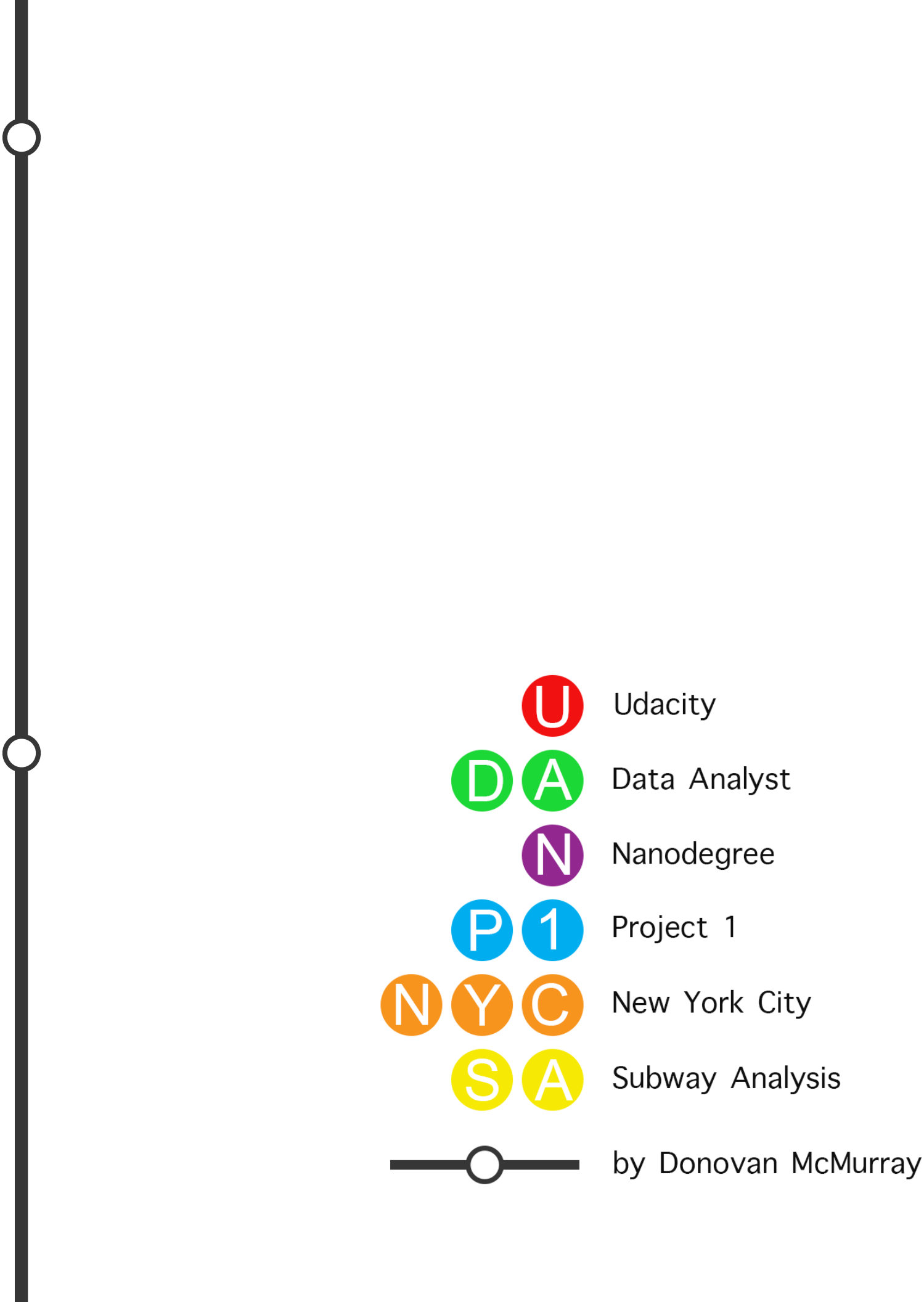
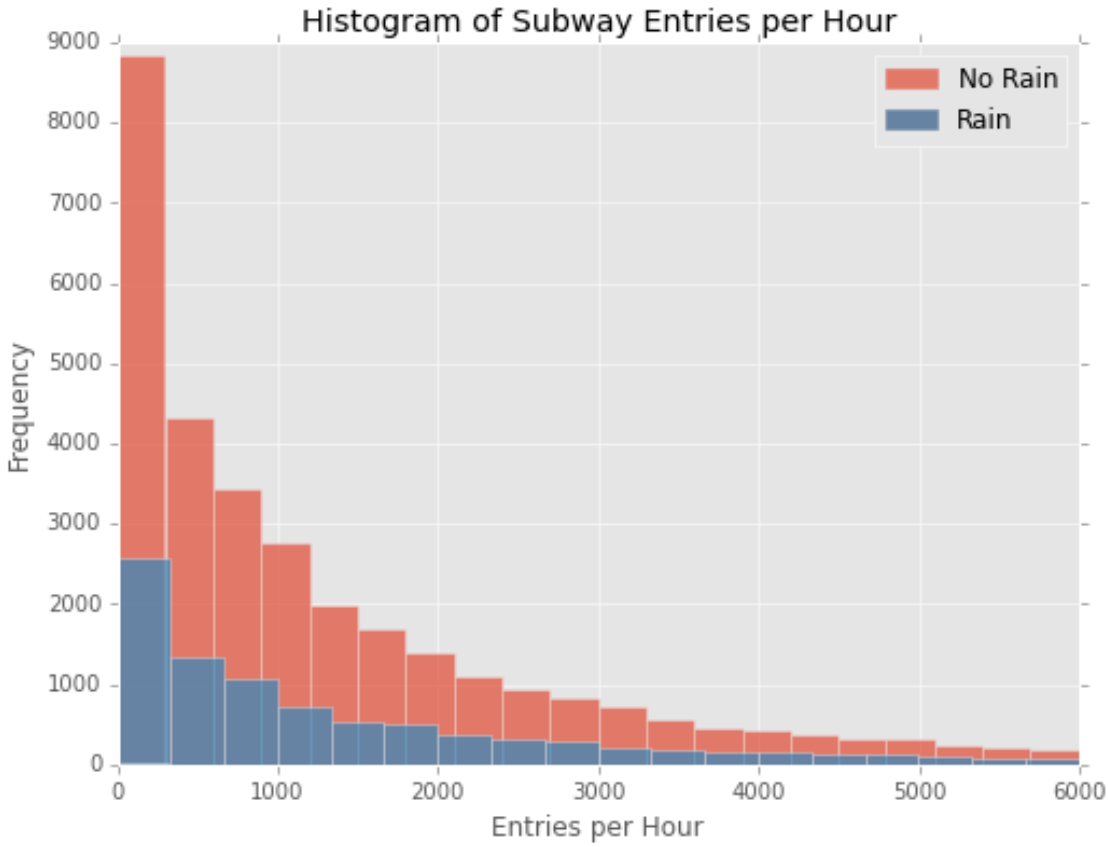

U
Udacity
D A
Data Analyst
N
Nanodegree
P 1
Project 1
N Y C
New York City
S A
Subway Analysis
by Donovan McMurray

Statistical Test

Data



This figure shows the histogram of the number of subway entries in a given hour according to the NYC MTA subway data. The data is split up into two populations based on whether it rained or not that day. The red bars represent entries per hour on days that it didn't rain, and the blue bars represent entries per hour on days when rain was detected. Bins over 6000 were omitted for visualization clarity, since the tails of these distributions are very long.

Method

By looking at it, the clear days look like they have more ridership than rainy days, but the histogram is misleading because the number of rainy days is less than the number of clear days. What we can tell from this histogram is that both of the populations have

the same underlying distribution. To test whether there's an actual difference between the two populations, we need to perform a statistical test on the populations.
The question to ask is "is this statistically significant?" We can be confident that the answer to this question is yes if we find a p-value of 0.05 or less. We will use the Mann Whitney U Test to perform a two-tailed p-test where the null hypothesis (H_0) is $\mu_{nr} = \mu_r$.

