MEASURE ENERGY CONSUMPTION

ABSTRACT

Energy consumption has been widely studied in the computer architecture field for decades. While the adoption of energy as a metric in machine learning is emerging, the majority of research is still primarily focused on obtaining high levels of accuracy without any computational constraint.We believe that one of the reasons for this lack of interest is due to their lack of familiarity with approaches to evaluate energy consumption. To address this challenge, we present a review of the different approaches to estimate energy consumption in general and machine learning applications in particular. Our goal is to provide useful guidelines to the machine learning community giving them the fundamental knowledge to use and build specific energy estimation methods for machine learning algorithms. We also present the latest software tools that give energy estimation values, together with two use cases that enhance the study of energy consumption in machine learning.

DATA PREPROCESSING

Data preprocessing is the process of preparing data for analysis by cleaning, transforming, and selecting relevant features. It involves identifying and handling **missing**or **duplicate** **data**, **scaling features**, **encoding**  and **splitting data**into training and testing sets.

FUTURE EXTRACTION

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data*.*

MODEL DEVELOPMENT

The ML model development involves data acquisition from multiple trusted sources, data processing to make suitable for building the model, choose algorithm to build the model, build model, compute performance metrics and choose best performing model.

©2019 The Authors. Publis

©2019 The Authors. Published b

©2019 The Authors. Published b

CODE

In [1]:

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

In [2]:

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

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

Data index is the date/hour, columns are for different regions within PJM.

Regions joined at different times, so not all have data for all dates. Regions also split (PJM\_Load split to East and West)

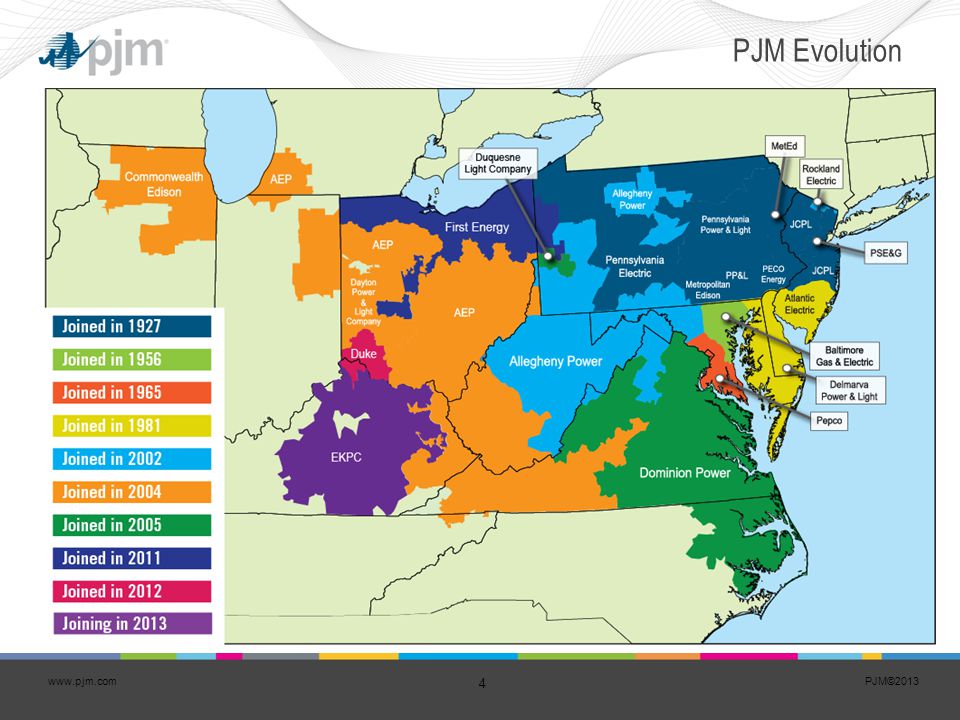
In [3]:

#Show PJM Regions

from IPython.display import Image

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

OUTPUT



In [4]:

df.head()

OUTPUT

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | AEP | COMED | DAYTON | DEOK | DOM | DUQ | EKPC | FE | NI | PJME | PJMW | PJM\_LOAD |
| DATETIME |  |  |  |  |  |  |  |  |  |  |  |  |
| 1998  12-31  01-00-00 | NaN | NaN | NaN | NaN | NaN | 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 |

