Impact Of Weather On NYC Subway Ridership

Intro to Data Science (Udacity) – Impact of Weather on NYC Subway Ridership

Author: Prema iyer (student id:iyerp1@gmail.com)
Last updated: July 29th 2014

Introduction

As you can see from http://www.pcac.org/wp-content/uploads/2012/04/subway-ridership-032212.pdf

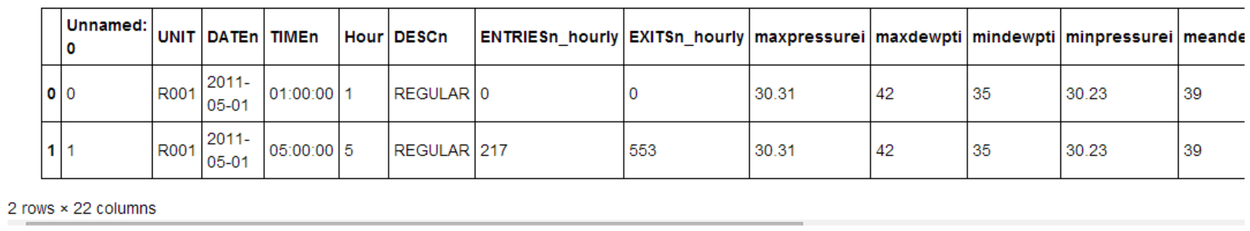
Many factors cause system-wide ridership to fluctuate:

- Seasonality
- Weather, such as unseasonable temperatures and precipitation
- Special events
- Students
- Timing of holidays
- Day of Week

In this article we will use data science tools and technologies to explore how and if the current weather conditions impact the number of people entering and exiting subway stations. Answering this question could help the Metro Transit Authority (MTA) to staff appropriately during peak hours. To accomplish this we will,

- Explore a large and complex set of data which is already cleaned up for us using data wrangling steps.
- Analysis of the data using appropriate statistics methods and machine learning techniques to test predictive models of data.
- Use data visualization technique to present the information gathered.
- Understand how MapReduce can help in processing larger amounts of data.

The data used in our analysis is stored in a csv format and has 22 columns, below are two sample rows

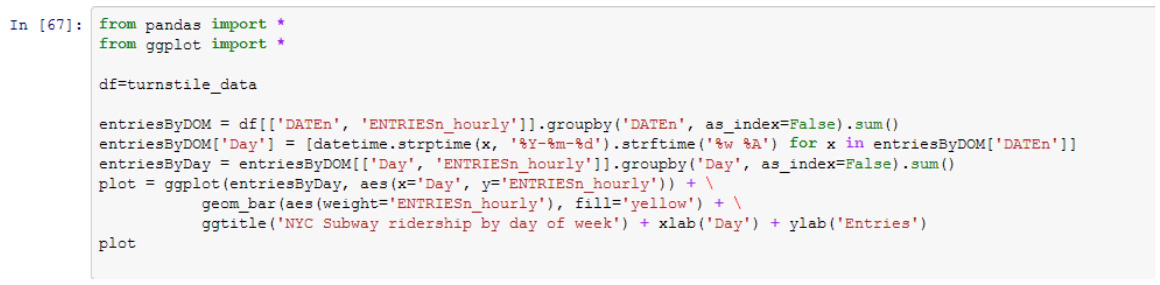


	Unnamed: 0	UNIT	DATEn	TIMEn	Hour	DESCn	ENTRIESn_hourly	EXITSn_hourly	maxpressurei	maxdewpti	mindewpti	minpressurei	meande
0	0	R001	2011-05-01	01:00:00	1	REGULAR	0	0	30.31	42	35	30.23	39
1	1	R001	2011-05-01	05:00:00	5	REGULAR	217	553	30.31	42	35	30.23	39