Rationale

To test null hypothesis, I used the two-tailed Mann Whitney U Test. The reason for using this statistical test is that two populations meet the following assumptions:

1. The observations are independent and identically distributed.
2. The data are ordinal since they're counts of entries per hour.
3. The data of both populations come from the same distribution. In this case, they're from a Poisson distribution since they're counts of independent events in hour-long time intervals. If the data had been normally distributed, we could have performed a t-test, but since they're not, we can use the non-parametric Mann Whitney U test instead.
4. Since there's no a priori reason to believe on of the population means is greater than or less than the other, a two-tailed test will be used.

Results

The mean number of entries per hour on non-rainy days is $\bar{x}_{nr} = 1090.28$, and the mean on rainy days is $\bar{x}_r = 1105.45$. Putting both sets of data through scipy's implementation of the Mann-Whitney U Test, we get a U statistic of 1,924,409,167.0 and a p-value of 0.04999982559. The no rain sample has 87847 data points, and the rain sample has 44104, so the maximum U value is the product of the magnitudes, or 3,874,404,088. Divide by 2 to get $m_u = 1{,}937{,}202{,}044$.

Conclusion

The p-value corresponds to the probability of getting a more extreme U statistic than m_u. In this case, the p-value is 0.04999982559, meaning that a more extreme U statistic than the one our data resulted in has a 4.999982559% chance of occurring. Since our critical p-value is 0.05, the statistical test ever so slightly passes our significance threshold and H_0 can be rejected.

Linear Regression

Method

After determining that there is a statistical difference in NYC subway ridership on rainy versus non-rainy days, a good next step is to come up with a predictive model so that we can use this knowledge to make decisions. For instance, if I could predict that a large number of people will be using the subway tomorrow based on the weather forecast, I might decide to ride my bike instead.

For my predictive model, I generated a linear regression model using ordinary least squares (OLS). OLS was used instead of the less computationally-demanding gradient descent because with the number of samples and dimensionality of the data, OLS was not computationally prohibitive, and in such cases it's preferred because OLS will find the absolute minimum whereas gradient descent can succumb to local minima.

Features

After choosing a modeling method, it's necessary to choose the best features on which to base your model. The r-squared value of your linear regression is a representation of the "goodness of fit" your model has to the data. R-squared values range from 0 (the model describes none of the variance in the data) to 1 (the model describes all of the variance in the data).

I started out with a very basic set of features, FS1, which used the 'rain' feature, a binary value for whether or non it rained at all that day, and the 'precipi' feature, which represents the amount of rain at the time and location of the subway data point.

> FS1: rain, precipi
> r^2 = 0.0021424885693342999

This model does not explain much of the variance in the data, so surely we can do better. One important feature to account for are the turnstile units themselves. Since this data is categorical, we will need to use the pandas function `get_dummies` to break out the 'UNIT' feature into a separate column for each possible category. For simplicity, I will still refer to 'UNIT' as a single feature. The same holds for the 'conds' feature, which is a

categorical description of the day's weather, such as "Clear" or "Cloudy". So, adding the 'UNIT' feature, we are able to improve our r-squared value greatly.

FS2: rain, precpi, UNIT
r^2 = 0.37807540408609885

Even still, we can improve r-squared further by accounting for the hour of the day and the mean temperature for that day in our model.

FS3: rain, precpi, UNIT, hour, meantempi
r^2 = 0.47924770782

I tried a few more things and was able to push r-square slightly higher. One approach I tried was to use all the remaining available weather features on which to build my model.

FS4: rain, precpi, UNIT, hour, meantempi, day_week, weekday, latitude, longitude, conds, fog, pressurei, tempi, wspdi, meanprecipi, meanpressurei, meantempi, meanwspdi, weather_lat, weather_lon
r^2 = 0.4928934458626717

We now have a model that explains nearly half of the variance in our data, which is the best r-squared value so far. However, using all of these features increases the risk of overfitting our model to our data. Also, as more data is gathered, using more features will increase our computation load.

So the next thing I did was to use a random forest to determine which features were the most important predictors. I used SciKit Learn's ExtraTreesClassifier to accomplish this. See *Appendix: Documentation* for more information on ExtraTreesClassifier.

Due to the complexity of the computation, I had to sample 20% of the total dataset to run this classifier, but the results were still useful in determining the importance of each feature. There is a big drop off in importance for the 6 least important features, so I decided to remove them from my next model.

FS5: UNIT, hour, meantempi, latitude, longitude, conds, pressurei, tempi, wspdi, meanpressurei, meantempi, meanwspdi, weather_lat, weather_lon
r^2 = 0.47199644644065109

Here are the features importances used to come up with FS5:

latitude	0.22906256416322501
longitude	0.22892853644161223
weather_lat	0.092219442846831118
weather_lon	0.09125496494717722
tempi	0.052091002367150148
pressurei	0.051527233585661587
hour	0.051446504213872354
wspdi	0.045086883119520721
meantempi	0.040908449430127697
meanpressurei	0.040789529642363972
meanwspdi	0.040358536328951852
day_week	0.015833672953118903
meanprecipi	0.0082153743969240016
precipi	0.0047897383810357021
rain	0.0044156981536202042
weekday	0.0021388287498403081
fog	0.00093304027896691879

Model Selection and Rationale

After generating all of these models, the question remains as to which model to pick. FS4 and FS5 are the best options of the five models based on their r-squared values. The drop in r-squared from FS4 to FS5 is less than 0.021, so is that made up for by the computational benefit and the lowered risk of overfitting with FS5?

I would say that the computational benefit is negligible since the dummy features are much more numerous that the features I cut in FS5. There are 240 'UNIT' columns and 12 'conds' columns, so the 6 cut features are not that impactful on overall computation.

As for the overfitting, we can test this by cross validating, breaking up our data into a training set to create our model and a testing set to see if our model generalizes to unseen data. Using SciKit Learn's `cross_validation` module, I used 60% of the data to train two linear regression models with the FS4 and FS5 feature sets, respectively.

FS4's cross-validated r^2 = 0.48935596603665432
FS5's cross-validated r^2 = 0.46805366111444557

Both models lose some of their predictive power when tested with unseen data, but neither loses much. Also, these results show that FS4 is not overfit to the data it was trained on, so the danger of overfitting is not an issue in deciding on a model.

For the reasons listed above, I chose FS4 as the best feature set to build a linear regression model for the subway data.

Multicollinearity

There is still one big problem with my model based on FS4, and that problem is multicollinearity, which means that there is a linear combination of the of features which equals 0. When this happens, the minima for the dataset don't look like inverted peaks but more like a trough. This problem results in some really high magnitude coefficients.

The best route in this case is to remove features which have this problem. Since I'm using a constant column (a column of all ones) to represent my linear regression's y-intercept, one source of multicollinearity is the presence of dummy variables. To correct for this, I removed one of the dummy columns for each categorical variable ('unit_003' from 'UNIT' and 'conds_Clear' from 'conds').

Then, I took a look at other features which might have too high of a correlation with each other, which would be another way the multicollinearity problem could creep into my model. Since 'UNIT' encodes the physical location of each data point, 'longitude', 'latitude', 'weather_lat', and 'weather_lon' were all redundant, and introduced multicollinearity. Additionally, 'weekday' and 'day_week' were strongly correlated, so I removed the less granular 'weekday' feature.

Furthermore, I considered removing 'meanprecipi', 'meanpressurei', 'meantempi', and 'meanwspdi' because they had high correlation with their respective, more-granular counterparts, 'precipi', 'pressurei', 'tempi, and 'wspdi'. However, removing these four mean fields dropped the r-squared value by 0.08, and doesn't reduce the conditional number by an order of magnitude (it goes from 2.68e+04 to 1.79e+04), so I decided to keep them. For more on the Condition Number, see *Appendix: References.*

By removing the features that I did remove, I was able to keep the coefficients in a more reasonable range. Whereas before the coefficients were at a magnitude of 10^{10}, now they are in the hundreds and thousands. However, the Condition Number is still over

20000, whereas under 50 is more acceptable. This suggests that there are other numerical problems with my model.

The final feature set based on this analysis then is the following:

FS4′: rain, precpi, UNIT, hour, meantempi, day_week, conds, fog, pressurei, tempi, wspdi, meanprecipi, meanpressurei, meantempi, meanwspdi
r^2 = 0.47893074574967498
Condition Number: 26,800

Results

Having chosen FS4′ for the basis of our model, let's examine exactly what the model looks like. The model in this section will be the cross-validated model from above.

