**ECP\_US\_Model Documentation**

**1. Introduction**

This document provides an overview of the ECP\_US\_Model, detailing the steps taken for exploratory data analysis (EDA), data preprocessing, model training, and evaluation. The model is implemented using Python with libraries such as Pandas, NumPy, Scikit-learn, Matplotlib, and Seaborn.

**2. Data Loading**

The dataset is read from a CSV file:

df = pd.read\_csv("energy\_consumption\_us.csv")

The first few rows of the dataset are displayed to verify successful loading:

print("Dataset Shape:\n", df.shape)

df.head()

**3. Exploratory Data Analysis (EDA)**

EDA is performed to understand the dataset structure:

**Dataset Overview**

Checking the shape:

print("Dataset Shape:\n", df.shape)

Displaying column data types:

print("Data Types:\n", df.dtypes)

Statistical summary:

print("Summary Statistics:\n", df.describe())

**Missing Values Analysis**

print("Missing Values:\n", df.isnull().sum())

Handling missing values:

df.fillna(df.mean(), inplace=True) # Replacing NaN with mean

**Visualization**

Distribution of numerical features:

for column in df.select\_dtypes(include=["number"]).columns:

sns.histplot(df[column], kde=True)

plt.title(f"Distribution of {column}")

plt.show()

Correlation heatmap:

plt.figure(figsize=(10, 8))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)

plt.title("Feature Correlation Heatmap")

plt.show()

**4. Data Preprocessing**

Preprocessing steps include handling missing values, encoding categorical variables, and scaling numerical features:

* **Encoding Categorical Variables**

encoder = OneHotEncoder(sparse=False)

categorical\_columns = ['categorical\_column']

encoded\_features = encoder.fit\_transform(df[categorical\_columns])

* **Feature Scaling**

scaler = StandardScaler()

numerical\_columns = ['numerical\_column']

df[numerical\_columns] = scaler.fit\_transform(df[numerical\_columns])

**5. Model Training**

The dataset is split into training and testing sets, and a linear regression model is trained:

X = df.drop(columns=['target\_column'])

y = df['target\_column']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression()

model.fit(X\_train, y\_train)

**6. Model Evaluation**

The model's performance is evaluated using various metrics:

y\_pred = model.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean Absolute Error (MAE):", mae)

print("Mean Squared Error (MSE):", mse)

print("R-squared Score:", r2)

* **Residual Plot**

residuals = y\_test - y\_pred

sns.histplot(residuals, kde=True)

plt.title("Residual Distribution")

plt.show()

* **Scatter Plot of Actual vs. Predicted Values**

plt.scatter(y\_test, y\_pred)

plt.xlabel("Actual Values")

plt.ylabel("Predicted Values")

plt.title("Actual vs. Predicted Values")

plt.show()

**7. Model Optimization**

To improve model performance, the following techniques can be applied:

* **Hyperparameter Tuning**: Using GridSearchCV or RandomizedSearchCV for optimization.
* **Feature Selection**: Using techniques like Recursive Feature Elimination (RFE) or Principal Component Analysis (PCA).
* **Experimenting with Other Models**: Trying alternative algorithms like Random Forest, Gradient Boosting, or Neural Networks.

**8. Conclusion**

The ECP\_US\_Model applies data preprocessing techniques and machine learning algorithms to predict energy consumption in the U.S. Future improvements may include hyperparameter tuning, feature selection, and trying different models for better accuracy.