PHASE 2 PROJECT
MEASURE ENERGY CONSUMPTION

Done by Sathya.P, Sowmiya.S
,Surya.L, Swastina.S, Vasundhra.J,
Vimala devi.S

OVERVIEW

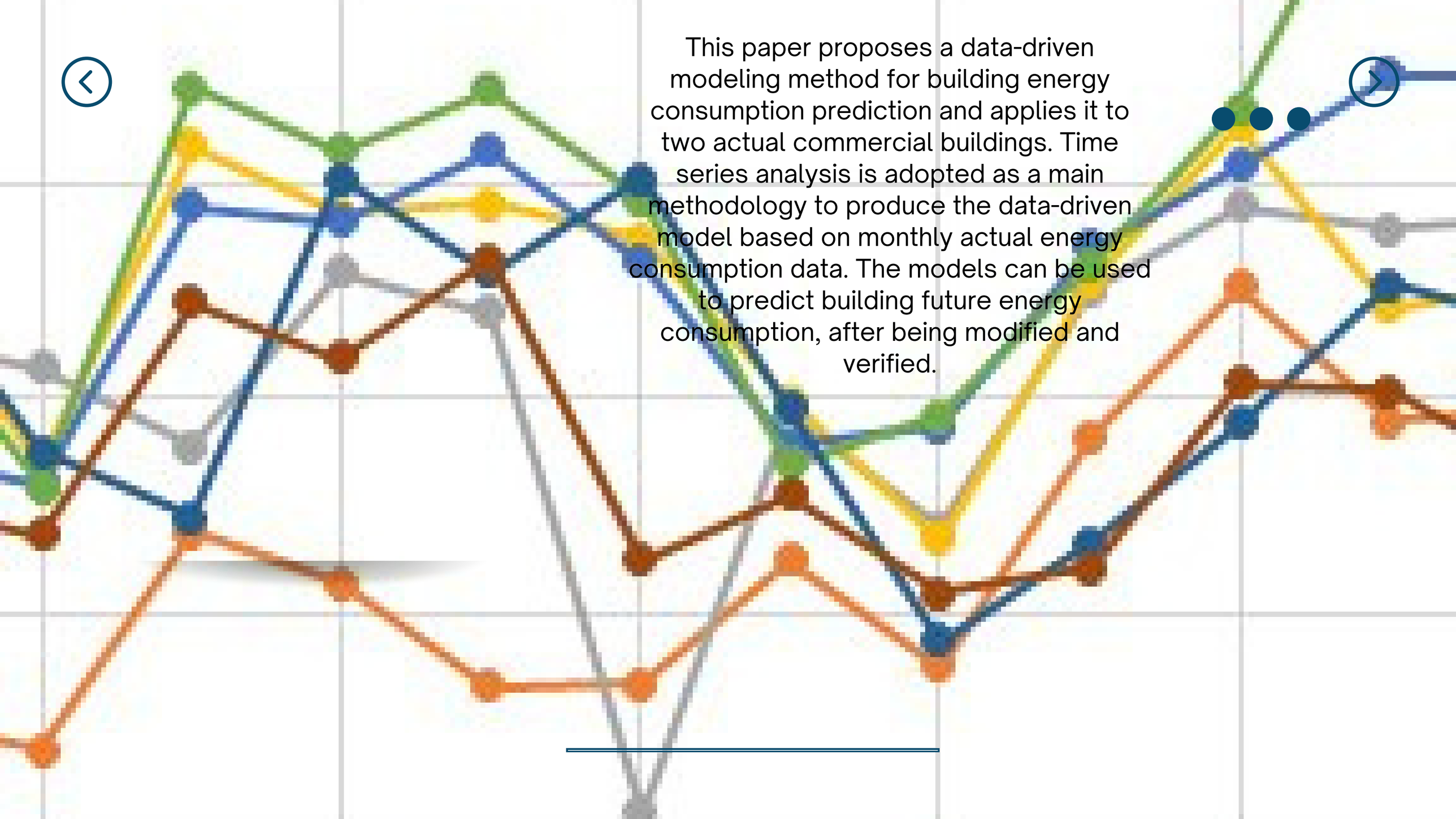
- Problem definition and design thinking
- Innovation
- Development part 1
- Development part 2
- Documentation

PROBLEM DEFINITION

- The problem at hand is to create an automated system that measures the energy consumption, analyzes the data, and provide visualisations for informed decision making.This solution aims to enhance efficiency ,accuracy, and ease of understanding in managing energy consumption across various sectors.