There are 263 features in total. Fourteen of these features are numerical, and the remaining ones are dummy features derived from the two categorical features 'UNIT' and 'conds' that were defined previously (minus two because of the multicollinearity problem discussed above). Likewise, there are 263 coefficients in the model. You can find all of them in *Appendix: Linear Regression Parameters*.

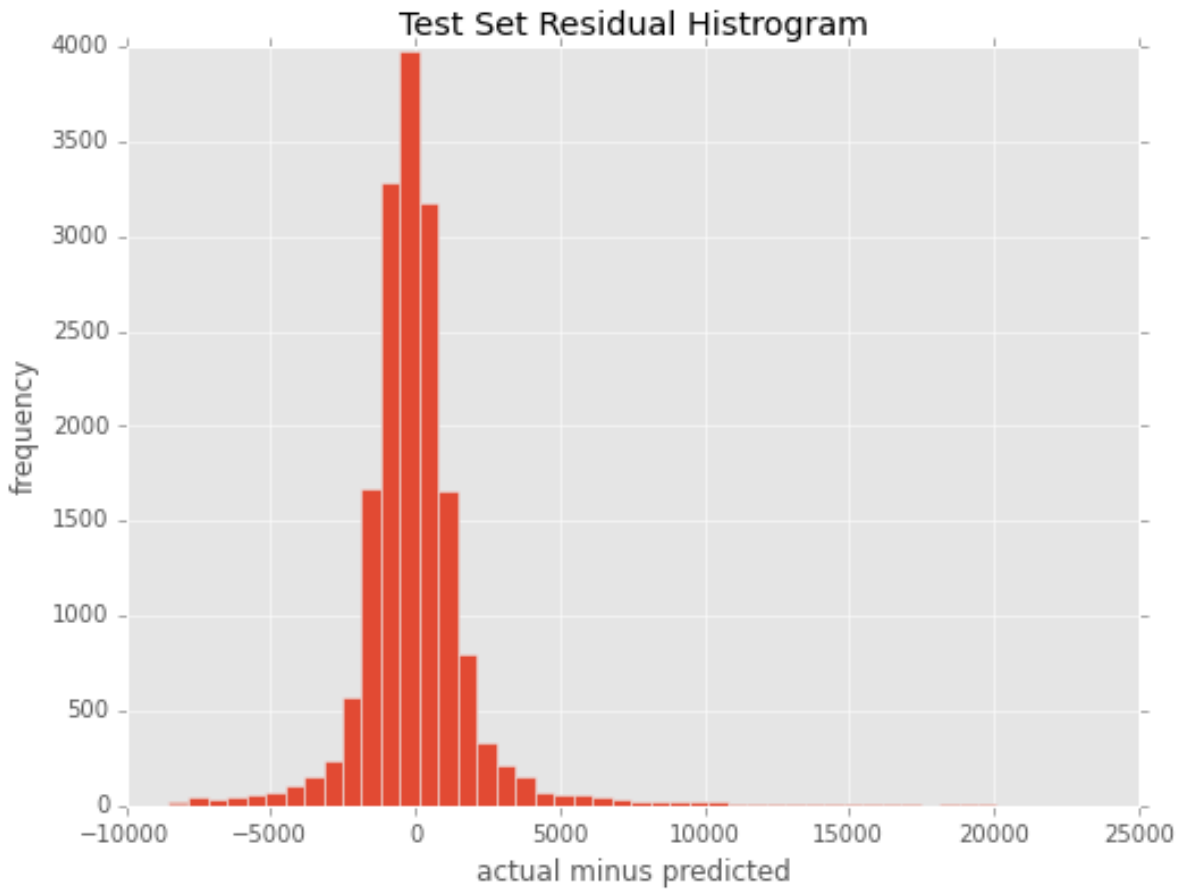
After creating the linear regression model from the training data, I looked at the residual distribution, which is defined by subtracting each training data point's hourly subway entries by the model's predicted value for the number of hourly entries. There were 17,060 data points in the training set. The mean of this residual distribution is 22.002, meaning that on average, my linear regression model underestimated the number of hourly entries by 22.002.

The standard deviation of the residuals, however, is 2,181.491, so in many cases the model overshot or undershot the actual value, but the overshot values and undershot values balanced each other out. This standard deviation is better than the total (training set plus testing set) original data set's standard deviation though, which is 2,952.386. However, this large standard deviation (which can be seen in the long tails of the residual histogram below) suggests that our linear regression model, and the linear regression technique in general, is not a great predictive model for this data.

Another feature of the residuals that shows our linear regression's weakness is the range. The values in the residual distribution go from -8,719.552 to 24,820.004,

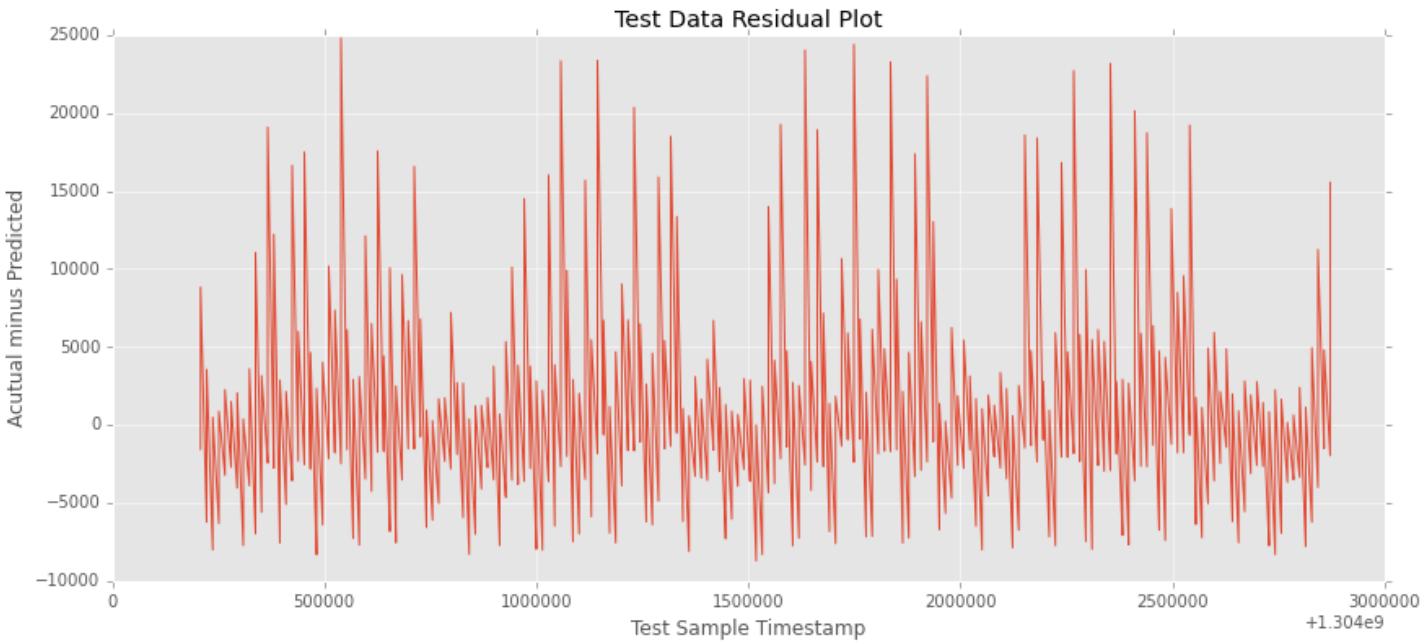
meaning that on the worst cases, the model at one point overshot a correct value by 8,719.552 and undershot a correct value by 24,820.004. So the range of the residual distribution is 33,539.556, which is not great since the range of the original subway data distribution goes from 0 entries to 32,814 entries, for a total range of 32,814, less than the residual distribution's total range.

Here is a histogram of the residual distribution. Notice the long tails which result in the large standard deviation and range mentioned above.



