Energy Demand Forecasting:
Predicting Energy Consumption/ EV Charging Behavior

Pritam Gajbhiye

Video Presentation: Click Here

Research Question

Motivation:

Studying energy demand forecasting for EV charging stations offers the opportunity to optimize resource allocation, minimize costs, and support sustainable transportation, contributing to efficient infrastructure planning and environmental conservation.

Research Question:

How can predictive modeling techniques be applied to accurately forecast energy demand for Electric Vehicle (EV) charging stations.

Data Source

- Electric Vehicle Charging Station Usage, Palo Alto, California

  https://data.cityofpaloalto.org/dataviews/244892/ELECT-VEHIC-CHARG-STATI-66721/

- Time Series Data
  - 07/29/2011 - 03/15/2013
  - Shape - 9999 Rows, 35 Columns
  - 5 Charging Stations Across the City

Model

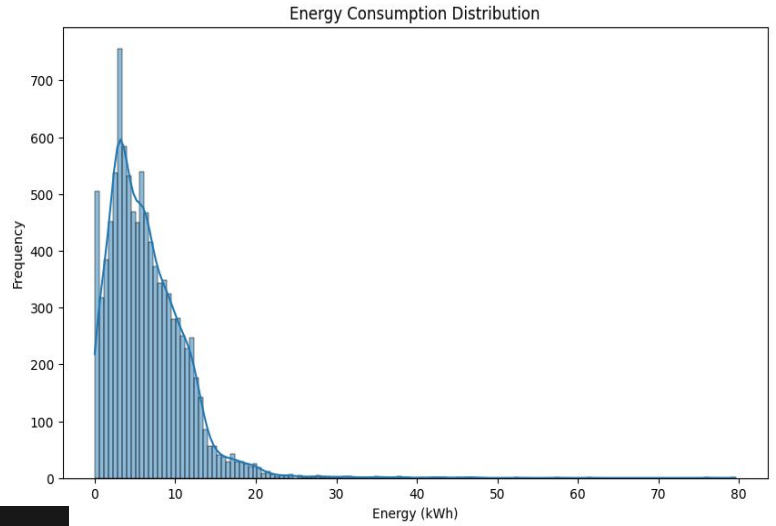
- Regression Task - Energy (kWh) Consumption, Continuous Variable

- **Machine Learning Models -**
  - Decision Tree and Random Forest
  - Linear Regression
  - Ridge and Lasso Regression

- **Deep Learning Models -**
  - Long Short Term Memory (LSTM)
  - Neural Basis Expansion Analysis Time Series (NBEATS)

Data Processing

- **Removed Outliers** i.e., Energy (kWh) > 20
- **For Machine Learning Task** -
  - Extracted Other Variables from Start Time Variable
  - Year, Month, Day, Hour, Weekday, Weekend
- **For Deep Learning Task** -
  - Convert the data into Time Series format
  - Aggregate Energy Consumption on Daily Basis
- **Train Test Split -**
  - **Test Size 20% (For ML Task)**
  - **Split After - 01/15/2013 (For DL Task)**




```
# get the features and target variable
x , y = data[['Park Duration (mins)', 'Charge Duration (mins)', 'Year',
              'Month', 'Hour', 'Weekday', 'Weekend', 'Day']], data[["Energy (kWh)"]]
```

Decision Tree and Random Forest

- Testing Accuracy, Decision Tree - 69.1%

```
# create a decision tree regressor model with best hyperparameters
dt_reg2 = DecisionTreeRegressor(max_depth=5, min_samples_leaf=20, min_samples_split=4, random_state=42)

# fit the model
dt_reg2 = dt_reg2.fit(x_train, y_train)
```

- Testing Accuracy, Random Forest - 72.3%

```
# create a random forest regressor model
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)
```

Park Duration (mins)	Charge Duration (mins)	Year	Month	Hour	Weekday	Weekend	Day	StartDate Integer	EndDate Integer
0.05	0.77	0.0	0.02	0.05	0.02	0.0	0.03	0.03	0.03

Linear, Polynomial, Ridge and Lasso Regression

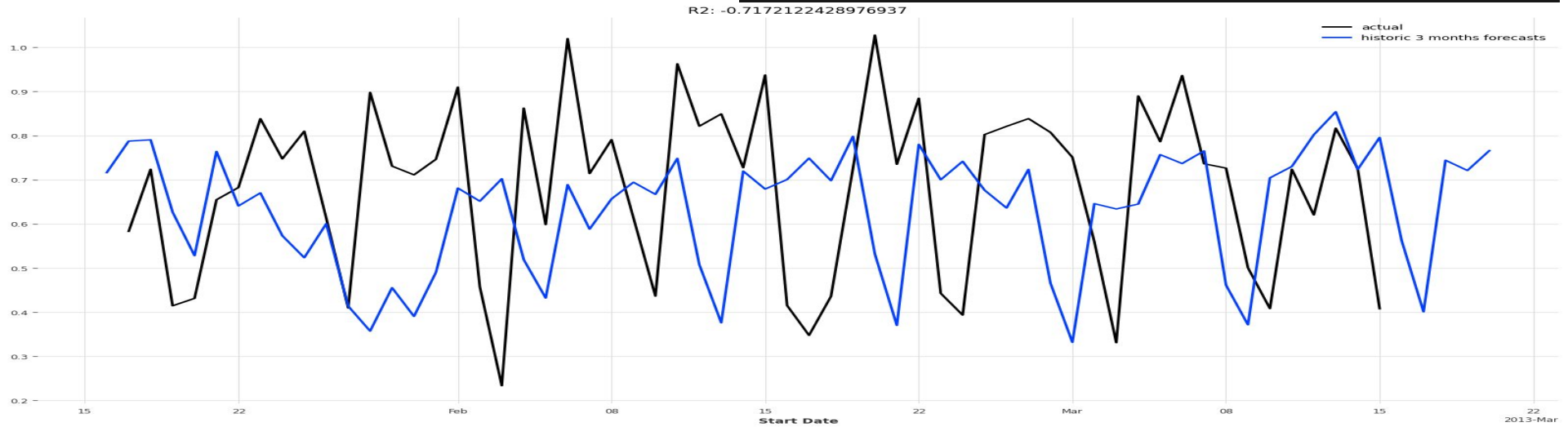
- Testing Accuracy
  - Linear - 63.6%
  - Ridge - 63.6%
  - Lasso - 63.4%
  - Polynomial with Degree 2 - 67.9%
  - Polynomial with Degree 3 - 39%

