A rainy day at Github by Shlomo The King.

Summary

- I do feature engineering to extract some new features from existing ones.
- I explored the data by plotting some of the features and looking at how they affect output.

References:

https://github.com/jonsedar/pymc3_vs_pystan/blob/master/30_BasicLinearRegression.ipynb (https://github.com/jonsedar/pymc3_vs_pystan/blob/master/30_BasicLinearRegression.ipynb)

http://nbviewer.jupyter.org/github/GoogleCloudPlatform/training-data-analyst/blob/master/CPB100/lab4a/demandforecast.ipynb (http://nbviewer.jupyter.org/github/GoogleCloudPlatform/training-data-analyst/blob/master/CPB100/lab4a/demandforecast.ipynb)

Install libs

```
#--- install libs, this is a docker image
# !pip install tensorflow watermark pymc3 xgboost edward pystan bokeh
# !pip install --upgrade pip
# !pwd
```

Import libs

```
#--- Libraries
import pandas  as pd                # stats packages
import numpy   as np                # linear algebra packages

import matplotlib.pyplot as plt     # ploting packages
import seaborn as sns               # more plotting routines

from scipy.stats import beta        # funtion defining beta distribution
from scipy.stats import binom       # funtion defining binomial distribution

import datalab.bigquery as bq
import shutil

#--- Configure plotting environments/defaults
# use 'cartoon-style'
plt.xkcd()
# use white background
sns.set_style('white')
# set color choices
c = sns.color_palette('deep')

# show plots in notebook
% matplotlib inline

from IPython.core.display import HTML
HTML("<style>.container { width:90% !important; } div.cell.selected { border-left-width: 1px !important;}</
>")
```

Setup

```
# filter warnings for presentation's sake
import warnings
warnings.filterwarnings('ignore')

# general packages
import sys
import sqlite3
from ipywidgets import interactive, fixed

# scientific packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import patsy as pt
from scipy import optimize
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors.kde import KernelDensity
import statsmodels.api as sm

# pymc3 libraries
import pymc3 as pm
import theano as thno
import theano.tensor as T
import pystan

sns.set(style="darkgrid", palette="muted")
pd.set_option('display.mpl_style', 'default')
plt.rcParams['figure.figsize'] = 12, 8
np.random.seed(0)
```

Versions

```
print('Python: {}'.format(sys.version))
print('Recursion limit {}'.format(sys.getrecursionlimit()))
print('theano: {}'.format(thno.__version__))
print('PyMC3: {}'.format(pm.__version__))
print('PyStan: {}'.format(pystan.__version__))
```

```
Python: 2.7.9 (default, Jun 29 2016, 13:08:31)
[GCC 4.9.2]
Recursion limit 10000
theano: 0.8.2
PyMC3: 3.0
PyStan: 2.14.0.0
```

Local Functions

```
spring = range(80, 172)
summer = range(172, 264)
fall = range(264, 355)

def get_season(doy):
    if doy in spring:
      season = 'spring'
    elif doy in summer:
      season = 'summer'
    elif doy in fall:
      season = 'fall'
    else:
      season = 'winter'
    return season

def prepare_plot_area(ax):
    # Remove plot frame lines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["left"].set_visible(False)

    # X and y ticks on bottom and left
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

# Defining a color pattern based
colrcode = [(31, 119, 180), (255, 127, 14),\
             (44, 160, 44), (214, 39, 40), \
             (148, 103, 189),  (140, 86, 75), \
             (227, 119, 194), (127, 127, 127), \
             (188, 189, 34), (23, 190, 207)]

for i in range(len(colrcode)):
    r, g, b = colrcode[i]
    colrcode[i] = (r / 255., g / 255., b / 255.)
```

Load Data

Basic meta data on tables

```
table = bq.Table('publicdata:samples.github_timeline')
table.metadata
```

```
INFO:oauth2client.client:Attempting refresh to obtain initial access_token
INFO:oauth2client.client:Refreshing access_token

<datalab.bigquery._table.TableMetadata at 0x7f8d1519a190>
```

```
# %%bigquery schema --table "publicdata:samples.github_timeline"
```