Another graph that illuminates the inadequacy of the linear regression model is to simply plot all of the regression values in a standard line graph. When we do that for all of the actual test data values minus their corresponding predictions, there is distinctive pattern to these residual values. The residual error has a cyclical structure, where weekday values are very underestimated (high, positive residual values) compared with the weekends. Since the error values take on this cyclical structure, a linear regression

model cannot reduce this nonlinear error, and therefore any linear regression model will be lacking in its predictive power.



Conclusion

Our linear regression model is very close to describing half of the variance present in our data, which, for something as inherently unpredictable as how humans get around in their day to day lives, is pretty good. However, analysis of the residual distribution showed that a better solution is out there that's not based on linear regression.

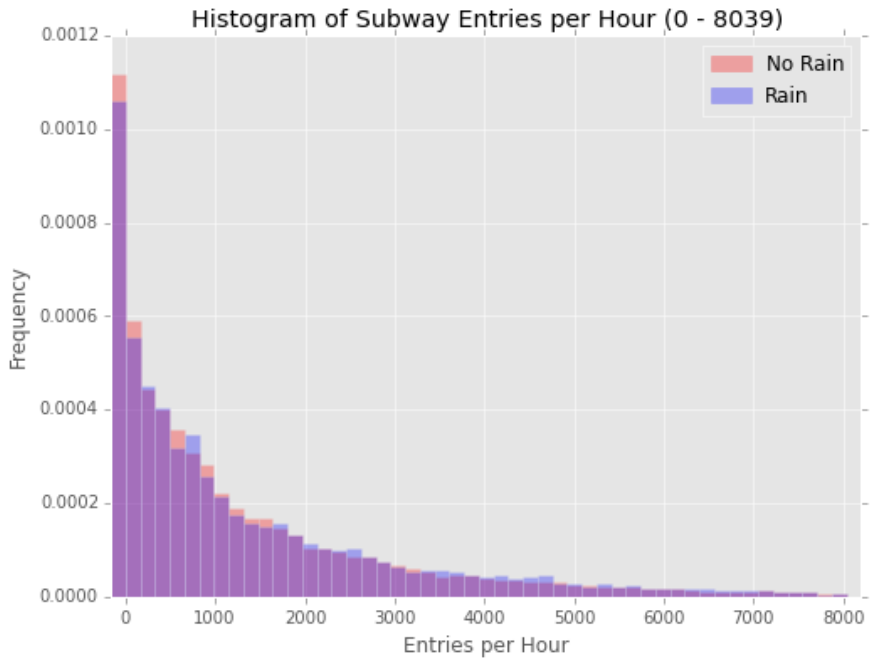
Also, the end use of the model would play a big role in determining how good of a model it is. If the model is used by an individual to decide whether to use Uber on a particular day in lieu of the subway, then the current model is good enough for that usage. However, if New York City wanted to determine whether additional transit infrastructure should be built to handle ridership under certain conditions, then more data should be collected and a better model be constructed.

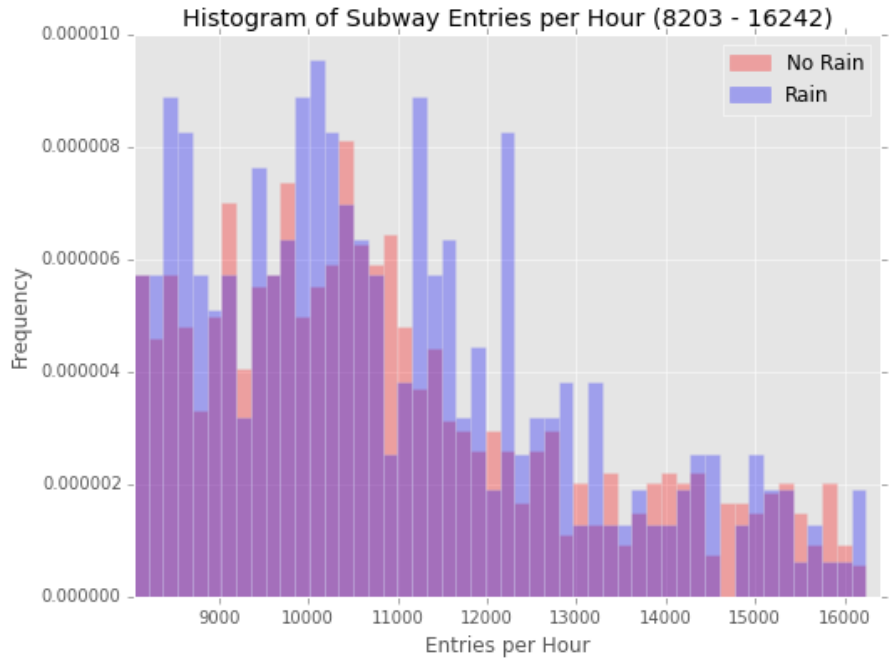
Visualization

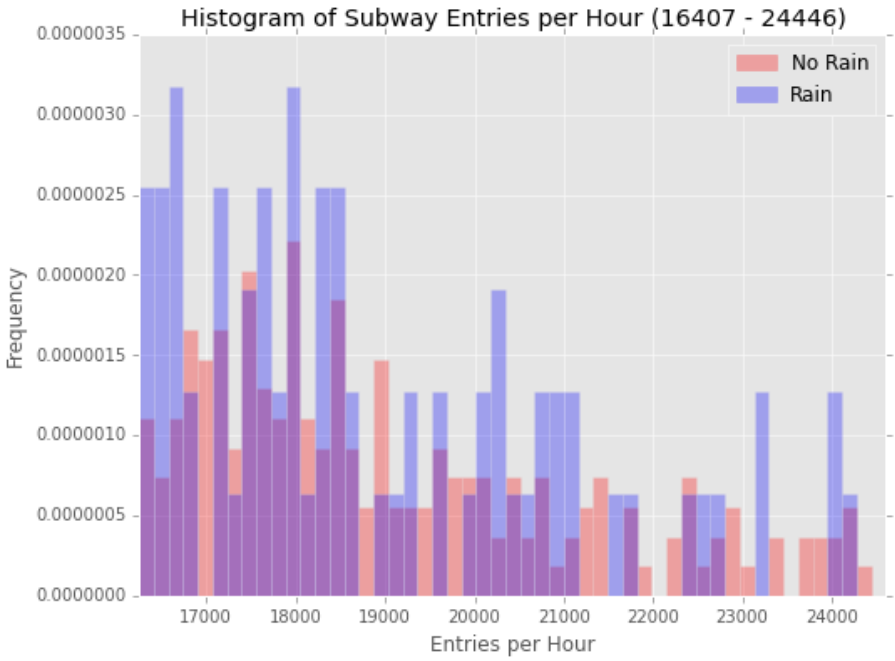
Rain vs Non-Rain Histogram

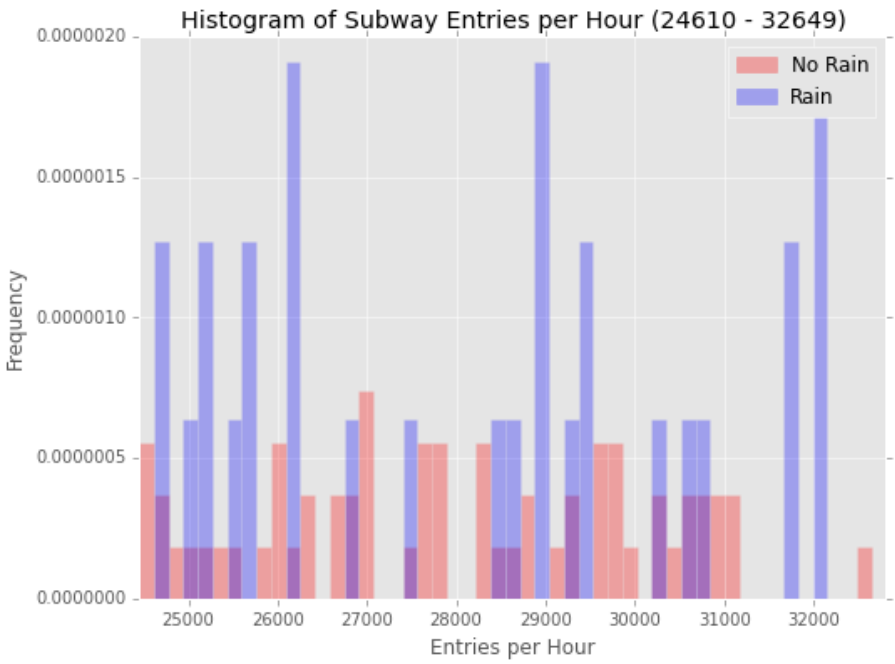
In the opening section of this report, I showed a histogram entitled “Histogram of Subway Entries per Hour”. This histogram is split up into two populations – one for days with rain and one for days without rain. Like I mentioned in the section where it appears, for clarity, bins over 6,000 were omitted. Also, I noted that the magnitudes of the histograms were misleading since the sizes of the two populations were different: 33,064 non-rain samples and 9,585 rainy samples.

