**Measure Energy Consumption**

**VASANTH RAM P**

**Au410721104121**

**Phase-3 Development Part 1**

**Project :Measure Energy Consumption**

**Phase-3 : Development Part 1**

**Topic :In this part you will begin building your project by loading and preprocessing the dataset.**

Measuring energy consumption helps us keep track of how much energy we're using and where it's going. There are several ways to do this. First, you can check your energy bills, which show how much electricity or gas you've used over a month or year.

Another way is by installing energy meters. These devices, like smart meters or submeters, measure your energy use in real-time. You can also manually read your utility meters for a basic idea of usage.

For more detailed insights, you can use energy monitoring devices or apps. They provide real-time data and can even tell you how much specific appliances are using. Energy audits can help professionals identify ways to use energy more efficiently in your home or business.

Looking at historical data can show usage patterns and when you use the most energy. Software, online tools, and benchmarking can provide more detailed analysis and comparisons to industry standards. Submetering allows businesses to measure energy use in specific areas or equipment.

By tracking and understanding your energy consumption, you can make informed choices to save money and reduce your environmental impact.

**Given Dataset**

|  |  |
| --- | --- |
| **Datetime** | **# PJM\_Load\_MW** |
| 1998-12-31 01:00:00 | 29309.0 |
| 1998-12-31 02:00:00 | 28236.0 |
| 1998-12-31 03:00:00 | 27692.0 |
| 1998-12-31 04:00:00 | 27596.0 |
| 1998-12-31 05:00:00 | 27888.0 |
| 1998-12-31 06:00:00 | 29382.0 |
| 1998-12-31 07:00:00 | 31373.0 |
| 1998-12-31 08:00:00 | 33272.0 |
| 1998-12-31 09:00:00 | 34133.0 |
| 1998-12-31 10:00:00 | 35232.0 |

**Importance of Loading and preprocessing datasets**

Loading and preprocessing datasets is a fundamental and critical step in data analysis, machine learning, and data-driven decision-making. Here are several key reasons for the importance of these tasks:

**Data Quality Assurance:** Loading and preprocessing data allow you to inspect and ensure the quality of the dataset. You can identify and address issues like missing values, outliers, and inconsistencies, which are essential for reliable and accurate analysis.

**Data Understanding:** Loading the data provides an initial understanding of its structure, size, and format. It helps you become familiar with the dataset, which is necessary for planning your analysis, choosing the right techniques, and making informed decisions.

**Data Cleaning:** Preprocessing involves data cleaning, which includes handling missing data, removing duplicates, and addressing outliers. Clean data is vital for obtaining meaningful and reliable insights, as well as for building robust machine learning models.

**Feature Engineering:** Preprocessing often involves feature engineering, where you create, transform, or select features to enhance the predictive power of your models or to gain better insights in data analysis.

**Normalization and Scaling:** Preprocessing ensures that data is on a consistent scale, which is particularly important for many machine learning algorithms. Normalization and scaling help prevent features with larger values from dominating the learning process and ensure the algorithms work effectively.

**Categorical Data Handling:** Many datasets include categorical variables that need to be transformed into numerical format for machine learning models to process. Preprocessing helps in encoding or transforming these variables, making them suitable for analysis.

**Efficiency:** Proper preprocessing can lead to more efficient analysis and modeling. By removing unnecessary or redundant information and optimizing data structures, you can speed up computations and reduce resource requirements.

**Data Privacy and Security:** Preprocessing can also play a role in data privacy and security. Steps like anonymization and data masking can be part of the preprocessing process to protect sensitive information.

