Analyzing the NYC Subway Dataset

October 10, 2015

- Author: Ren Zhang
- email: zhang_ren@bentley.edu

0.1 Reference

1. Dataset:
   - I am using the improved data set downloaded from the Udacity website. A link to a description of the variables in the data set
2. Forum Posts:
   - Mann-Whitney U Test on improved dataset yields p=NaN?
   - Project Mann-Whitney U Test p-value
3. Wikipedia Articles:
   - Mann–Whitney U test
4. Book:
   - Elements of Statistical Learning: data mining, inference and prediction by Hastie, Tibshirani and Friedman

0.2 Statistical Test

The variable we are interested in analyzing is `ENTRIESn_hourly` in the dataset. We want to investigate whether the `ENTRIESn_hourly` on rainy days and the `ENTRIESn_hourly` on days without rain are drawn from the same population.

Judging from the two histograms(see Appendix) of `ENTRIESn_hourly` in these two situations, we find out that the distributions are not normal. We decide to perform a one-tail Mann-Whitney U test, because this test does not make any assumptions on distributions whereas t-test can only be applied on normal distributions.

- The null hypothesis is: the mean value of `ENTRIESn_hourly` on rainny days is the same as that on days without rain.
- The alternative hypothesis is: the mean value of `ENTRIESn_hourly` on rainny days is higher than that on days without rain.

The mean value of `ENTRIESn_hourly` on rainny days is 2028.1960 and the mean value of `ENTRIESn_hourly` on days without rain is 1845.5394. The U value is 153635120.5 and the associated p value is 5.4827e-06. The p value is very small, even if we use a very small critical value 0.0001, we can still reject the null hypothesis and conclude that the mean value of `ENTRIESn_hourly` on rainny days is significantly higher than that on days without rain.

0.3 Linear Regression

We used lasso regression to perform feature selection and model fitting at the same time.

> The Lasso is a shrinkage and selection method for linear regression. It minimizes the usual sum of squared errors, with a bound on the sum of the absolute values of the coefficients.

Lasso regression uses a l_1 penlty that has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large. Thus, in the resulting model, only subset of the features are used.

Since lasso will perform feature selection for us, we can put as many features as possible into the fitting. The features entered to fit a lasso regression model are: `precipi`, `pressurei`, `tempi`, `wspdi`, `meanprecipi`, `meanpressurei`, `meantempi`, `meanwspdi`, `hour`, `day_week`, `weekday`, `conds`, `rain`, `fog` and `UNIT`. Among them, we performed dummy variable transformations on `hour`, `day_week`, `weekday`, `conds`, `rain`, `fog` and `UNIT`, becasue these variables are categorical rather than numerical.

We fitted the lasso regression model using a tuning parameter of 0.2 and the resulting R^2 value is 0.5125. The features selected in the model and the associated coefficients are shown in the Appendix. Unfortuntly, only the dummy variables remained in the model.

The R^2 value 0.5125 means that 51.25% of variability in the dependent variable `ENTRIESn_hourly` in the sample we have is explained by the lasso regression model. This R^2 value is moderate and the model is not appropriate for making perdictions on ridership. Also notice that the R^2 is not an estimate for the whole population, so making perdictions using this model for out of sample data points might be even more inaccurate.

0.4 Visualization

The first visualization is the histograms of the dependent variable `ENTRIESn_hourly` on rainny days and on days without rain (see Appendix for the plots). Judging from the histograms, the distributions are not normally distributed but rather following a power law distribution.

The second visualization is a bar plot of the average value of the dependent variable `ENTRIESn_hourly` faceted by the day of week. We can clearly see that the ridership during weekend is much less than the ridership during weekdays.

0.5 Conclusion

Based on the result of the statistical test, we conclude that more people ride the NYC subway when it is raining than when it is not raining. If indeed there is no difference in ridership between rainny days and days without rain, the chance that we obtain the data we have in hand is too small that no reasonable people would attribute that to chance of sampling.

Since the lasso regression model we built did not include the variable `rain` as a explanatory variable, so the previous conclusion we have is based solely on the statistical test. Also, we construced a random forest regression model and looked at the feature importance provided by the model (see Appendix), `rain` is not among the 20 most important variabels.

0.6 Reflection

One potential shortcomming of this analysis is that, I did not perform any test about the outliers. Outliers may have great impact on model fitting.

The second thing is that, R^2 is not a good measure when comparing models with different numbers of features. Adding more explanatory variables into a model will never causing the R^2 value to decrease.