To balance for that, I normalized the histograms of each population. Then, I graphed four separate segments of the histogram so that I could adjust the y-axis scale (otherwise, the bins toward the right were not even visible). In these normalized histograms, you can sort of see that the rainy days win out in the higher intervals, but it is much less clear to the naked eye. If anything, the main thing that this visualization shows is how nearly impossible it is to just guess whether the two populations are statistically different from each other as the data becomes more sporadic at the higher values. Luckily, that’s what the statistical testing was for in the earlier section.




Histogram of Subway Entries per Hour (8203 - 16242)
No Rain
Rain
Frequency
0.000010
0.000008
0.000006
0.000004
0.000002
0.000000
9000
10000
11000
12000
13000
14000
15000
16000
Entries per Hour

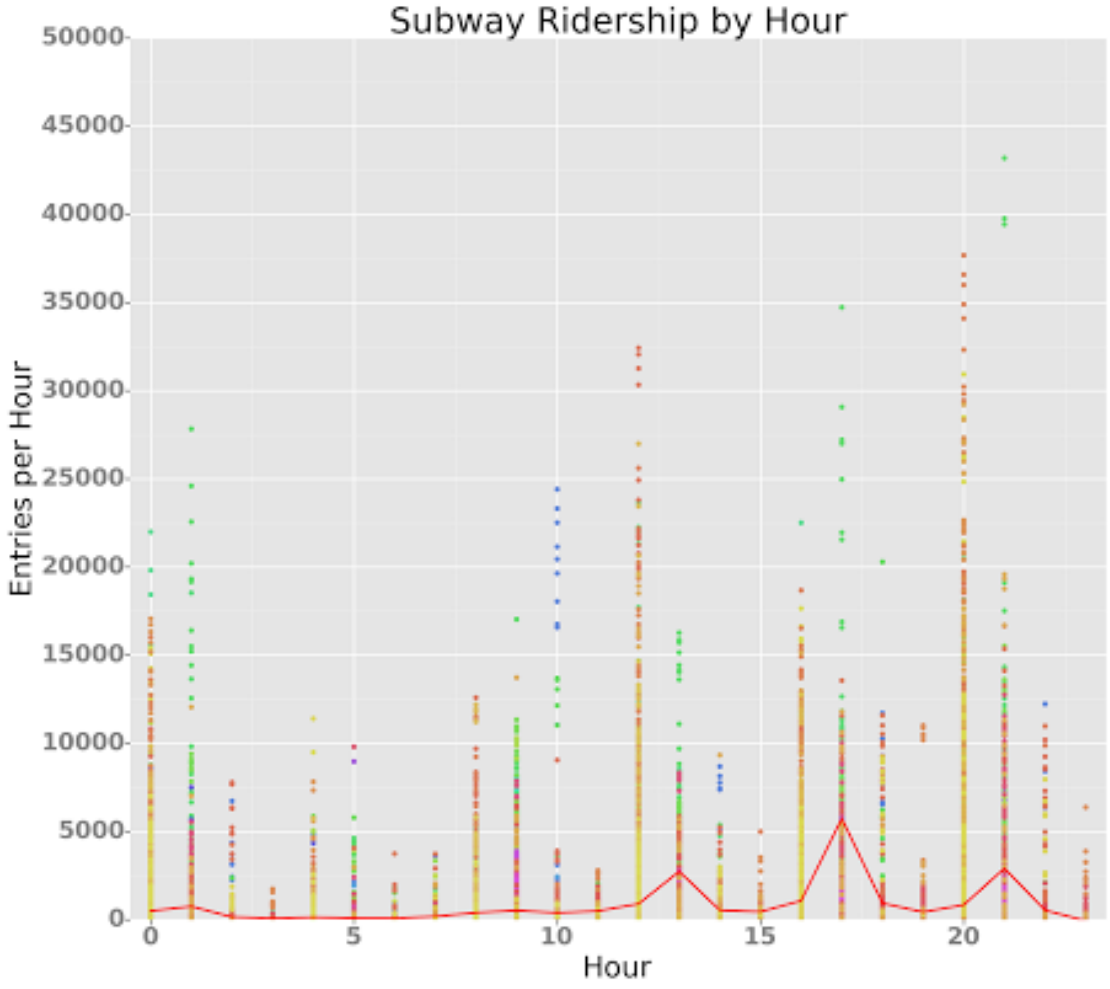

Histogram of Subway Entries per Hour (16407 - 24446)
No Rain
Rain
Frequency
0.0000035
0.0000030
0.0000025
0.0000020
0.0000015
0.0000010
0.0000005
0.0000000
17000
18000
19000
20000
21000
22000
23000
24000
Entries per Hour


Histogram of Subway Entries per Hour (24610 - 32649)
No Rain
Rain
Frequency
0.0000020
0.0000015
0.0000010
0.0000005
0.0000000
25000
26000
27000
28000
29000
30000
31000
32000
Entries per Hour