Weather data

```
# http://stackoverflow.com/questions/34804654/how-to-get-the-historical-weather-for-any-city-with-bigquery
```

```
table = bq.Table('fh-bigquery:weather_gsod.gsod2015')
table.metadata.rows
```

```
INFO:oauth2client.client:Attempting refresh to obtain initial access_token
INFO:oauth2client.client:Refreshing access_token

4200939
```

```
# df_avgquery = bq.Query(avgquery, YEAR=2015).to_dataframe()
# df_avgquery[:3]
```

```
%%sql --module avg_rain_2015
SELECT *
FROM [or-h-159214:or_db.tbl_rain_temp_daily_2015] as wx
```

Feature engineering

Here I modify some of the features in the dataset to help with the machine learning.

- I extract hour, month, year and season from the timestamp data.
- I determine the weekday based on date.
- I categorize continuous data and create bins (not yet)

```
# see http://www.manishkurse.com/PythonProjects/BikeSharingDemand.html
df_avg_rain_2015 = bq.Query(avg_rain_2015).to_dataframe()
import datetime
from datetime import *
import calendar

df_avg_rain_2015['day'] = pd.to_datetime(df_avg_rain_2015['timestamp'], coerce=True) # This must be done
# df_avg_rain_2015['day'] = pd.to_datetime(df_avg_rain_2015['wt_date'], coerce=True) # This must be done

df_avg_rain_2015['weekday'] = df_avg_rain_2015['day'].dt.dayofweek
df_avg_rain_2015['yearday'] = df_avg_rain_2015['day'].dt.dayofyear
df_avg_rain_2015['month'] = df_avg_rain_2015['day'].map(lambda x: x.strftime('%m'))
df_avg_rain_2015['season'] = df_avg_rain_2015['day'].map(lambda x:(get_season(x)))

df_avg_rain_2015[:5]
```

```
INFO:oauth2client.client:Attempting refresh to obtain initial access_token
INFO:oauth2client.client:Refreshing access_token
```

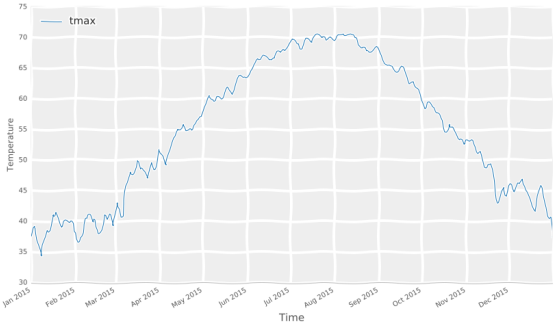
	timestamp	temperature	visibility	wind_speed	wind_gust	precipitation	snow_depth	day	weekday	yearday	month	season
0	2015-12-01	45.800573	6.561940	6.517354	4.314993	0.072531	0.625453	2015-12-01	1	335	12	winter
1	2015-12-03	45.190363	6.594378	6.281900	4.634983	0.065781	0.684361	2015-12-03	3	337	12	winter
2	2015-12-19	41.612213	6.322952	6.717726	5.598873	0.060877	0.841363	2015-12-19	5	353	12	winter
3	2015-01-01	37.581882	6.345000	6.846149	5.028181	0.043168	1.190530	2015-01-01	3	1	01	winter
4	2015-02-28	40.413339	6.671328	6.693237	4.356629	0.036585	2.029996	2015-02-28	5	59	02	winter