DESIGN THINKING

- Data Source: Identify an available dataset containing energy consumption measurements.
- Data Preprocessing: Clean, transform, and prepare the dataset for analysis.
- Feature Extraction: Extract relevant features and metrics from the energy consumption data.
- Model Development: Utilize statistical analysis to uncover trends, patterns, and anomalies in the data.
- Visualization: Develop visualizations (graphs, charts) to present the energy consumption trends and insights.
- Automation: Build a script that automates data collection, analysis, and visualization processes.



This paper proposes a data-driven modeling method for building energy consumption prediction and applies it to two actual commercial buildings. Time series analysis is adopted as a main methodology to produce the data-driven model based on monthly actual energy consumption data. The models can be used to predict building future energy consumption, after being modified and verified.

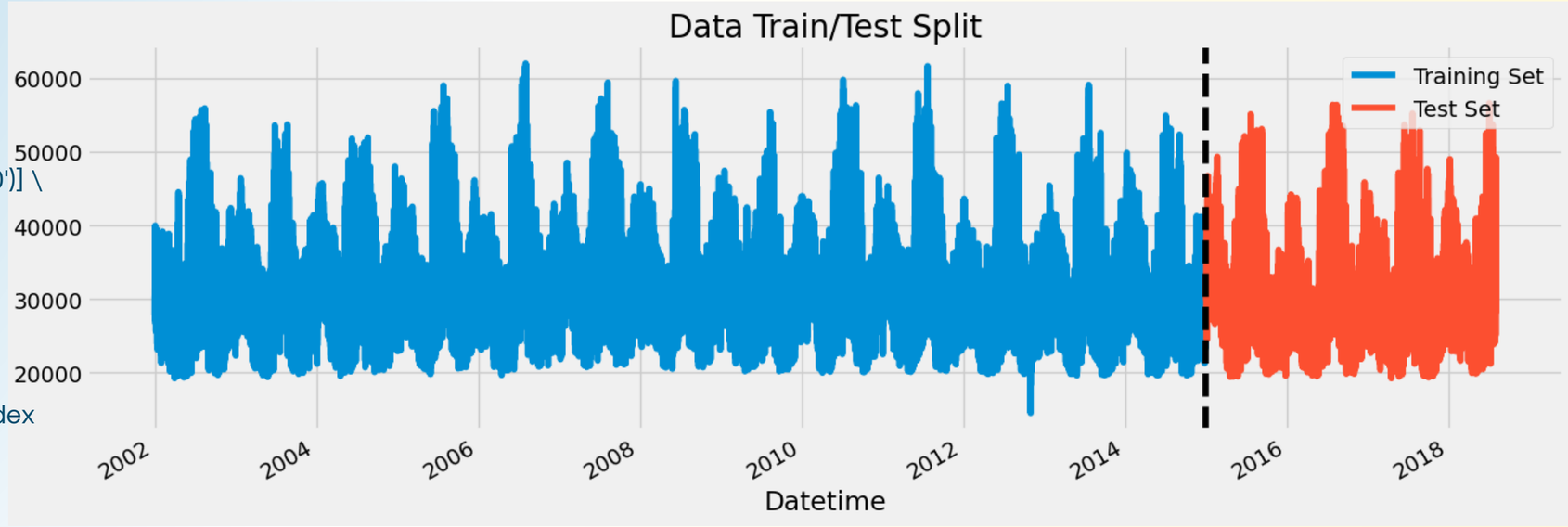
```
import pandas as p
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import xgboost as xgb
from sklearn.metrics import mean_squared_error
color_pal = sns.color_palette()
plt.style.use('fivethirtyeight')
df = pd.read_csv('../input/hourly-energy-consumption/PJME_hourly.csv')
df = df.set_index('Datetime')
df.index = pd.to_datetime(df.index)
df.plot(style='.',
        figsize=(15, 5),
        color=color_pal[0],
        title='PJME Energy Use in MW')
plt.show()
```