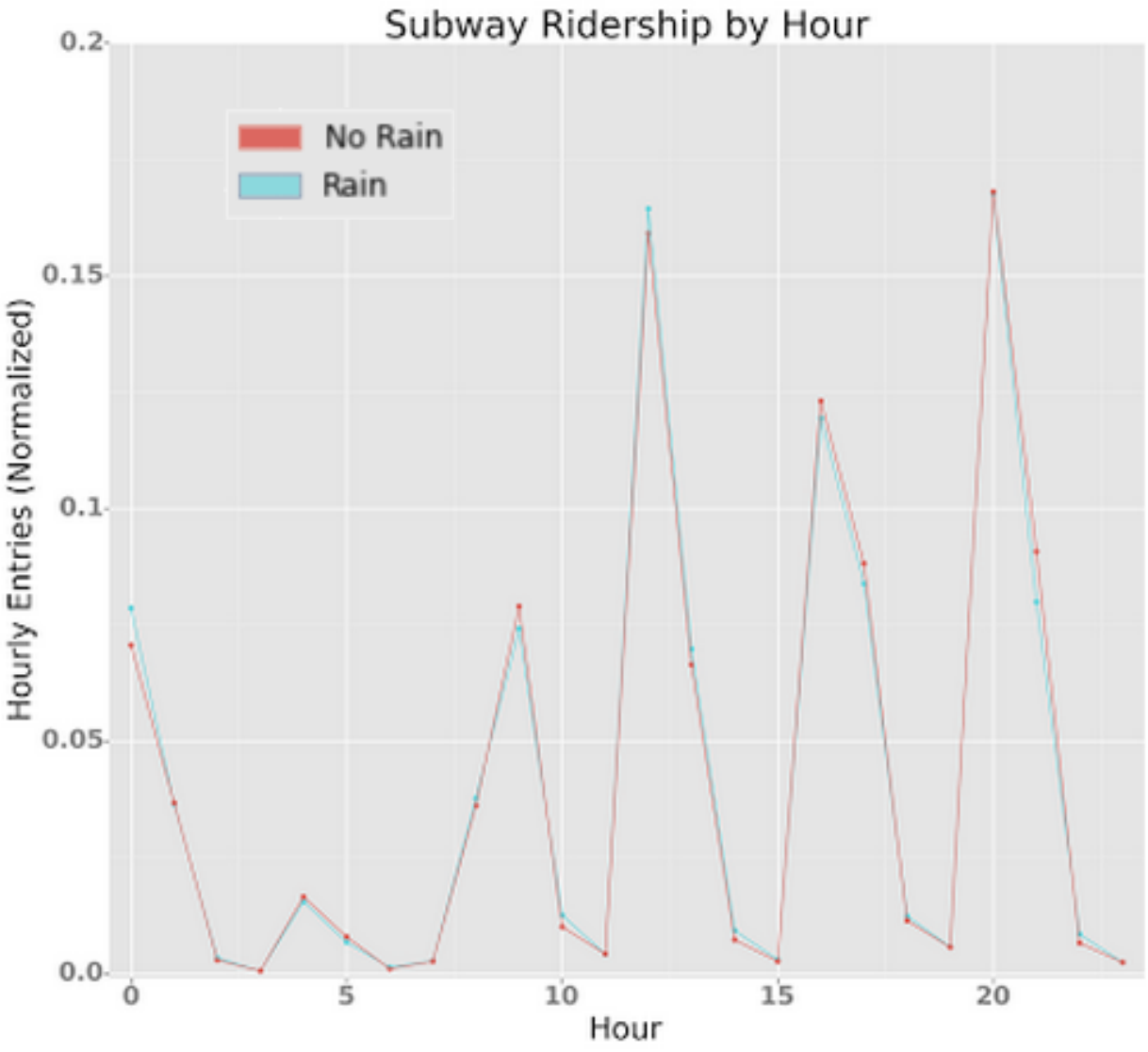
Subway Ridership by Hour

One of the more important predictive features of the subway data set was the hour of the day, which makes sense because more people will be using public transportation at 5pm rush hour than at midnight. I decided to graph this feature to see just how much it predicts the subway ridership data.

In the first figure, I plotted the hourly entries on the y-axis and the hour of the day on the x-axis. I plotted each turnstile unit in a different color, which has the added effect of showing how certain turnstiles groups are much more popular that others based on their clustering. Finally, I plotted the mean of all the subway entry data and drew a regression line to connect them all to show exactly how the data was concentrated for each hour, since the actual point density is virtually indiscernible to the naked eye.



I also wanted to see how the hour of the day affected the data differently on rainy days and non-rainy days. I plotted a line graph of the data separated on this feature. This graph is normalized to account for the difference in population sizes. It's remarkable just how similar these two plots are. While there's a little deviation here and there, this graph is evidence that the "rush hour" peaks of subway ridership are not that influenced by whether or not it rained that day.



Conclusion

Analysis

Based on the statistical test and the linear regression, I am somewhat confident that the subway ridership on rainy days is greater than the subway ridership on non-rainy days. When I ran the Mann-Whitney U Test on the data, I found that the populations differed with a p-value passed my significance threshold. With my linear regression model, I was able to explain over 49% of the variance in the data with the model, which pretty successful given the inherent randomness of human behavior along any axis, transportation preferences included. However, there is a high chance that a better model exists other than a linear regression one.

Reflection

In doing the analysis on this data, I came to realize a couple of things. Firstly, when doing an analysis on human behavior, a lot of data is required. When I started the project, I thought that having 42,649 data points was enough. However, after working with the data, I believe that a larger data set would have helped my analysis greatly. Since my p-value was so close to the significance threshold, more data could push it out of the significant interval, causing me to reconsider this result.

Along the lines of collecting more data is the importance of getting a variety of data so that our model can be as generalized as possible. All of the data analyzed here is from May 2011. On a weekly basis, we saw cycles in the data. It's very conceivable that there are seasonal cycles as well. So the current data set is very limited in that regard.

Another interesting discovery I made in doing this analysis was that, even though rain was a good predictor for subway ridership, it wasn't the best feature to use. The most important weather feature appeared to be temperature, followed closely by pressure. However, it's important to note that with this weather data, not all of the features were orthogonal, since the pressure is related to temperature and whether or not it will rain.

Appendix

References

Mann Whitney U Test —
https://en.wikipedia.org/wiki/Mann%E2%80%93Whitney_U_test

SciPy Mann Whitney U Test Implementation —
http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.mannwhitneyu.html

Poisson Distribution —
https://en.wikipedia.org/wiki/Poisson_distribution

SciKit Learn Cross Validation Implementation —
http://scikit-learn.org/stable/modules/cross_validation.html

R-Squared —
Think Stats Book, p. 107 “Goodness of Fit”
http://people.duke.edu/~rnau/rsquared.htm

Pandas DataFrame Sample —
http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sample.html

SciKit Learn ExtraTreesClasifier Implementation —
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier
http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html#example-ensemble-plot-forest-importances-py

Multicollinearity –
https://discussions.udacity.com/t/webcast-multicollinearity-in-linear-regression-tuesday-16th-june-at-5pm-pacific-time/21994

Condition Number –
http://work.thaslwanter.at/Stats/html/statsModels.html#condition-number

Data Science Concepts –
https://www.udacity.com/course/intro-to-data-science--ud359

Documentation

SciPy Mann Whitney U:

```
scipy.stats.mannwhitneyu(x, y, use_continuity=True)
```

Computes the Mann-Whitney rank test on samples x and y.

Pandas Get Dummies:

```
pandas.get_dummies(data, prefix=None, prefix_sep='_',
     dummy_na=False, columns=None, sparse=False)
```

Convert categorical variable into dummy/indicator variables

StatsModel OLS:

```
statsmodels.regression.linear_model.OLS(endog, exog=None,
     missing='none', hasconst=None, **kwargs)
```

A simple ordinary least squares model.

SciKit Learn Train/Test Split:

```
sklearn.cross_validation.train_test_split(*arrays, **options)
```

Split arrays or matrices into random train and test subsets. Quick utility that wraps input validation and ``next(iter(ShuffleSplit(n_samples)))`` and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

SciKit Learn *Extra Trees Classifier:*

This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Code Snippets

```
def linear_regression(features, values):
  features = sm.add_constant(features)
  model = sm.OLS(values, features)
  results = model.fit()
  intercept = results.params[0]
  params = results.params[1:]

  return intercept, params



def calculate_r_squared(data, predictions):
  ss_tot = sum((data - np.mean(data)) ** 2)
  ss_res = sum((data - predictions) ** 2)
  r_squared = 1 - ss_res / ss_tot
  return r_squared



def feature_importances(input_data, outcomes):
  forest = ExtraTreesClassifier(n_estimators=30, random_state=0)
  forest.fit(input_data, outcomes)
  return forest.feature_importances_
```

[Continued on next page]

```
def predictions(dataframe, feature_list, test_data=None):

  feature_list += ['ENTRIESn_hourly', 'datetime']

  if test_data is None:
    test_data = dataframe

  add_dummy_units, add_dummy_conds = False, False
  if 'UNIT' in feature_list:
    feature_list = filter(lambda x: x != 'UNIT', feature_list)
    add_dummy_units = True
  if 'conds' in feature_list:
    feature_list = filter(lambda x: x != 'conds', feature_list)
    add_dummy_conds = True

  features = dataframe[feature_list]
  test_data_features = test_data[feature_list]

  if add_dummy_units:
    dummy_units = pandas.get_dummies(dataframe['UNIT'], prefix='unit')
    features = features.join(dummy_units)
    features.drop('unit_R003', axis=1, inplace=True)
    test_dummy_units = pandas.get_dummies(test_data['UNIT'], prefix='unit')
    test_data_features = test_data_features.join(test_dummy_units)
    test_data_features.drop('unit_R003', axis=1, inplace=True)

  if add_dummy_conds:
    dummy_conds = pandas.get_dummies(dataframe['conds'], prefix='conds')
    features = features.join(dummy_conds)
    features.drop('conds_Clear', axis=1, inplace=True)
    test_dummy_conds = pandas.get_dummies(test_data['conds'], prefix='conds')
    test_data_features = test_data_features.join(test_dummy_conds)
    test_data_features.drop('conds_Clear', axis=1, inplace=True)

  features.dropna(axis=0, inplace=True)
  test_data_features.dropna(axis=0, inplace=True)

  values = features['ENTRIESn_hourly']
  datetimes = test_data_features['datetime']

  features.drop('ENTRIESn_hourly', axis=1, inplace=True)
  features.drop('datetime', axis=1, inplace=True)
  test_data_features.drop('ENTRIESn_hourly', axis=1, inplace=True)
  test_data_features.drop('datetime', axis=1, inplace=True)
  intercept, params = linear_regression(features, values)

  predictions = intercept + np.dot(test_data_features, params)

  return predictions, datetimes
```