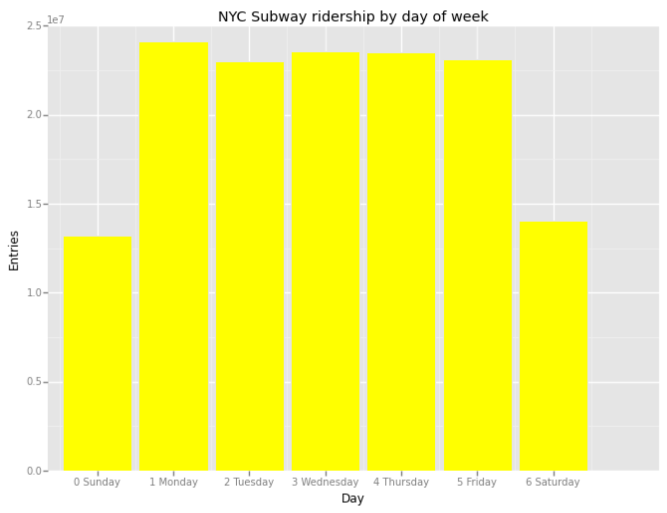
2 rows × 22 columns

Visualization and Exploratory Analysis of Data:

In the exploratory analysis we are trying to answer how the data looks like.
Visualization of data is NYC Subway ridership by day of week: -- How the day of the week affects ridership?



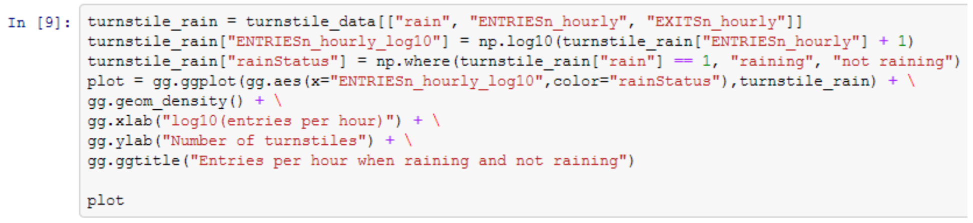
```
In [67]: from pandas import *
         from ggplot import *

         df=turnstile_data

         entriesByDOM = df[['DATEn', 'ENTRIESn_hourly']].groupby('DATEn', as_index=False).sum()
         entriesByDOM['Day'] = [datetime.strptime(x, '%Y-%m-%d').strftime('%w %A') for x in entriesByDOM['DATEn']]
         entriesByDay = entriesByDOM[['Day', 'ENTRIESn_hourly']].groupby('Day', as_index=False).sum()
         plot = ggplot(entriesByDay, aes(x='Day', y='ENTRIESn_hourly')) + \
                    geom_bar(aes(weight='ENTRIESn_hourly'), fill='yellow') + \
                    ggtitle('NYC Subway ridership by day of week') + xlab('Day') + ylab('Entries')
         plot
```



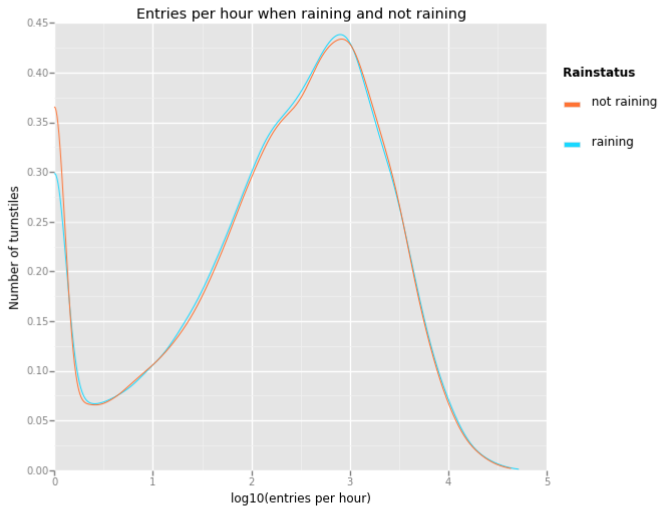
← *Saturday and Sunday have low ridership*

By looking at the data in its entirety what effect does rain have on the ridership of the subway? In our data “rain" column has 1 for raining and 0 if it was not, in the hourly turnstile measurement.