- Train / Test Split

```
train = df.loc[df.index < '01-01-2015']
test = df.loc[df.index >= '01-01-2015']

fig, ax = plt.subplots(figsize=(15, 5))
train.plot(ax=ax, label='Training Set', title='Data Train/Test Split')
test.plot(ax=ax, label='Test Set')
ax.axvline('01-01-2015', color='black', ls='--')
ax.legend(['Training Set', 'Test Set'])
plt.show()

df.loc[(df.index > '01-01-2010') & (df.index < '01-08-2010')] \
.plot(figsize=(15, 5), title='Week Of Data')
plt.show()
```



Feature Creation

```
def create_features(df):
    Create time series features based on time series index
    df = df.copy()
    df['hour'] = df.index.hour
    df['dayofweek'] = df.index.dayofweek
    df['quarter'] = df.index.quarter
    df['month'] = df.index.month
    df['year'] = df.index.year
    df['dayofyear'] = df.index.dayofyear
    df['weekofmonth'] = df.index.day
    df['weekofyear'] = df.index.isocalendar().week
    return df

df = create_features(df)
```

- Visualize our Feature / Target Relationship

```
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x='hour', y='PJME_MW')
ax.set_title('MW by Hour')
plt.show()

fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x='month', y='PJME_MW', palette='Blues')
ax.set_title('MW by Month')
plt.show()
```

Create our Model

```
train = create_features(train)
test = create_features(test)

FEATURES = ['dayofyear', 'hour', 'dayofweek', 'quarter', 'month', 'year']
```

- TARGET = 'PJME_MW'

```
x_train = train[FEATURES]
y_train = train[TARGET]

x_test = test[FEATURES]
y_test = test[TARGET]
reg =xgb.XGBRegressor(base_score=0.5, booster='gbtree', n_estimators=1000,
                      early_stopping_rounds=50,
                      objective='reg:linear',
                      max_depth=3,
                      learning_rate=0.01)
reg.fit(x_train, y_train,
        eval_set= [(x_train, y_train), (x_test, y_test)],
        verbose=100)
XGBRegressor
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=50,

             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,

             interaction_constraints=None, learning_rate=0.01, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=3, max_leaves=None,

             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=1000, n_jobs=None, num_parallel_tree=None,
             objective='reg:linear', predictor=None, ...)
```

- Feature Importance

```
fi = pd.DataFrame(data=reg.feature_importances_,
                  index=reg.feature_names_in_,
                  columns=['importance'])
fi.sort_values('importance').plot(kind='barh', title='Feature Importance')
plt.show()
```

Forecast on Test

```
test['prediction'] = reg.predict(x_test)
df = df.merge(test[['prediction']], how= 'left', left_index=True, right_index=True)
ax = df[['PJME_MW']].plot(figsize=(15, 5))
df['prediction'].plot(ax=ax, style='.')
plt.legend(['Truth Data', 'Predication'])
ax.set_title('Raw Dat and Prediction')
plt.show()

ax = df.loc[(df.index > '04-01-2018') & (df.index < '04-08-2018')]['PJME_MW'] \
    .plot(figsize=(15, 5), title='Week Of Data')
df.loc[(df.index > '04-01-2018') & (df.index < '04-08-2018')]['prediction'] \
    .plot(style='.')
plt.legend(['Truth Data','Prediction'])
plt.show()
```