**Python Program**

import matplotlib.pyplot as plt *# plotting*

import numpy as np *# linear algebra*

import os # accessing directory structure

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

import seaborn as sns

plt.style.use('ggplot') *# Make it pretty*

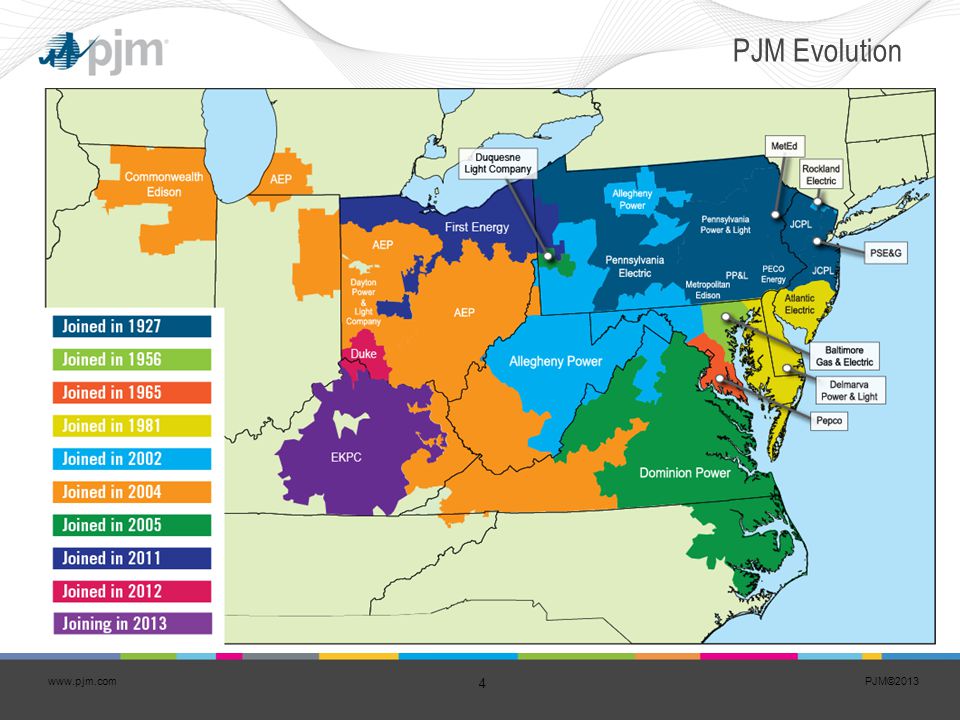
*# Data is saved in parquet format so schema is preserved.*

df = pd.read\_parquet('../input/est\_hourly.paruqet')

*#Show PJM Regions*

from IPython.display import Image

Image(url= "http://slideplayer.com/4238181/14/images/4/PJM+Evolution.jpg")

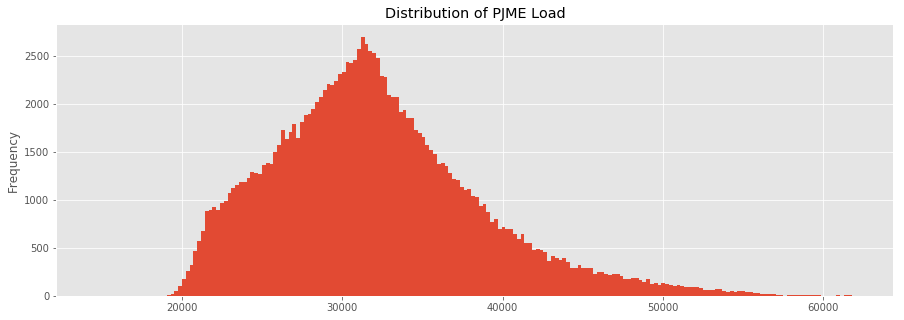


df.head()

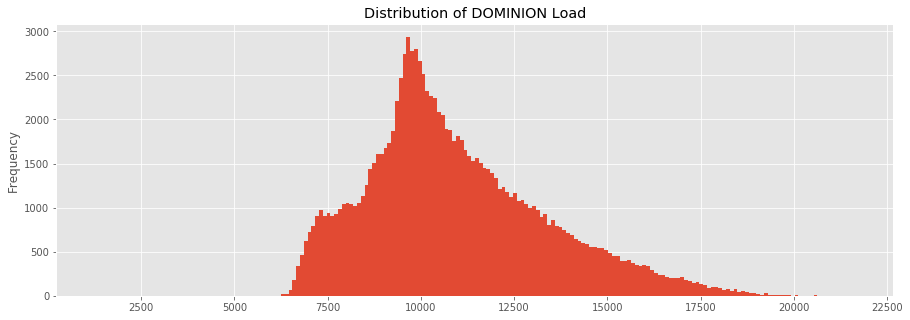
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | AEP | COMED | DAYTON | DEOK | DOM | DUQ | EKPC | FE | NI | PJME | PJMW | PJM\_Load |
| Datetime |  |  |  |  |  |  |  |  |  |  |  |  |
| 1998-12-31 01:00:00 | NaN | NaN | NaN | NaN | NaNz | NaN | NaN | NaN | NaN | NaN | NaN | 29309.0 |
| 1998-12-31 02:00:00 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 28236.0 |
| 1998-12-31 03:00:00 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 27692.0 |
| 1998-12-31 04:00:00 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 27596.0 |
| 1998-12-31 05:00:00 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 27888.0 |