With the x-axis as "log10" (log base 10) of Entries by hour (This is to accommodate the entire dataset and still visualize some clarity compared to the "Entries/Exits per Time of Date".) and Y-axis as number of turnstiles



```
In [9]: turnstile_rain = turnstile_data[["rain", "ENTRIESn_hourly", "EXITSn_hourly"]]
        turnstile_rain["ENTRIESn_hourly_log10"] = np.log10(turnstile_rain["ENTRIESn_hourly"] + 1)
        turnstile_rain["rainStatus"] = np.where(turnstile_rain["rain"] == 1, "raining", "not raining")
        plot = gg.ggplot(gg.aes(x="ENTRIESn_hourly_log10",color="rainStatus"),turnstile_rain) + \
        gg.geom_density() + \
        gg.xlab("log10(entries per hour)") + \
        gg.ylab("Number of turnstiles") + \
        gg.ggtitle("Entries per hour when raining and not raining")

        plot
```



When you look at the graph there is not a whole lot of difference between numbers of people using the subway when raining compared to when it is not. Even though our numeric estimate below indicates that mean and median ridership are higher when it is raining than when it is not, this conclusion isn't clear when looking at the graph. The ridership pattern doesn’t seem to change according to rain.

```
In [7]: turnstile_rain.groupby("rainStatus").describe()
```

```
Out[7]:
```

		rain	ENTRIESn_hourly	EXITSn_hourly	ENTRIESn_hourly_log10
rainStatus					
not raining	**count**	87847	87847.000000	87847.000000	87847.000000
	mean	0	1090.278780	883.259610	2.215524
	std	0	2320.004938	1998.516762	1.110371
	min	0	0.000000	0.000000	0.000000
	25%	0	38.000000	31.000000	1.591065
	50%	0	278.000000	231.000000	2.445604
	75%	0	1111.000000	846.000000	3.046105
	max	0	43199.000000	45249.000000	4.635484
raining	**count**	44104	44104.000000	44104.000000	44104.000000
	mean	1	1105.446377	894.123572	2.234901
	std	0	2370.527674	2028.552487	1.093296
	min	1	0.000000	0.000000	0.000000
	25%	1	41.000000	33.000000	1.623249
	50%	1	282.000000	235.000000	2.451786
	75%	1	1103.250000	849.000000	3.043067
	max	1	51839.000000	41503.000000	4.714665

16 rows × 4 columns

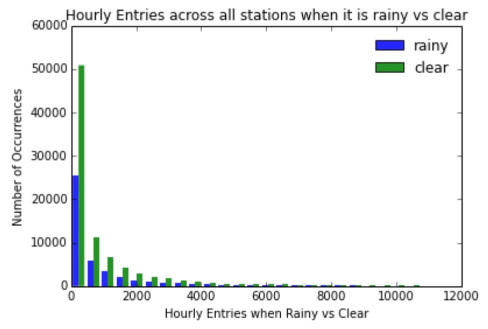
The numeric difference however is very small, as shown above in the plot of the probability distribution function (PDF) of our data. We have to verify if statistically the distributions of both groups are identical. ***Let us further analyze our data using statistical methods.***

Statistical Analysis of Data

Before trying to analyze using a statistical method, it might be useful to take a look at the data we're hoping to analyze so that we can select the appropriate statistical method. More specifically, let us examine the hourly entries in our NYC subway data and determine what distribution the data follows.

Below histograms on the same axes, shows hourly entries when raining vs. when not raining.

```
import matplotlib.pyplot as plt
import scipy.stats as sstat
plt.figure()
rainy = turnstile_data[turnstile_data['rain']==1.0]['ENTRIESn_hourly']
clear = turnstile_data[turnstile_data['rain']==0.0]['ENTRIESn_hourly']

limit = 99
rainy_99 = sstat.scoreatpercentile(rainy, limit)
clear_99 = sstat.scoreatpercentile(clear, limit)
rainy = rainy[rainy<=rainy_99]
clear = clear[clear<=clear_99]

plt.hist([rainy.values,clear.values], bins=25, alpha=0.85, ec='None', label=['rainy', 'clear'])
plt.legend(frameon=False)
plt.ylabel("Number of Occurrences")
plt.xlabel("Hourly Entries when Rainy vs Clear")
plt.title("Hourly Entries across all stations when it is rainy vs clear")
plt
```



Since the data is not normally distributed we can use the Mann-Whitney U test as it doesn't assume the data is normally distributed to answer the above question.

```
rainy = turnstile_data[turnstile_data['rain']==1.0]['ENTRIESn_hourly']
clear = turnstile_data[turnstile_data['rain']==0.0]['ENTRIESn_hourly']
(u, pvalue) = scipy.stats.mannwhitneyu(clear,rainy)
pvalue*=2
print "mean entries per hour when not raining: %s" % clear.mean()
print "mean entries per hour when raining: %s" % rainy.mean()
print "U={}, pvalue={}".format(U,pvalue)
```

```
mean entries per hour when not raining: 1090.27878015
mean entries per hour when raining: 1105.44637675
U=89.0, pvalue=0.0386192688276
```

The p-value (since Mann-Whitneyu test returns 1 sided p-value we have multiplied by 2) is $0.038 < 0.05$.
So a low p value indicates greater statistical significance and stronger evidence for rejecting the hypothesis that there is no difference between the data sets we are comparing. This means our initial assumption that ridership increases while raining isn't statistically correct.
(*For more information about null hypothesis and p-value check in the NYCRidership-Appendix*).

