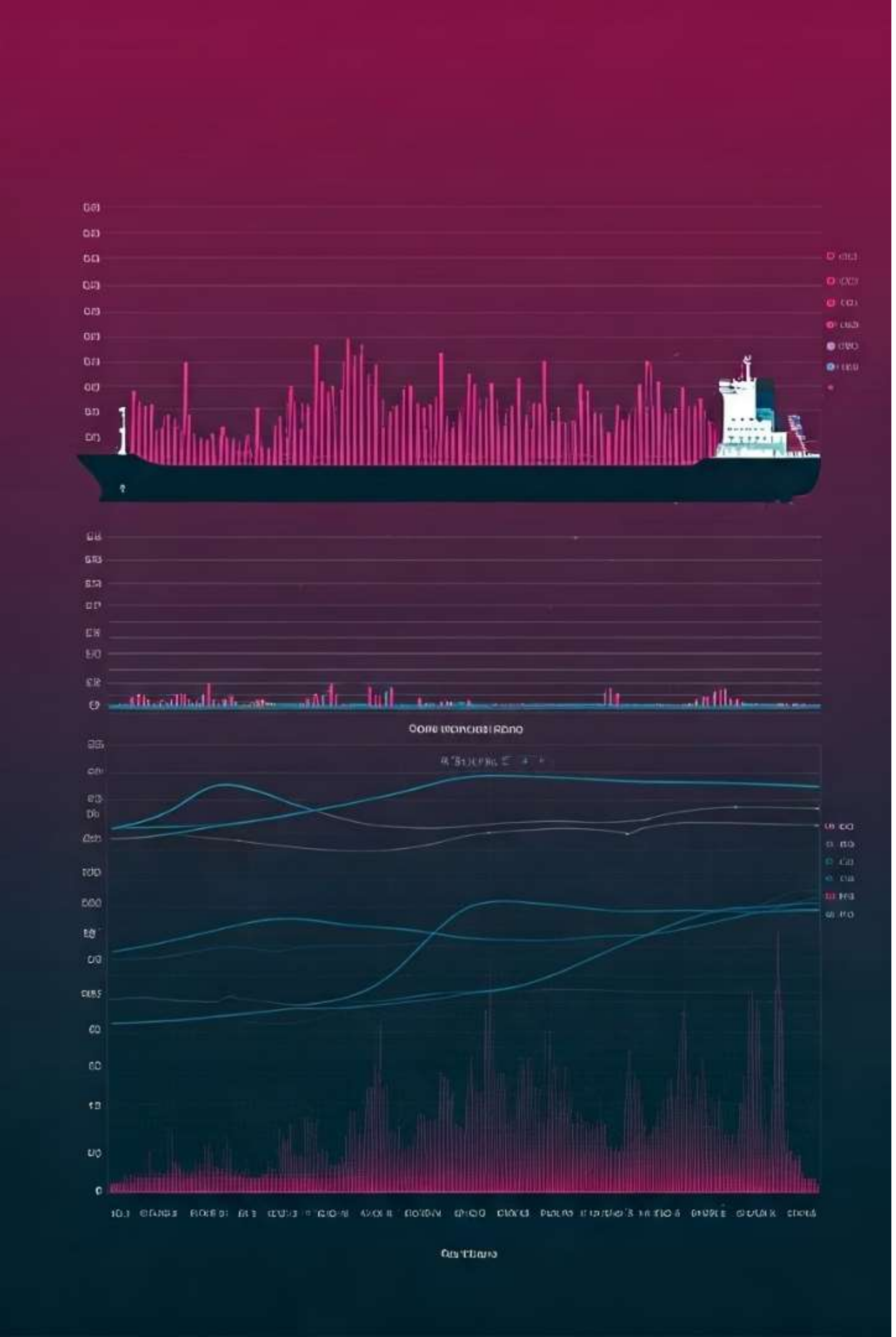

Random Forest - Q-Q Plot
R²: 0.9958 | MAE: 181.7123
Random Forest - Residual Distribution
R²: 0.9958 | MAE: 181.7123
XGBoost - Q-Q Plot
R²: 0.9956 | MAE: 177.8077
XGBoost - Residual Distribution
R²: 0.9956 | MAE: 177.8077
Ordered Values
Theoretical quantiles
Count
Residuals

Key Visualizations





These visualizations reveal feature relationships, model insights, and error distribution patterns.



Visual Data Analysis

1

Histogram

Shows fuel consumption distribution

2

Boxplot

Comparison across various ship types

3

Heatmap

Correlations between features and target

4

Scatter Plots

Feature-wise relationships with fuel consumption

Machine Learning Model Conclusions

ML models clearly outperformed traditional mathematical approaches.

XGBoost emerged as the top-performing algorithm.