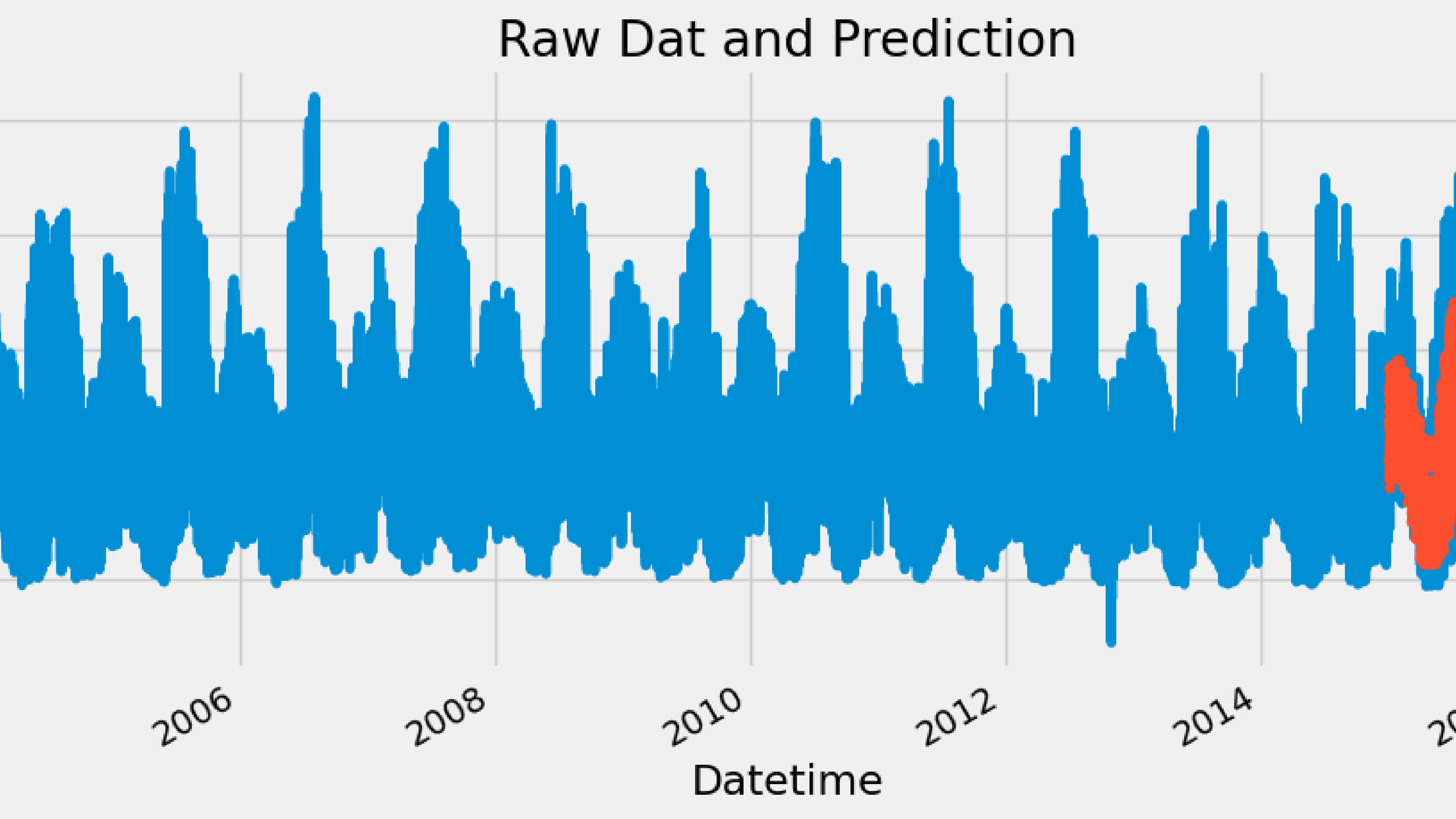
- Score (RMSE)

```
score = np.sqrt(mean_squared_error(test['PJME_MW'], test['prediction']))
print(f'RMSE Score on Test set: {score:0.2f}')
RMSE Score on Test set: 3721.75
Calculate Error Look at the worst and best predicted days

test['error'] = np.abs(test[TARGET] - test['prediction'])
test['date'] = test.index.date
test.groupby(['date'])['error'].mean().sort_values(ascending=False).head(10)
date
```

Output

- 2016-08-13 12839.597087
- 2016-08-14 12780.209961
- 2016-09-10 11356.302979
- 2015-02-20 10965.982259
- 2016-09-09 10864.954834
- 2018-01-06 10506.845622
- 2016-08-12 10124.051595
- 2015-02-21 9881.803711
- 2015-02-16 9781.552246
- 2018-01-07 9739.144206


Raw Dat and Prediction
2006
2008
2010
2012
2014
Datetime

Thank you