Use Machine Learning to predict ridership

So far we have used statistical method to analyze the existing data in hand and drawn conclusions from it. Now we turn to machine learning techniques to predict the NYC subway ridership given the different features in play. The task of predicting a continuous variable like number of entries per hour, etc. is called regression. We are going to use linear regression to predict the number of entries per hour using the following features

Raining or not raining	→ rain
The amount of rain falling	→ precipi
The hour of the day	→ Hour
The mean temperature	→ meantempi
Metro station	→ UNIT (to represent a categorical variable we use dummy variables)

Using the Linear regression by gradient descent method with the above list of features we get a coefficient of determination of 46%.

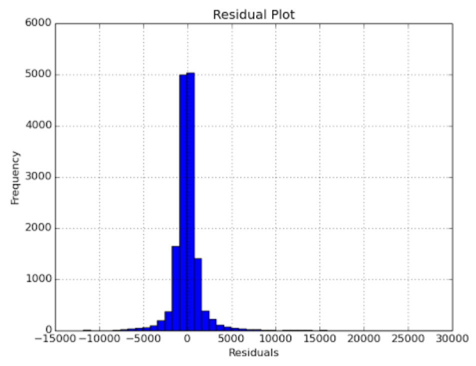
```
Your r^2 value is 0.461129645866
```

← 46%

The **coefficient of determination**, denoted R^2 or r^2 and pronounced **R squared**, indicates how well data fit a statistical model.
It provides a measure of how well observed outcomes are replicated by the model, as the proportion of total variation of outcomes explained by the model.

To determine how good our statistical model is to plot its residuals, which are defined as the differences between the predicted and actual values. Residuals are elements of variation in the data unexplained by a model. A good model's residual plot will be distributed as a Normal distribution with a mean of 0 and some finite variance Residual Plot (more about residual plot in appendix)

```
p-value of normaltest on residuals: 0.0
mean of residuals: 0.444605608728
The image produced by your code is shown below:
Can you infer how well our linear regression model performed from this histogram?
```

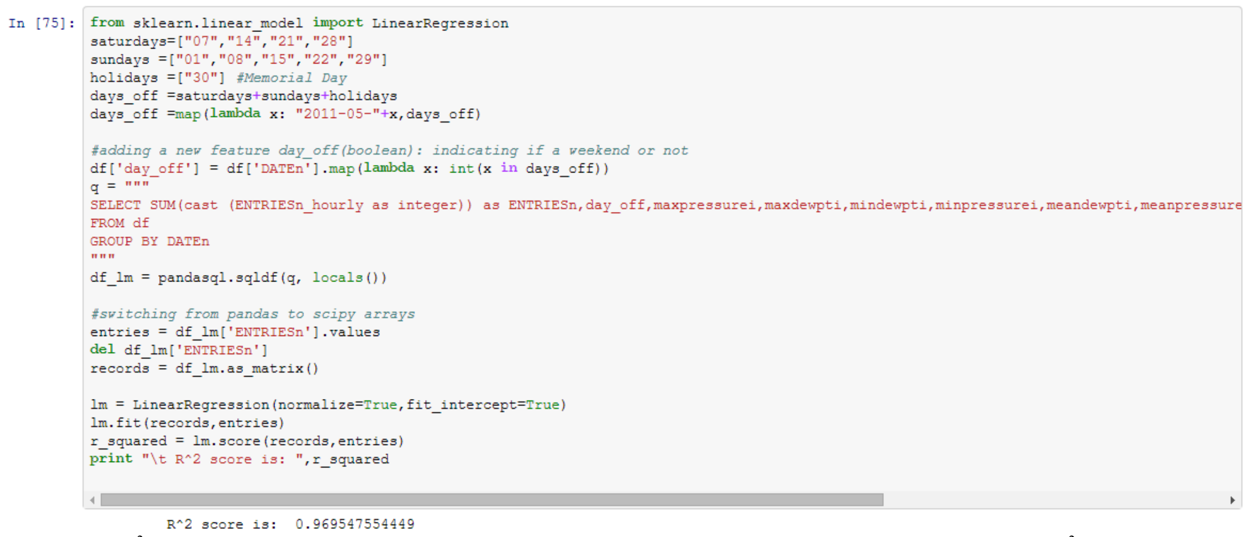


← *** Linear regression and residual plot code are in NYCRidership-Appendix.pdf code section.*