0.7 Appendix Python Code for the project

0.7.1 Load libraries and set ipython notebook inline plotting

```
In [1]: import os
        import pandas as pd
```

```
import numpy as np
import scipy
from sklearn import linear_model
from sklearn.metrics import r2_score
from sklearn.ensemble.forest import RandomForestRegressor
from ggplot import *
%matplotlib inline
```

0.7.2 Read in data set

```
In [2]: filename = "turnstile_weather_v2.csv"
        path = "d:\GithubRepos\Udacity\P2"
        df = pd.read_csv(os.path.join(path,filename))
```

0.7.3 Statistical Test

get *ENTRIESn_hourly* in two situations

```
In [3]: rain_ENTRIESn_hourly = df[df["rain"] == 1]["ENTRIESn_hourly"]
        nonrain_ENTRIESn_hourly = df[df["rain"] == 0]["ENTRIESn_hourly"]
```

Calculating the mean values of *ENTRIESn_hourly* in rainy days and in days without rain.

```
In [4]: print "rainny days    |  days without rain"
        print np.mean(rain_ENTRIESn_hourly)," | ", np.mean(nonrain_ENTRIESn_hourly)
```

```
rainny days    |  days without rain
2028.19603547  |  1845.53943866
```

There is strange behavior of the `mannwhitneyu()` function from the scipy library when used in windows machines against the improved data set. This is discussed in a couple of posts (see the Reference part). I used the formula for p value as described in the wikipeida page to calculate it.

```
In [5]: U, _ = scipy.stats.mannwhitneyu(rain_ENTRIESn_hourly, nonrain_ENTRIESn_hourly)
        mu_u = len(rain_ENTRIESn_hourly)*len(nonrain_ENTRIESn_hourly)/2
        sigma_u = np.sqrt(len(rain_ENTRIESn_hourly)*len(nonrain_ENTRIESn_hourly)*
                          (len(rain_ENTRIESn_hourly)+len(nonrain_ENTRIESn_hourly)+1)/12)
        z = (U - mu_u)/sigma_u
        p = 2*scipy.stats.norm.cdf(z)
        print "U value      |  p value"
        print U," | ", p
```

```
U value      |  p value
153635120.5  |  5.48269387142e-06
```

0.7.4 Linear regression

Prepare the features and training labels.

```
In [6]: features_start_with = ["precipi","pressurei","tempi","wspdi",
                               "meanprecipi","meanpressurei","meantempi","meanwspdi",]

        categorical_features_to_dummies = ["hour","day_week","weekday","conds","rain","fog","UNIT"]

        X = df[features_start_with]

        for feature in categorical_features_to_dummies:
```

```
        dummy = pd.get_dummies(df[feature], prefix = feature)
        X = X.join(dummy)

    y = df['ENTRIESn_hourly']
```

Fit lasso regression model

```
In [7]: clf = linear_model.Lasso(alpha=0.2, fit_intercept = True, normalize = True)
        clf.fit(X,y)
```

```
Out[7]: Lasso(alpha=0.2, copy_X=True, fit_intercept=True, max_iter=1000,
           normalize=True, positive=False, precompute=False, random_state=None,
           selection='cyclic', tol=0.0001, warm_start=False)
```

Make prediction on the sample data and calculate the R Squared value

```
In [8]: y_pred = clf.predict(X)
        r2_score(y,clf.predict(X))
```

```
Out[8]: 0.51248812846310254
```

See the variables used in the model as well as the coefficients

```
In [9]: names = X.columns[clf.coef_ > 0.01]
        coefficients = clf.coef_[clf.coef_ > 0.01]
        row_name, coef_value = "",""
        for i in range(len(names)):
            row_name += names[i].rjust(12)
            coef_value += str(round(coefficients[i],5)).rjust(12)
            if len(row_name) > 80:
                print row_name
                print coef_value
                print
                row_name, coef_value = "",""
```