Linear Regression Parameters

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
const	2.226E+04	3459.216	6.434	0.000	1.55E+04	2.9E+04
hour	103.4656	2.302	44.950	0.000	98.954	107.977
day_week	-158.7057	7.204	-22.031	0.000	-172.825	-144.586
fog	146.7429	242.227	0.606	0.545	-328.037	621.522
precipi	-1442.1964	1166.031	-1.237	0.216	-3727.685	843.292
pressurei	100.8339	334.214	0.302	0.763	-554.245	755.912
rain	-73.3674	47.838	-1.534	0.125	-167.132	20.397
tempi	45.3130	3.334	13.592	0.000	38.778	51.848
wspdi	9.9159	4.659	2.129	0.033	0.785	19.047
meanprecipi	9453.3488	1341.376	7.048	0.000	6824.175	1.21E+04
meanpressurei	-779.1690	355.377	-2.193	0.028	-1475.728	-82.610
meantempi	-73.0303	3.984	-18.330	0.000	-80.840	-65.221
meanwspdi	-38.9178	8.144	-4.778	0.000	-54.881	-22.954
unit_R004	369.1180	290.271	1.272	0.204	-199.831	938.067
unit_R005	424.7895	287.540	1.477	0.140	-138.805	988.384
unit_R006	536.2610	283.050	1.895	0.058	-18.532	1091.054
unit_R007	115.6512	295.605	0.391	0.696	-463.751	695.054
unit_R008	226.1727	287.440	0.787	0.431	-337.227	789.572
unit_R009	137.7867	288.811	0.477	0.633	-428.300	703.874
unit_R011	6590.2943	284.206	23.188	0.000	6033.234	7147.354
unit_R012	8764.7369	289.695	30.255	0.000	8196.919	9332.555
unit_R013	2284.4922	284.682	8.025	0.000	1726.500	2842.484
unit_R016	365.7057	284.140	1.287	0.198	-191.225	922.637
unit_R017	3737.9490	282.415	13.236	0.000	3184.400	4291.498
unit_R018	7327.3948	292.277	25.070	0.000	6754.515	7900.274
unit_R019	3055.2329	282.994	10.796	0.000	2500.548	3609.918
unit_R020	5917.7979	292.573	20.227	0.000	5344.337	6491.259
unit_R021	4420.7246	281.786	15.688	0.000	3868.407	4973.042
unit_R022	8744.5686	284.808	30.703	0.000	8186.328	9302.809
unit_R023	5373.9571	284.079	18.917	0.000	4817.146	5930.768

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
unit_R024	2931.9525	284.850	10.293	0.000	2373.630	3490.275
unit_R025	5393.4550	276.867	19.480	0.000	4850.779	5936.131
unit_R027	2939.3423	289.858	10.141	0.000	2371.203	3507.481
unit_R029	6271.5037	287.192	21.837	0.000	5708.591	6834.416
unit_R030	2508.6079	294.691	8.513	0.000	1930.997	3086.219
unit_R031	4101.5833	289.759	14.155	0.000	3533.639	4669.528
unit_R032	3922.2240	281.815	13.918	0.000	3369.850	4474.598
unit_R033	8077.6616	281.247	28.721	0.000	7526.400	8628.923
unit_R034	825.5554	292.629	2.821	0.005	251.986	1399.125
unit_R035	2543.7616	287.174	8.858	0.000	1980.883	3106.640
unit_R036	518.3149	296.190	1.750	0.080	-62.234	1098.864
unit_R037	663.8376	287.935	2.306	0.021	99.469	1228.206
unit_R038	21.9910	291.516	0.075	0.940	-549.397	593.379
unit_R039	649.3575	286.607	2.266	0.023	87.591	1211.124
unit_R040	1055.7065	285.283	3.701	0.000	496.536	1614.877
unit_R041	2627.6849	286.530	9.171	0.000	2066.070	3189.299
unit_R042	269.4517	287.827	0.936	0.349	-294.705	833.608
unit_R043	2555.9296	282.350	9.052	0.000	2002.508	3109.351
unit_R044	4132.8682	289.849	14.259	0.000	3564.747	4700.989
unit_R046	7539.4200	285.774	26.382	0.000	6979.287	8099.553
unit_R049	2528.0324	286.675	8.818	0.000	1966.132	3089.933
unit_R050	3495.1624	291.252	12.000	0.000	2924.292	4066.033
unit_R051	4786.5955	287.880	16.627	0.000	4222.333	5350.858
unit_R052	1142.6829	287.275	3.978	0.000	579.608	1705.758
unit_R053	3124.0426	287.946	10.849	0.000	2559.651	3688.434
unit_R054	976.7851	284.163	3.437	0.001	419.809	1533.761
unit_R055	7999.8456	290.747	27.515	0.000	7429.964	8569.727
unit_R056	1038.8799	301.443	3.446	0.001	448.035	1629.725
unit_R057	4574.4907	281.898	16.227	0.000	4021.955	5127.027
unit_R058	335.6148	289.359	1.160	0.246	-231.545	902.774

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
unit_R059	802.8771	281.267	2.855	0.004	251.578	1354.176
unit_R060	423.5996	285.780	1.482	0.138	-136.546	983.746
unit_R061	270.4252	286.524	0.944	0.345	-291.178	832.028
unit_R062	2214.2651	291.218	7.603	0.000	1643.462	2785.069
unit_R063	808.6889	287.797	2.810	0.005	244.590	1372.788
unit_R064	415.2890	288.570	1.439	0.150	-150.325	980.903
unit_R065	639.6195	297.105	2.153	0.031	57.276	1221.963
unit_R066	-87.9332	294.096	-0.299	0.765	-664.378	488.512
unit_R067	837.0564	293.021	2.857	0.004	262.718	1411.395
unit_R068	330.4013	297.031	1.112	0.266	-251.797	912.599
unit_R069	807.3210	287.932	2.804	0.005	242.958	1371.685
unit_R070	1341.6815	290.404	4.620	0.000	772.474	1910.889
unit_R080	3194.7573	290.677	10.991	0.000	2625.013	3764.502
unit_R081	3063.9000	279.748	10.952	0.000	2515.578	3612.222
unit_R082	1125.8593	289.229	3.893	0.000	558.955	1692.764
unit_R083	2814.1676	285.822	9.846	0.000	2253.940	3374.396
unit_R084	9472.5322	289.805	32.686	0.000	8904.497	1E+04
unit_R085	2330.6915	283.100	8.233	0.000	1775.799	2885.584
unit_R086	2250.5119	281.890	7.984	0.000	1697.991	2803.033
unit_R087	806.4665	285.989	2.820	0.005	245.912	1367.021
unit_R089	88.8938	285.918	0.311	0.756	-471.522	649.309
unit_R090	370.6586	286.617	1.293	0.196	-191.128	932.445
unit_R091	957.7376	295.411	3.242	0.001	378.715	1536.760
unit_R092	1989.9977	297.034	6.700	0.000	1407.794	2572.201
unit_R093	1875.7983	279.938	6.701	0.000	1327.103	2424.494
unit_R094	1543.4892	284.665	5.422	0.000	985.529	2101.449
unit_R095	2079.3758	281.053	7.399	0.000	1528.496	2630.256
unit_R096	2308.9962	278.826	8.281	0.000	1762.482	2855.511
unit_R097	2879.1563	276.742	10.404	0.000	2336.726	3421.587
unit_R098	1424.4380	291.868	4.880	0.000	852.359	1996.517