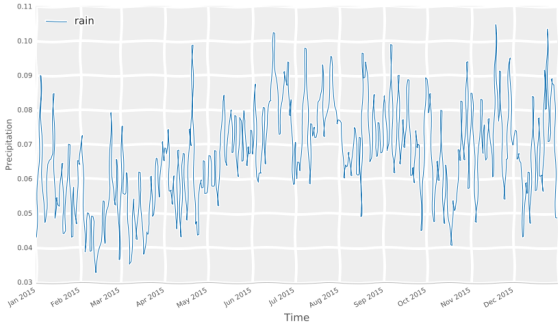
Exploratory Data Analysis

```
# see http://www.manishkurse.com/PythonProjects/BikeSharingDemand.html
# build the figure
df_avg_rain_2015_PLOT = df_avg_rain_2015.set_index(df_avg_rain_2015.day)
fig,axes = plt.subplots(figsize=(30, 8), nrows=1, ncols=2)

plt.sca(axes[0])
df_avg_rain_2015_PLOT['temperature'].plot(kind = 'line',label = 'tmax', color = colrcode[0])
plt.legend(loc='upper left',fontsize = 15)
plt.xlabel('Time',fontsize = 15)
plt.ylabel('Temperature')
prepare_plot_area(plt.gca())

plt.sca(axes[1])
df_avg_rain_2015_PLOT['precipitation'].plot(kind = 'line',label = 'rain', color = colrcode[0])
plt.legend(loc='upper left',fontsize = 15)
plt.xlabel('Time',fontsize = 15)
plt.ylabel('Precipitation')
prepare_plot_area(plt.gca())
```





Github commits data

- Contains a timeline of actions such as pull requests and comments on GitHub repositories with a flat schema. Created in May 2012.

Per day september 2015 pushes

```
%%sql --module daily_llvm_2012
SELECT
  *
FROM
    [or-h-159214:or_db.tbl_pushes_sept_2015] AS dpllvm
```

```
df_daily_llvm_2012 = bq.Query(daily_llvm_2012).to_dataframe()

df_daily_llvm_2012['day'] = pd.to_datetime(df_daily_llvm_2012['day'], coerce=True) # This must be done

df_daily_llvm_2012['weekday'] = df_daily_llvm_2012['day'].dt.dayofweek
df_daily_llvm_2012['yearday'] = df_daily_llvm_2012['day'].dt.dayofyear
df_daily_llvm_2012['month'] = df_daily_llvm_2012['day'].map(lambda x: x.strftime('%m'))
df_daily_llvm_2012['season'] = df_daily_llvm_2012['day'].map(lambda x:(get_season(x)))

# df_daily_llvm_2012 = df_daily_llvm_2012.set_index(df_daily_llvm_2012.day)
df_daily_llvm_2012.head()
```

```
INFO:oauth2client.client:Attempting refresh to obtain initial access_token
INFO:oauth2client.client:Refreshing access_token
```

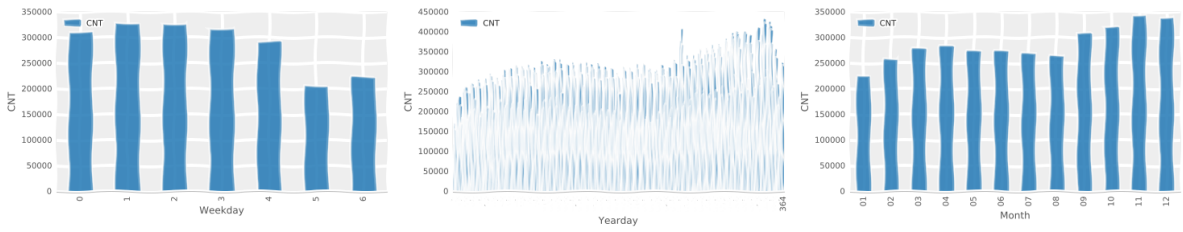
	day	type	cnt	weekday	yearday	month	season
0	2015-01-01	PushEvent	119241	3	1	01	winter
1	2015-01-02	PushEvent	169875	4	2	01	winter
2	2015-01-03	PushEvent	155315	5	3	01	winter
3	2015-01-04	PushEvent	169144	6	4	01	winter
4	2015-01-05	PushEvent	221202	0	5	01	winter

```
fig,axes = plt.subplots(figsize=(25, 4), nrows=1, ncols=3)
plt.sca(axes[0])
ts = df_daily_llvm_2012.groupby(['weekday'])['cnt'].mean()
ts.plot(kind = 'bar',x = [0,1,2,3],y = ts,color = colrcode[0],alpha = 0.6, label = 'CNT')
plt.xlabel('Weekday')
plt.ylabel('CNT')
prepare_plot_area(plt.gca())
plt.legend(loc = 'upper left')

plt.sca(axes[1])
ts = df_daily_llvm_2012.groupby(['yearday'])['cnt'].mean()
ts.plot(kind = 'bar',x = list(range(len(ts))),y = ts,color = colrcode[0],alpha = 0.6,label = 'CNT')
plt.xlabel('Yearday')
plt.ylabel('CNT')
prepare_plot_area(plt.gca())
plt.legend(loc = 'upper left')

plt.sca(axes[2])
ts = df_daily_llvm_2012.groupby(['month'])['cnt'].mean()
ts.plot(kind = 'bar',x = list(range(len(ts))),y = ts,color = colrcode[0],alpha = 0.6,label = 'CNT')
plt.xlabel('Month')
plt.ylabel('CNT')
prepare_plot_area(plt.gca())
plt.legend(loc = 'upper left')
```

```
<matplotlib.legend.Legend at 0x7fd6c0eecf50>
```