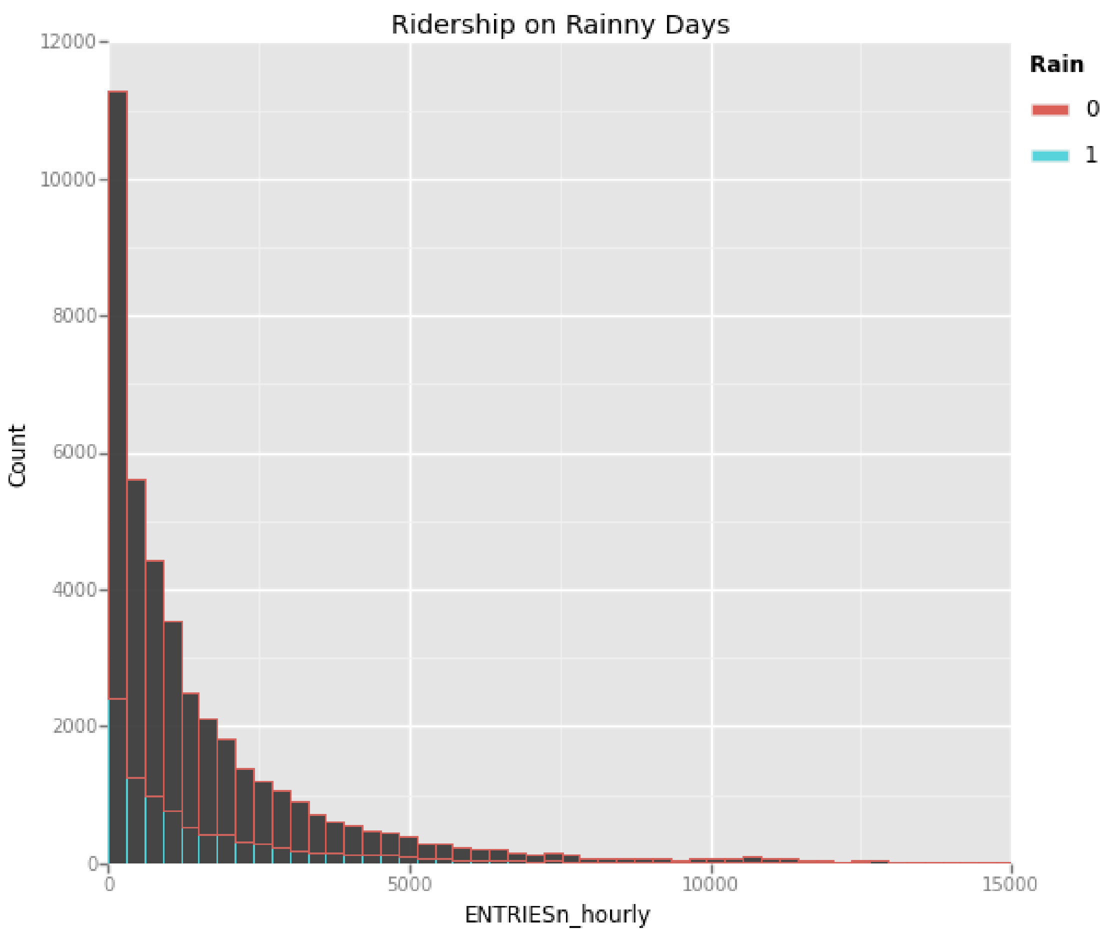
```
hour_12      hour_20   day_week_3  conds_Clear    UNIT_R011    UNIT_R012    UNIT_R013
   705.82337    912.87137      0.75025     42.73925   5401.16449   6753.92547     652.3663

   UNIT_R017    UNIT_R018    UNIT_R019    UNIT_R020    UNIT_R021    UNIT_R022    UNIT_R023
  2267.36089   5913.86907    1398.4282   4443.45774   2752.76777   7587.85015   4223.02754

   UNIT_R024    UNIT_R025    UNIT_R027    UNIT_R029    UNIT_R030    UNIT_R031    UNIT_R032
  1358.29953   3495.32068   1037.46312    5299.4362   1169.62972   2421.39851   2515.19138

   UNIT_R033    UNIT_R035    UNIT_R041    UNIT_R043    UNIT_R044    UNIT_R046    UNIT_R049
  6304.40921    866.38262   1169.44144     956.5758   2747.98437   6414.43595    842.24775

   UNIT_R050    UNIT_R051    UNIT_R053    UNIT_R055    UNIT_R057    UNIT_R062    UNIT_R080
  2093.80474   3203.98963    1308.7658   6473.91472   2952.37678    806.23692   1681.54336

   UNIT_R081    UNIT_R083    UNIT_R084    UNIT_R085    UNIT_R086    UNIT_R092    UNIT_R093
  1628.34557   1195.95191   8100.22607    678.76364    669.54322    122.42588    155.24393

   UNIT_R095    UNIT_R096    UNIT_R097    UNIT_R099    UNIT_R101    UNIT_R102    UNIT_R105
   312.92751    499.53521   1122.90662    461.12943    895.52192    1784.3606    1434.0595
```