	features	coefficients
0	Park Duration (mins)	-0.000
1	Charge Duration (mins)	0.048
2	Year	-0.445
3	Month	-0.079
4	Hour	0.051
5	Weekday	-0.019
6	Weekend	0.235
7	Day	-0.002

NBEATS

MAPE: 28.89%

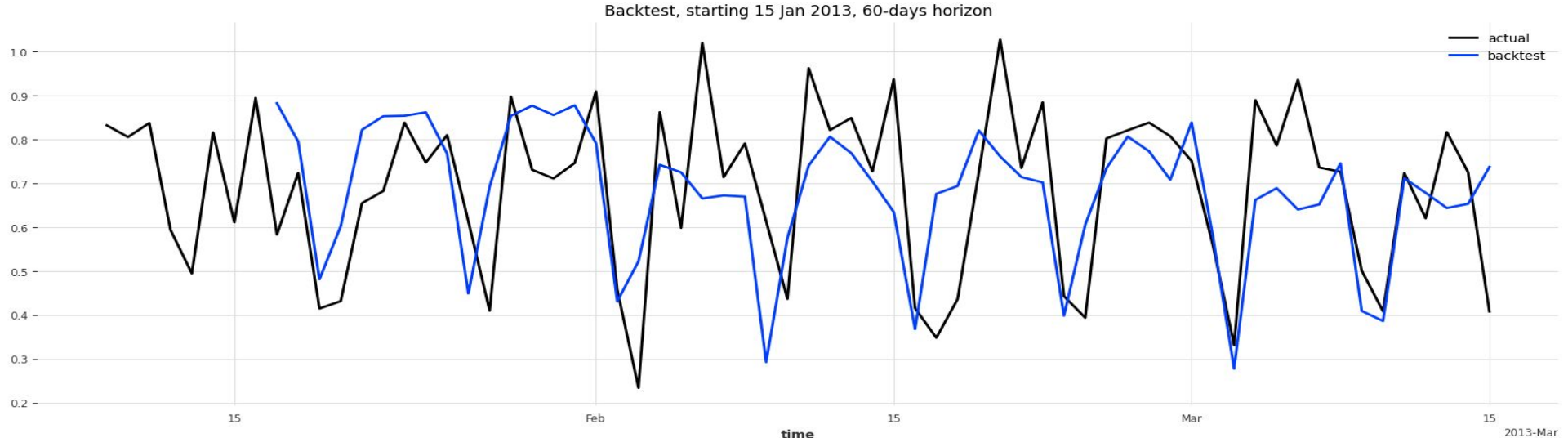
```python
# define the NBEATS model
model_name = "nbeats_run"
model_nbeats = NBEATSModel(
    input_chunk_length=30, output_chunk_length=7,
    generic_architecture=True,
    num_stacks=10, num_blocks=1, num_layers=4,
    layer_widths=512, n_epochs=100,
    nr_epochs_val_period=1, batch_size=800,
    random_state=42,model_name=model_name,
    save_checkpoints=True,force_reset=True,
    **generate_torch_kwargs(),
)
```



RNN - LSTM

MAPE: 22.03%

```python
#define RNN -LSTM model
my_model = RNNModel(
    model="LSTM",
    hidden_dim=20, dropout=0, batch_size=16,
    n_epochs=200, optimizer_kwargs={"lr": 1e-3},
    model_name="Energy_RNN",log_tensorboard=True,
    random_state=42,training_length=20,
    input_chunk_length=14,force_reset=True,
    save_checkpoints=True,
)
```



Error Analysis - LSTM

- High Variability in Charging Patterns
- Data Anomalies or Outliers
- Insufficient Training Data
- Seasonal Patterns
- Incomplete Feature Representation
- Long Term Trends

Potential Solutions -

- Feature Engineering
- Model Architecture Optimization
- Ensemble Learning

Conclusion

- Deep Learning models such as LSTM are well suited for time series forecasting tasks.
  - Ability to effectively capture and learn from long-term dependencies and temporal patterns present in sequential data
  - Memory cells allow them to retain information over extended time intervals
  - Automatically adapt to varying sequence lengths
  - Capable of learning hierarchical representations

Thank You !