ML

Merge the two data sets

- By merging the weather and github datasets, we end up with the complete dataset to use for machine learning

```
merged_df=pd.merge(df_daily_llvm_2012, df_avg_rain_2015, on='day')
# merged_df = merged_df.set_index(merged_df.day)
merged_df.head()
```

	day	type	cnt	weekday_x	yearday_x	month_x	season_x	timestamp	temperature	visibility	wind_speed	wind_gust	precipitation	snow_depth	weekday_y	yearday_y	month_y	season_y
0	2015-01-01	PushEvent	119241	3	1	01	winter	2015-01-01	37.581882	6.345000	6.846149	5.028181	0.043168	1.190530	3	1	01	winter
1	2015-01-02	PushEvent	169875	4	2	01	winter	2015-01-02	38.966949	6.383479	6.682754	4.261686	0.059223	1.228399	4	2	01	winter
2	2015-01-03	PushEvent	155315	5	3	01	winter	2015-01-03	39.235535	6.161382	6.761690	5.234270	0.069616	1.161558	5	3	01	winter
3	2015-01-04	PushEvent	169144	6	4	01	winter	2015-01-04	38.881516	6.303655	7.028843	6.557914	0.090075	1.208168	6	4	01	winter
4	2015-01-05	PushEvent	221202	0	5	01	winter	2015-01-05	37.054700	6.631184	6.913143	6.118824	0.061689	1.281856	0	5	01	winter

DF must be scaled

```
# merged_df.apply(lambda x: (x - np.mean(x)) / (np.max(x) - np.min(x)))
from sklearn import preprocessing
# scaler = MinMaxScaler()
scaler = preprocessing.MinMaxScaler()

def scaleColumns(df, cols_to_scale):
    for col in cols_to_scale:
        df[col] = pd.DataFrame(scaler.fit_transform(pd.DataFrame(df[col])),columns=[col])
    return df

df_scaled = scaleColumns(merged_df,['temperature','cnt','visibility','precipitation','snow_depth'])
df_scaled.head()
```

	day	type	cnt	weekday_x	yearday_x	month_x	season_x	timestamp	temperature	visibility	wind_speed	wind_gust	precipitation	snow_depth	weekday_y	yearday_y	month_y	season_y
0	2015-01-01	PushEvent	0.000000	3	1	01	winter	2015-01-01	0.089811	0.171567	6.846149	5.028181	0.143186	0.554175	3	1	01	winter
1	2015-01-02	PushEvent	0.161756	4	2	01	winter	2015-01-02	0.127943	0.191611	6.682754	4.261686	0.366374	0.571909	4	2	01	winter
2	2015-01-03	PushEvent	0.115242	5	3	01	winter	2015-01-03	0.135337	0.075923	6.761690	5.234270	0.510852	0.540608	5	3	01	winter
3	2015-01-04	PushEvent	0.159421	6	4	01	winter	2015-01-04	0.125591	0.150031	7.028843	6.557914	0.795265	0.562435	6	4	01	winter
4	2015-01-05	PushEvent	0.325726	0	5	01	winter	2015-01-05	0.075297	0.320639	6.913143	6.118824	0.400649	0.596942	0	5	01	winter