UNIT_R108	UNIT_R111	UNIT_R116	UNIT_R119	UNIT_R122	UNIT_R127	UNIT_R137
3323.22618	1320.34981	1299.49494	4.28494	727.18736	2902.11328	619.82327
UNIT_R139	UNIT_R163	UNIT_R172	UNIT_R179	UNIT_R188	UNIT_R194	UNIT_R198
635.79067	1437.84983	3.88743	4884.6616	436.89373	91.82522	219.21858
UNIT_R202	UNIT_R207	UNIT_R208	UNIT_R211	UNIT_R218	UNIT_R223	UNIT_R235
401.42173	112.64656	684.90573	522.81744	165.34377	303.48108	688.61001

0.7.5 Visualization

create histogram for *ENTRIESn_hourly* on rainy days and on days without rain.

```
In [11]: ggplot(aes(x="ENTRIESn_hourly", color = "rain") , data = df) + \
             geom_histogram(binwidth = 300, alpha = 0.9) + \
             ylab("Count") + \
             ggtitle("Ridership on Rainny Days") + \
             xlim(0,15000)
```

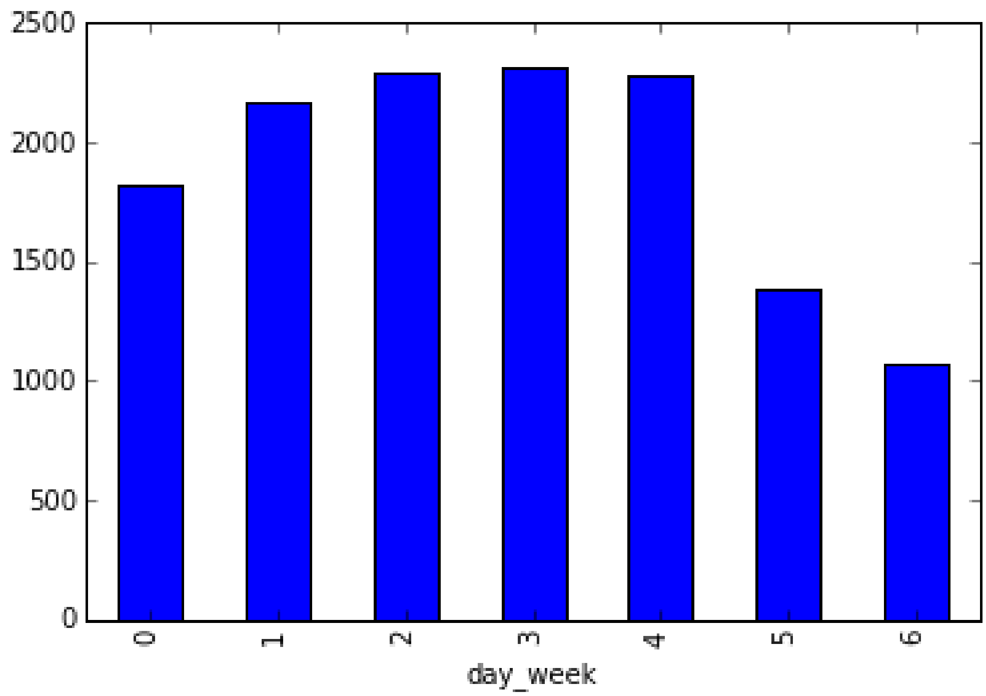



```
Out[11]: <ggplot: (21003019)>
```

Both distributions are more likly following the power law distribution rather than normal distribution.

```
In [12]: grouped = df.groupby("day_week")
         avg_entri_by_weekday = grouped["ENTRIESn_hourly"].mean()
         avg_entri_by_weekday.plot(kind = "bar")
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x142b3da0>
```



The two bars on the right hand side are the average value of *ENTRIESn_hourly* on saturday and sunday. The bars are relatively shorter than the bars on weekdays.

```
In [13]: RF_model = RandomForestRegressor(random_state=0, n_estimators=10)
         RF_model.fit(X,y)
```

```
Out[13]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=10, n_jobs=1, oob_score=False, random_state=0,
                    verbose=0, warm_start=False)
```

The most important 20 features based on random forest model.

```
In [15]: names = X.columns[RF_model.feature_importances_ >= np.sort(RF_model.feature_importances_)[::-1]
         print names
```

```
Index([u'meantempi', u'hour_0', u'hour_4', u'hour_8', u'hour_12', u'hour_16',
       u'hour_20', u'weekday_0', u'weekday_1', u'UNIT_R011', u'UNIT_R012',
       u'UNIT_R018', u'UNIT_R020', u'UNIT_R022', u'UNIT_R023', u'UNIT_R029',
       u'UNIT_R033', u'UNIT_R046', u'UNIT_R055', u'UNIT_R084', u'UNIT_R179'],
      dtype='object')
```