In [5]:

df.describe().T

OUTPUT

count mean std min 25% 50% 75% max

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

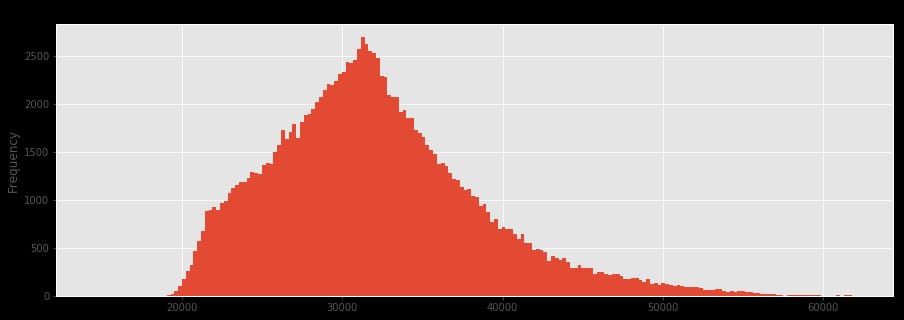
PJM\_Load 32896.0 29766.427408 5849.769954 17461.0 25473.0 29655.0 33073.25 54030.0

In [6]:

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

OUTPUT

DISTRIBUTION OF PJME LOAD

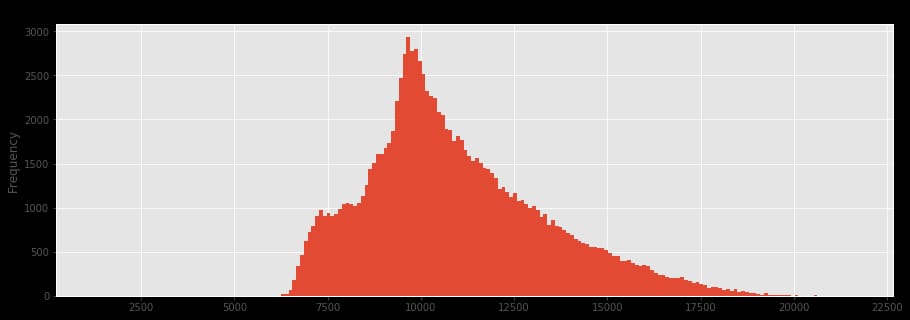


[7]:

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

OUTPUT

DISTRIBUTION OF DOMINION LOAD

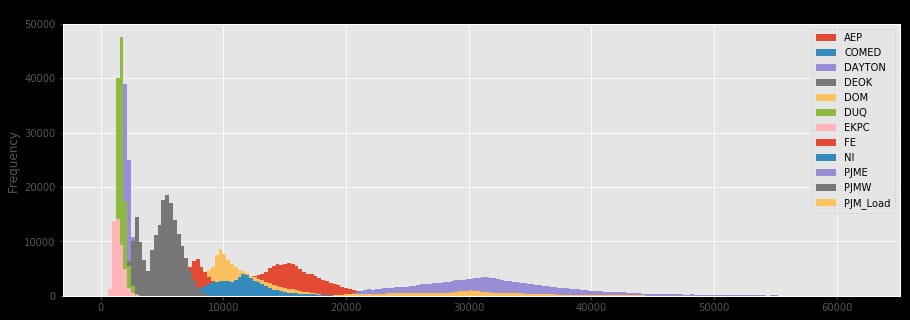


In [8]:

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

OUTPUT

DISTRIBUTION OF LOAD BY REGION



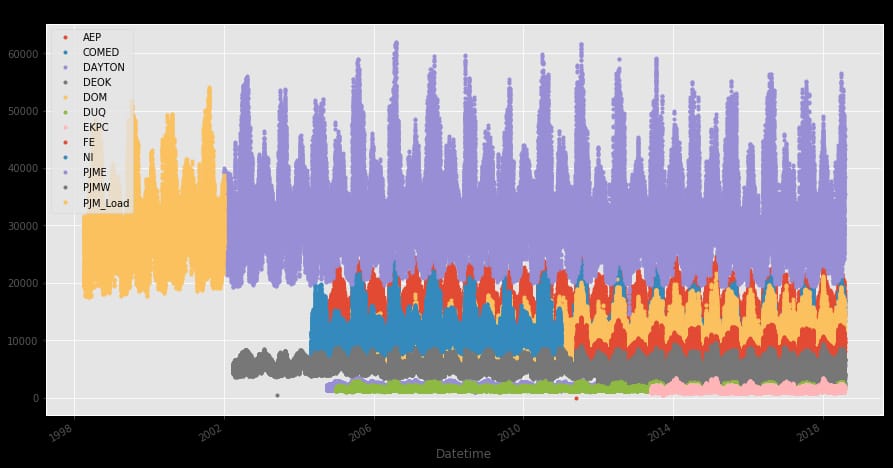
Plot Time Series

In [9]:

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

OUTPUT

ENTIRE PJM LOAD 1998-2001



Plotting Regions

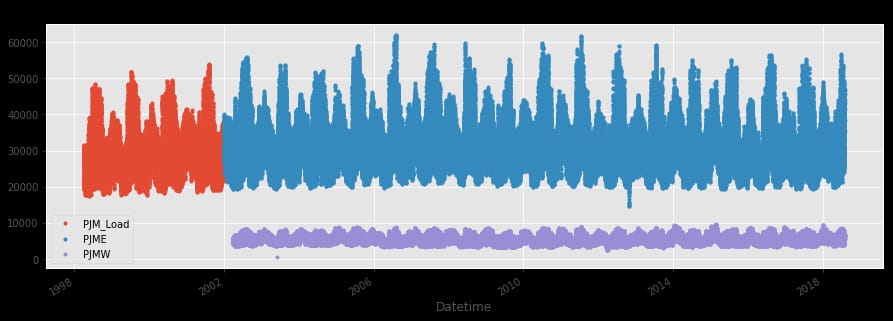
In [10]:

\_ = df[['PJM\_Load','PJME','PJMW']] \

.plot(style='.', figsize=(15, 5), title='PJM Load 1998-2002 - Split East and West 2002-2018')

OUTPUT

PJM Load 1998-2002 - Split East and West 2002-2018



Summer Demand vs Winter Demand

In [11]:

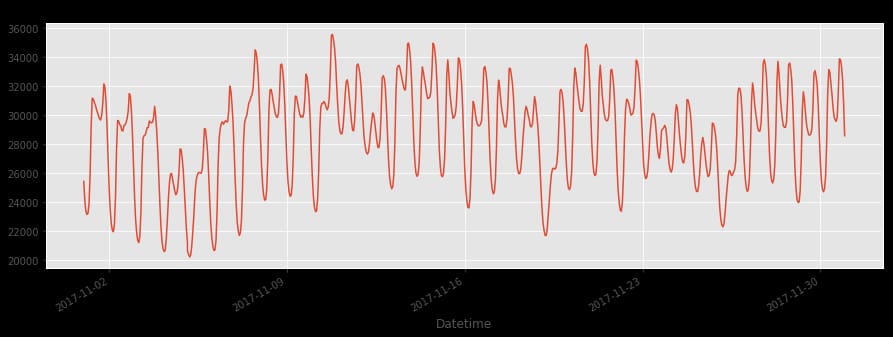
\_ = df['PJME'].loc[(df['PJME'].index >= '2017-11-01') &

(df['PJME'].index < '2017-12-01')] \

.plot(figsize=(15, 5), title = 'November 1999')

OUTPUT

NOVEMBER 1999



In [12]:

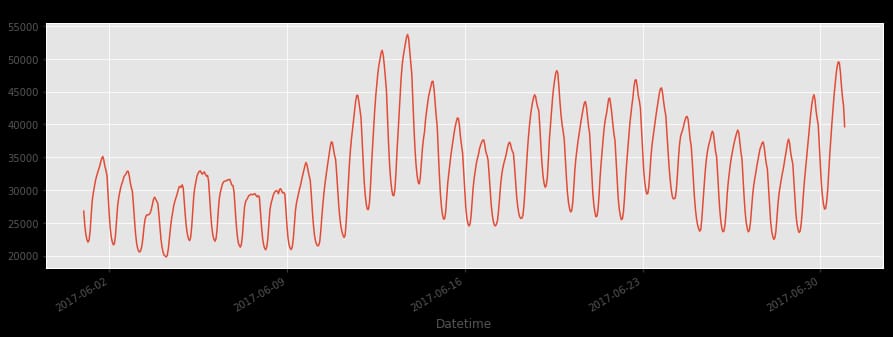
\_ = df['PJME'].loc[(df['PJME'].index >= '2017-06-01') &

(df['PJME'].index < '2017-07-01')] \

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

OUTPUT

JUNE 2017



Create Time Series Features

In [13]:

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

In [14]:

\_ = df[['PJM\_Load','hour']].plot(x='hour',

y='PJM\_Load',

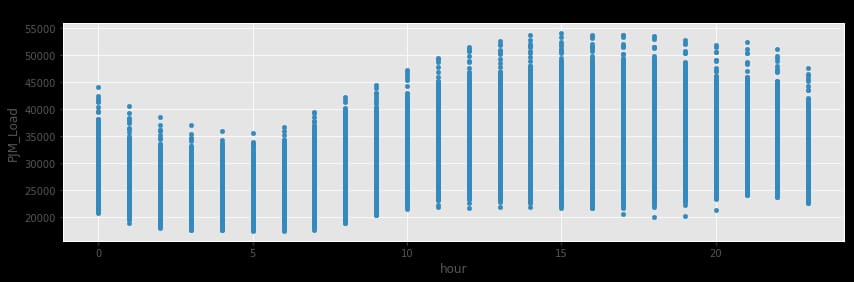
kind='scatter',

figsize=(14,4),

title='Consumption by Hour of Day')

OUTPUT

CONSUMPTION BY HOUR OF DAY



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

In [15]:

\_ = df.pivot\_table(index=df['hour'],

columns='weekday',

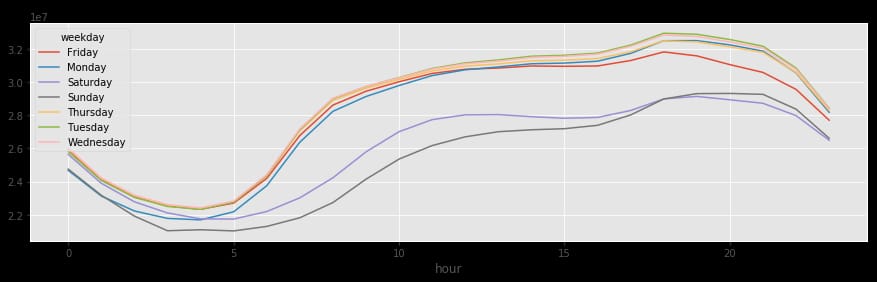
values='PJME',

aggfunc='sum').plot(figsize=(15,4),

title='PJM East - Daily Trends')

OUTPUT

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))

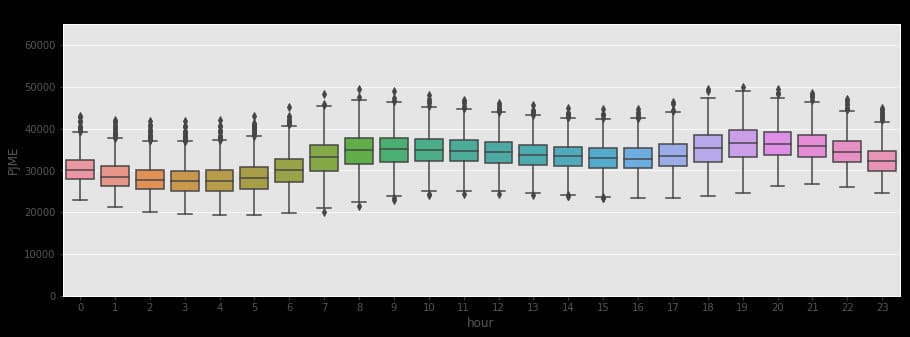
sns.boxplot(df.loc[df['quarter']==4].hour, df.loc[df['quarter']==4].PJME)

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

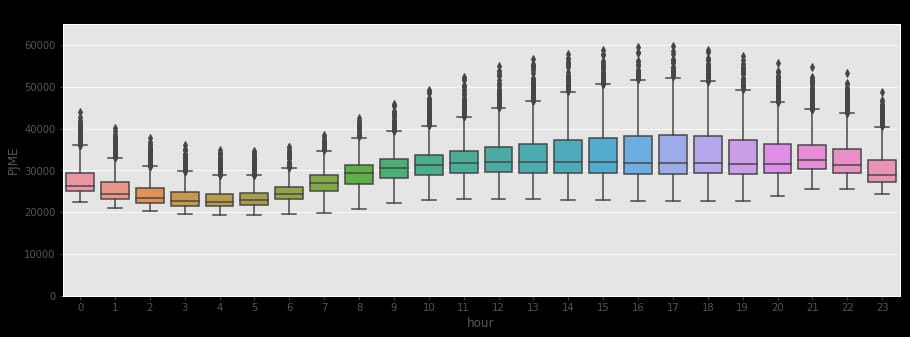
\_ = ax.set\_ylim(0,65000)

OUTPUT

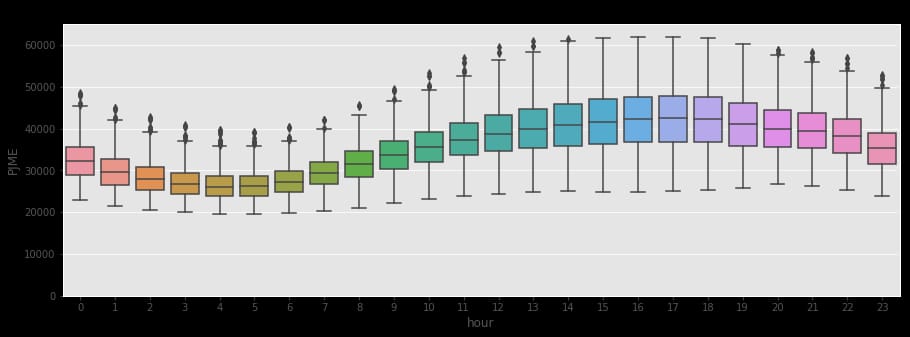
HOURLY BOXPLOT PJME Q1



HOURLY BOXPLOT PJME Q2



HOURLY BOXPLOT PJME Q3



HOURLY BOXPLOT PJME Q4