View merged DF

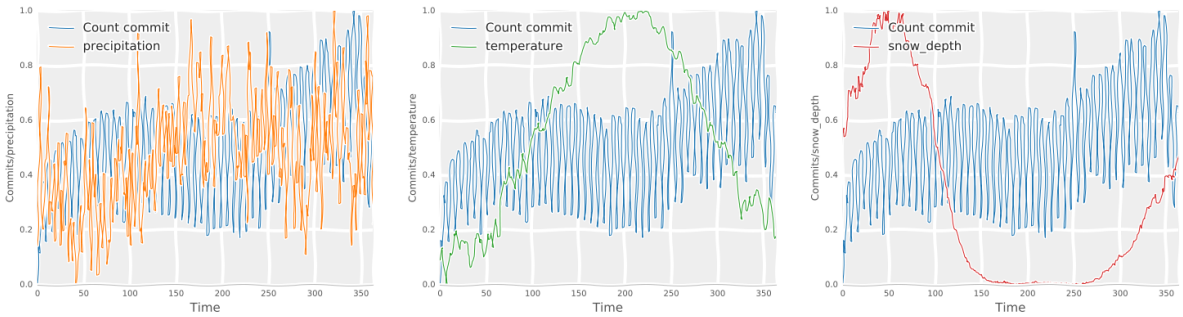
```
# merged_df = merged_df.set_index(merged_df.day)
# see http://www.manishkurse.com/PythonProjects/BikeSharingDemand.html
# build the figure
fig,axes = plt.subplots(figsize=(25, 6), nrows=1, ncols=3)

plt.sca(axes[0])
merged_df['cnt'].plot(kind = 'line',label = 'Count commit', color = colrcode[0])
merged_df['precipitation'].plot(kind = 'line',label = 'precipitation', color = colrcode[1])
plt.legend(loc='upper left',fontsize = 15)
plt.xlabel('Time',fontsize = 15)
plt.ylabel('Commits/precipitation')
prepare_plot_area(plt.gca())

plt.sca(axes[1])
merged_df['cnt'].plot(kind = 'line',label = 'Count commit', color = colrcode[0])
merged_df['temperature'].plot(kind = 'line',label = 'temperature', color = colrcode[2])
plt.legend(loc='upper left',fontsize = 15)
plt.xlabel('Time',fontsize = 15)
plt.ylabel('Commits/temperature')
prepare_plot_area(plt.gca())

plt.sca(axes[2])
merged_df['cnt'].plot(kind = 'line',label = 'Count commit', color = colrcode[0])
merged_df['snow_depth'].plot(kind = 'line',label = 'snow_depth', color = colrcode[3])
plt.legend(loc='upper left',fontsize = 15)
plt.xlabel('Time',fontsize = 15)
plt.ylabel('Commits/snow_depth')
prepare_plot_area(plt.gca())

# see good tip here https://cloud.google.com/blog/big-data/2017/02/four-seasons-in-one-post-using-google-bi
-to-explore-weather-effects-on-nyc
```