df.describe().T

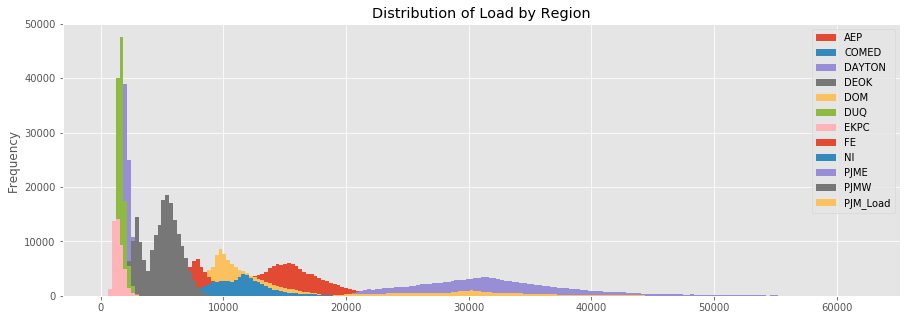
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | count | mean | std | min | 25% | 50% | 75% | max |
| AEP | 121273.0 | 15499.513717 | 2591.399065 | 9581.0 | 13630.0 | 15310.0 | 17200.00 | 25695.0 |
| COMED | 66497.0 | 11420.152112 | 2304.139517 | 7237.0 | 9780.0 | 11152.0 | 12510.00 | 23753.0 |
| DAYTON | 121275.0 | 2037.851140 | 393.403153 | 982.0 | 1749.0 | 2009.0 | 2279.00 | 3746.0 |
| DEOK | 57739.0 | 3105.096486 | 599.859026 | 907.0 | 2687.0 | 3013.0 | 3449.00 | 5445.0 |
| DOM | 116189.0 | 10949.203625 | 2413.946569 | 1253.0 | 9322.0 | 10501.0 | 12378.00 | 21651.0 |
| DUQ | 119068.0 | 1658.820296 | 301.740640 | 1014.0 | 1444.0 | 1630.0 | 1819.00 | 3054.0 |
| EKPC | 45334.0 | 1464.218423 | 378.868404 | 514.0 | 1185.0 | 1386.0 | 1699.00 | 3490.0 |
| FE | 62874.0 | 7792.159064 | 1331.268006 | 0.0 | 6807.0 | 7700.0 | 8556.00 | 14032.0 |
| NI | 58450.0 | 11701.682943 | 2371.498701 | 7003.0 | 9954.0 | 11521.0 | 12896.75 | 23631.0 |
| PJME | 145366.0 | 32080.222831 | 6464.012166 | 14544.0 | 27573.0 | 31421.0 | 35650.00 | 62009.0 |
| PJMW | 143206.0 | 5602.375089 | 979.142872 | 487.0 | 4907.0 | 5530.0 | 6252.00 | 9594.0 |
| PJM\_Load | 32896.0 | 29766.427408 | 5849.769954 | 17461.0 | 25473.0 | 29655.0 | 33073.25 | 54030.0 |

\_ = df['PJME'].plot.hist(figsize=(15, 5), bins=200, title='Distribution of PJMELoad')

\_ = df['DOM'].plot.hist(figsize=(15, 5), bins=200, title='Distribution of DOMINION Load')