Results have strong real-world and environmental relevance.

Modifications Overview

New Dataset

Utilized recent, comprehensive data for accuracy

Formula Adjustments

Enhanced Kwon's formula for improved predictive power

Feature Engineering

Introduced additional variables to strengthen models

Advanced Visualizations

Implemented dynamic graphs for interpretation

Key Learnings

ML Application

Practical insight into deploying ML models

Data Engineering

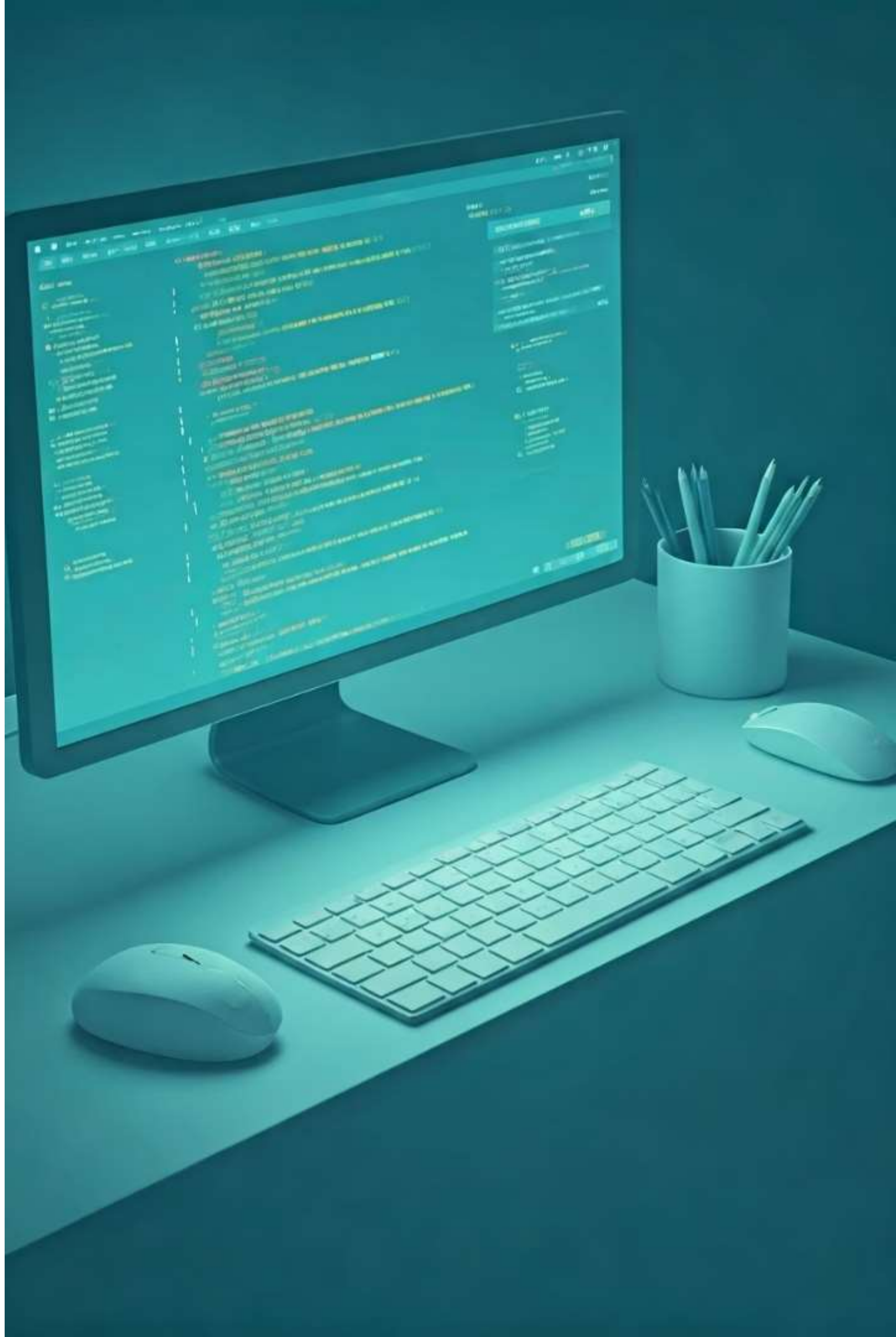
Critical data cleaning and feature creation

Visualization Skills

Effective model communication through graphs

Algorithm Expertise

Hands-on experience with RF and XGBoost



Future Directions

1. **Time-Series Forecasting**
2. **CO2 Emission Prediction**
3. **Real-Time Data Integration**
4. **Deep Learning (LSTM)**
5. **App and Dashboard**

   Enhanced data accessibility and user interaction

Thank You

Acknowledgments

Grateful for teamwork and support