Correlation plot

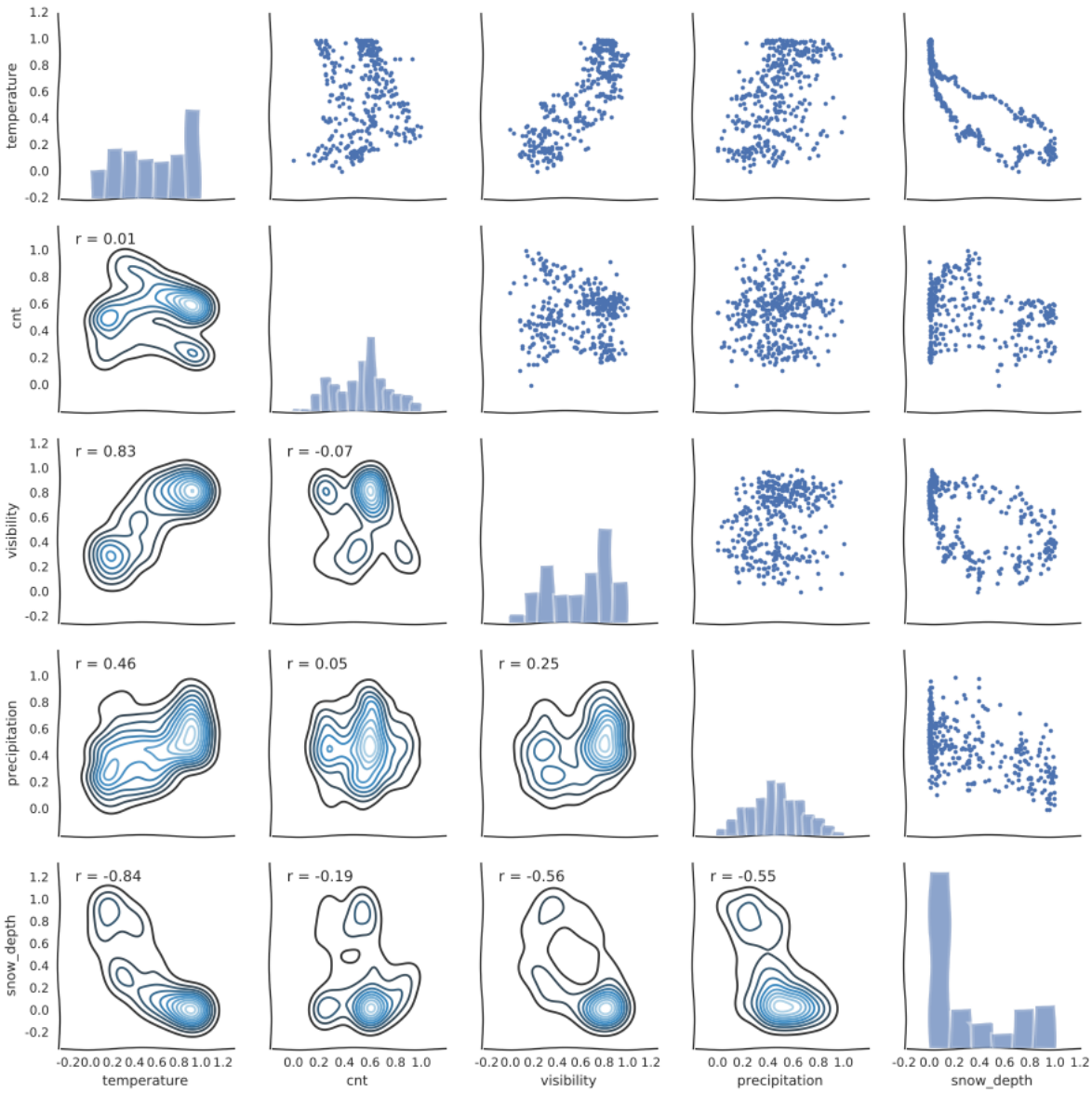
```
import numpy as np
from scipy import stats
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white")

df_corr = pd.DataFrame(merged_df, columns=['temperature','cnt','visibility','precipitation','snow_depth'])

def corrfunc(x, y, **kws):
    r, _ = stats.pearsonr(x, y)
    ax = plt.gca()
    ax.annotate("r = {:.2f}".format(r),
                xy=(.1, .9), xycoords=ax.transAxes)

g = sns.PairGrid(df_corr, palette=["red"])
g.map_upper(plt.scatter, s=10)
g.map_diag(sns.distplot, kde=False)
g.map_lower(sns.kdeplot, cmap="Blues_d")
g.map_lower(corrfunc)
```

```
<seaborn.axisgrid.PairGrid at 0x7fd6bea14d90>
```



Feature importances

Regression Models

Looking at how the weather variables effect puhses

```
import sys
import time
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn import cross_validation
from datetime import datetime, date, time
```

```

```