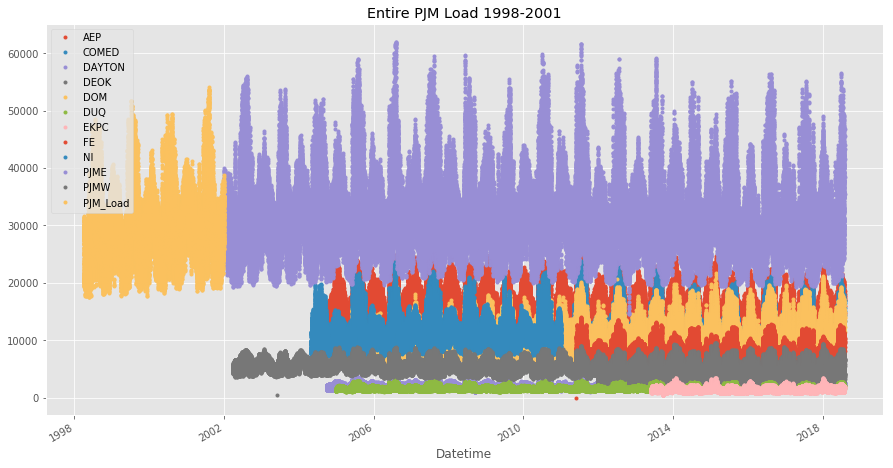


\_ = df.plot.hist(figsize=(15, 5), bins=200, title='Distribution of Load by Region')



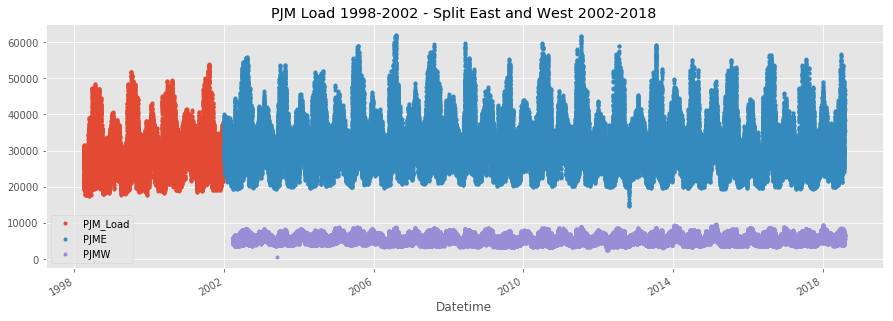
## Plot Time Series

plot = df.plot(style='.', figsize=(15, 8), title='Entire PJM Load 1998-2001')



# Plotting Regions

\_ = df[['PJM\_Load','PJME','PJMW']] \.plot(style='.', figsize=(15, 5), title='PJM Load 1998-2002 - Split East and West 2002-2018')

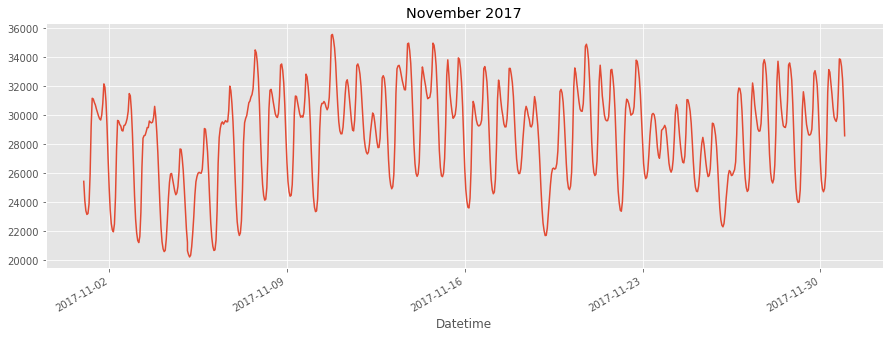


## Summer Demand vs Winter Demand

Note the dips mid-day in the winter months. Conversely in summer months the daily load is more bell shaped. This is due to high mid-day energy consumtion by air conditioning. In winter months people tend to use less energy mid-day.

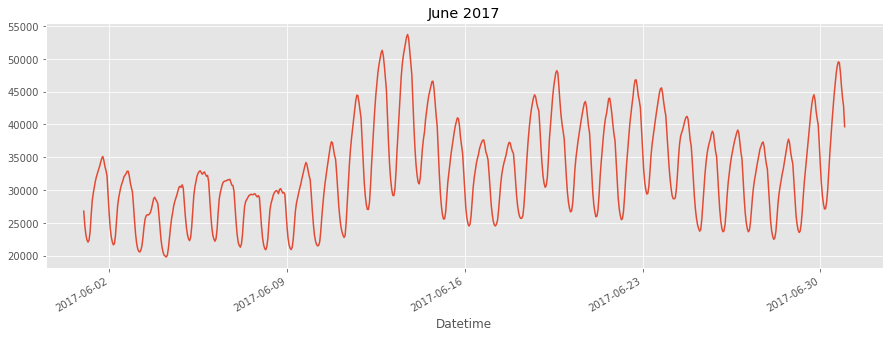
\_ = df['PJME'].loc[(df['PJME'].index >= '2017-11-01') &(df['PJME'].index < '2017-12-01')] \