Our current model seems to be quite a good fit and the residual plot indicates that there might be some structure left in our residual errors that could be reduced.

How can the model be improved?

With the outcome variable still the entries per day, with the variables being the weather variables and the date. Since we previously observed that weekends had low ridership, If we add a new feature "*whether it is a weekend or not"* and compute R$^{2.}$, we get an R^2 of 0.969 as you can see from below



```
In [75]: from sklearn.linear_model import LinearRegression
         saturdays=["07","14","21","28"]
         sundays =["01","08","15","22","29"]
         holidays =["30"] #Memorial Day
         days_off =saturdays+sundays+holidays
         days_off =map(lambda x: "2011-05-"+x,days_off)

         #adding a new feature day_off(boolean): indicating if a weekend or not
         df['day_off'] = df['DATEn'].map(lambda x: int(x in days_off))
         q = """
         SELECT SUM(cast (ENTRIESn_hourly as integer)) as ENTRIESn,day_off,maxpressurei,maxdewpti,mindewpti,minpressurei,meandewpti,meanpressure
         FROM df
         GROUP BY DATEn
         """
         df_lm = pandasql.sqldf(q, locals())

         #switching from pandas to scipy arrays
         entries = df_lm['ENTRIESn'].values
         del df_lm['ENTRIESn']
         records = df_lm.as_matrix()

         lm = LinearRegression(normalize=True,fit_intercept=True)
         lm.fit(records,entries)
         r_squared = lm.score(records,entries)
         print "\t R^2 score is: ",r_squared

                   R^2 score is:  0.969547554449
```

Even though R^2 alone is quite insufficient and the data we are studying is just for one month we have improved R^2 as you can see from above. That being said, increased R^2 does not always mean a good fit, we need to base our analysis on much larger data and consider all possible scenarios to predict accurately. One limitation of linear regression is that we must restrict our interpretation of the model to the range of values of the predictor variables that we observe in our data. We cannot assume this linear relation continues outside the range of our sample data.

We can use ***Polynomial regression*** which models a non-linear relationship between the independent and dependent variables (technically, between the independent variable and the conditional mean of the dependent variable). Compared to `0.461129645866` obtained using Gradient descent, we have improved R^2 with Polynomial Regression $\rightarrow$ `by adding more of existing features with polynomial combinations R^2: 0.475`

But the downside to this is it will increase the complexity of our model as we feed in more data and this is where Map Reduce comes in play were we need to gather and process large amounts of data in concurrency.

Map Reduce Model

In these situations where we need analysis of big data we can use Map Reduce model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. The main purpose is to take all the data that is far too big to fit in a single machine and aggregate it down or reduce it down to a point where it is small enough to look at it in a single machine or make a reduced plot.

Today other tools provide functions on a higher level to perform these steps (like Pig); it abstracts away from user how the implementation of calculation is done. So we can write in abstract language as to what kind of operations need to be performed on the data rather than how to do it and Pig will decide the fastest way of doing it for us.

The data exploration directly with Hadoop MapReduce would also had been somewhat harder, but today there are a number of data exploration tools with Hadoop connectors (such as Tableau, also used for this study) that make this step much easier.

*** For further information about Map Reduce and polynomial regression please check the NYCRidership-Appendix.pdf*

Summary:

We analyzed the impact of various weather elements on NYC Subway ridership using statistical method and predicted the extent to which each of the features impact ridership using machine learning techniques. But the data at hand alone is not sufficient to determine the NYC Subway Ridership pattern. Because we were basing our predictions and conclusion only on one month of data we need to record data for more months and for different seasons to accurately predict the impact of weather on NYC Ridership. This article is a brief synopsis of how data science knowledge and tools can help us predict occurrences of events and understand and draw meaningful conclusions of the huge amount of data that is available out there.