.plot(figsize=(15, 5), title = 'November 2017')



\_ = df['PJME'].loc[(df['PJME'].index >= '2017-06-01') & (df['PJME'].index < '2017-07-01')] \

.plot(figsize=(15, 5), title = 'June 2017')



# Create Time Series Features

df['dow'] = df.index.dayofweek

df['doy'] = df.index.dayofyear

df['year'] = df.index.year

df['month'] = df.index.month

df['quarter'] = df.index.quarter

df['hour'] = df.index.hour

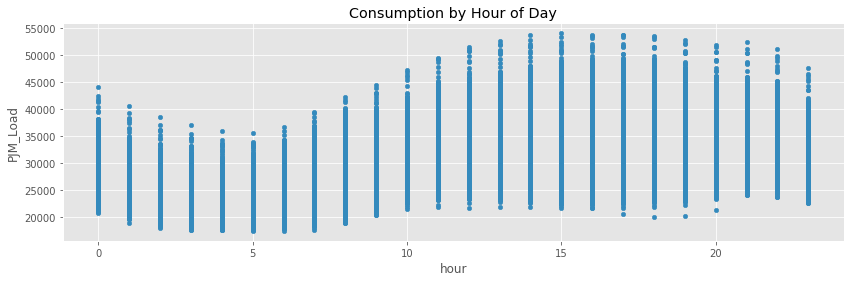
df['weekday'] = df.index.weekday\_name

df['woy'] = df.index.weekofyear

df['dom'] = df.index.day # Day of Month

df['date'] = df.index.date

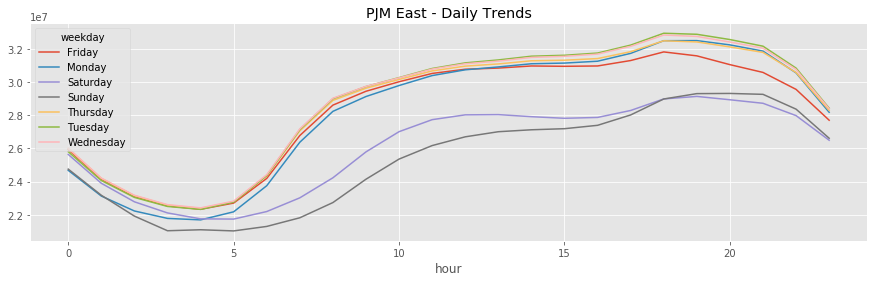
\_ = df[['PJM\_Load','hour']].plot(x='hour',y='PJM\_Load',kind='scatter',figsize=(14,4),title='Consumption by Hour of Day')



Note Saturday and Sunday demand is much less than during a work week. This is also true for holidays.

\_ = df.pivot\_table(index=df['hour'], columns='weekday', values='PJME',

aggfunc='sum').plot(figsize=(15,4), title='PJM East - Daily Trends')



## Trends change depending on time of year

fig, ax = plt.subplots(figsize=(15,5))

sns.boxplot(df.loc[df['quarter']==1].hour, df.loc[df['quarter']==1].PJME)

ax.set\_title('Hourly Boxplot PJME Q1')

ax.set\_ylim(0,65000)

fig, ax = plt.subplots(figsize=(15,5))

sns.boxplot(df.loc[df['quarter']==2].hour, df.loc[df['quarter']==2].PJME)

ax.set\_title('Hourly Boxplot PJME Q2')

ax.set\_ylim(0,65000)

fig, ax = plt.subplots(figsize=(15,5))

sns.boxplot(df.loc[df['quarter']==3].hour, df.loc[df['quarter']==3].PJME)

ax.set\_title('Hourly Boxplot PJME Q3')

ax.set\_ylim(0,65000)

fig, ax = plt.subplots(figsize=(15,5))

sns.boxplot(df.loc[df['quarter']==4].hour, df.loc[df['quarter']==4].PJME)

ax.set\_title('Hourly Boxplot PJME Q4')

\_ = ax.set\_ylim(0,65000)