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
unit_R099	1953.8391	285.901	6.834	0.000	1393.456	2514.222
unit_R100	465.6265	284.671	1.636	0.102	-92.346	1023.599
unit_R101	2546.9823	285.411	8.924	0.000	1987.560	3106.404
unit_R102	3301.0898	290.631	11.358	0.000	2731.437	3870.742
unit_R103	1222.6313	287.266	4.256	0.000	659.574	1785.689
unit_R104	1212.2577	285.944	4.240	0.000	651.792	1772.723
unit_R105	2876.3852	289.818	9.925	0.000	2308.326	3444.444
unit_R106	1014.1680	298.750	3.395	0.001	428.601	1599.735
unit_R107	397.3630	293.785	1.353	0.176	-178.472	973.198
unit_R108	5049.1696	283.539	17.808	0.000	4493.416	5604.923
unit_R111	2985.4523	285.871	10.443	0.000	2425.130	3545.775
unit_R112	1566.0312	286.611	5.464	0.000	1004.256	2127.806
unit_R114	795.0164	289.490	2.746	0.006	227.599	1362.434
unit_R115	1148.9323	282.465	4.068	0.000	595.284	1702.581
unit_R116	3108.0997	285.865	10.873	0.000	2547.788	3668.412
unit_R117	841.5472	297.912	2.825	0.005	257.622	1425.472
unit_R119	1658.1274	285.926	5.799	0.000	1097.696	2218.559
unit_R120	1308.8213	290.766	4.501	0.000	738.902	1878.740
unit_R121	1225.9715	290.404	4.222	0.000	656.762	1795.181
unit_R122	2874.0784	285.293	10.074	0.000	2314.888	3433.269
unit_R123	1236.0655	291.224	4.244	0.000	665.250	1806.881
unit_R124	404.3783	293.297	1.379	0.168	-170.501	979.258
unit_R126	1348.6070	287.733	4.687	0.000	784.633	1912.581
unit_R127	4779.5790	284.121	16.822	0.000	4222.685	5336.473
unit_R137	2286.5869	282.818	8.085	0.000	1732.247	2840.927
unit_R139	2324.2650	283.453	8.200	0.000	1768.681	2879.849
unit_R163	2935.0971	290.448	10.105	0.000	2365.803	3504.391
unit_R172	1623.9504	280.869	5.782	0.000	1073.430	2174.471
unit_R179	6681.2743	284.120	23.516	0.000	6124.382	7238.167
unit_R181	1575.1729	292.558	5.384	0.000	1001.743	2148.603

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
unit_R183	712.5761	293.032	2.432	0.015	138.216	1286.936
unit_R184	777.3892	297.721	2.611	0.009	193.839	1360.940
unit_R186	641.7250	292.547	2.194	0.028	68.317	1215.133
unit_R188	2085.2741	278.105	7.498	0.000	1540.172	2630.376
unit_R189	1076.7923	293.350	3.671	0.000	501.809	1651.775
unit_R194	1697.0941	284.606	5.963	0.000	1139.250	2254.938
unit_R196	967.4297	285.302	3.391	0.001	408.221	1526.639
unit_R198	1852.2273	281.187	6.587	0.000	1301.084	2403.371
unit_R199	348.2234	294.076	1.184	0.236	-228.183	924.629
unit_R200	1020.8060	292.451	3.491	0.000	447.585	1594.027
unit_R202	2326.4155	292.251	7.960	0.000	1753.587	2899.244
unit_R203	1519.6524	297.202	5.113	0.000	937.120	2102.185
unit_R204	1036.0495	283.552	3.654	0.000	480.272	1591.827
unit_R205	1431.4476	281.048	5.093	0.000	880.578	1982.317
unit_R207	1631.7441	284.171	5.742	0.000	1074.753	2188.735
unit_R208	2264.2214	285.971	7.918	0.000	1703.703	2824.740
unit_R209	536.8334	305.949	1.755	0.079	-62.844	1136.511
unit_R210	236.0282	284.780	0.829	0.407	-322.156	794.212
unit_R211	1990.0594	291.816	6.820	0.000	1418.084	2562.035
unit_R212	1525.2092	287.820	5.299	0.000	961.066	2089.353
unit_R213	780.1234	287.109	2.717	0.007	217.374	1342.873
unit_R214	680.2273	307.087	2.215	0.027	78.318	1282.136
unit_R215	1277.6953	288.375	4.431	0.000	712.464	1842.927
unit_R216	511.9814	281.301	1.820	0.069	-39.385	1063.347
unit_R217	780.6152	294.049	2.655	0.008	204.261	1356.969
unit_R218	2007.7891	295.402	6.797	0.000	1428.784	2586.794
unit_R219	1242.9144	276.211	4.500	0.000	701.525	1784.304
unit_R220	1036.3829	292.722	3.540	0.000	462.630	1610.136
unit_R221	1198.1287	287.126	4.173	0.000	635.345	1760.912
unit_R223	2122.7702	287.278	7.389	0.000	1559.689	2685.852

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
unit_R224	441.1025	287.108	1.536	0.124	-121.645	1003.850
unit_R225	288.3298	289.976	0.994	0.320	-280.039	856.699
unit_R226	647.9430	287.260	2.256	0.024	84.898	1210.988
unit_R227	655.2775	283.577	2.311	0.021	99.451	1211.104
unit_R228	806.0117	289.678	2.782	0.005	238.226	1373.797
unit_R229	222.8797	297.236	0.750	0.453	-359.721	805.480
unit_R230	274.7926	285.150	0.964	0.335	-284.118	833.703
unit_R231	739.7763	293.981	2.516	0.012	163.557	1315.996
unit_R232	574.1640	286.509	2.004	0.045	12.589	1135.739
unit_R233	977.8415	297.135	3.291	0.001	395.441	1560.242
unit_R234	0.8029	291.952	0.003	0.998	-571.439	573.045
unit_R235	2378.6726	292.653	8.128	0.000	1805.056	2952.289
unit_R236	1528.1180	282.831	5.403	0.000	973.753	2082.483
unit_R237	616.0315	291.496	2.113	0.035	44.682	1187.381
unit_R238	2019.0075	282.223	7.154	0.000	1465.834	2572.181
unit_R239	571.7306	285.314	2.004	0.045	12.499	1130.962
unit_R240	2410.6648	284.621	8.470	0.000	1852.790	2968.539
unit_R242	240.1784	291.904	0.823	0.411	-331.970	812.326
unit_R243	1225.3011	284.674	4.304	0.000	667.324	1783.279
unit_R244	1451.0262	294.580	4.926	0.000	873.632	2028.420
unit_R246	311.5219	296.575	1.050	0.294	-269.782	892.826
unit_R247	-55.2941	298.669	-0.185	0.853	-640.703	530.115
unit_R248	2687.0459	291.921	9.205	0.000	2114.863	3259.229
unit_R249	898.3936	284.789	3.155	0.002	340.192	1456.596
unit_R250	734.3598	304.031	2.415	0.016	138.442	1330.278
unit_R251	1046.9323	294.804	3.551	0.000	469.100	1624.765
unit_R252	640.0978	287.809	2.224	0.026	75.976	1204.219
unit_R253	476.2455	283.444	1.680	0.093	-79.322	1031.813
unit_R254	2243.0641	284.039	7.897	0.000	1686.331	2799.797
unit_R255	699.9680	281.796	2.484	0.013	147.632	1252.304

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
unit_R256	819.9948	282.313	2.905	0.004	266.644	1373.345
unit_R257	1593.9342	282.913	5.634	0.000	1039.409	2148.459
unit_R258	1235.6886	283.588	4.357	0.000	679.839	1791.538
unit_R259	524.2286	283.608	1.848	0.065	-31.660	1080.117
unit_R260	717.6335	298.791	2.402	0.016	131.985	1303.282
unit_R261	1550.1696	297.897	5.204	0.000	966.275	2134.064
unit_R262	154.0903	294.158	0.524	0.600	-422.477	730.657
unit_R263	-156.0383	291.163	-0.536	0.592	-726.735	414.658
unit_R264	166.3364	290.417	0.573	0.567	-402.899	735.571
unit_R265	597.3835	297.145	2.010	0.044	14.963	1179.804
unit_R266	792.8106	288.660	2.747	0.006	227.020	1358.601
unit_R269	567.9348	284.793	1.994	0.046	9.724	1126.145
unit_R270	168.9309	285.458	0.592	0.554	-390.583	728.445
unit_R271	90.4773	291.310	0.311	0.756	-480.506	661.461
unit_R273	1086.3085	294.858	3.684	0.000	508.370	1664.247
unit_R274	743.8983	291.756	2.550	0.011	172.040	1315.756
unit_R275	775.6650	291.518	2.661	0.008	204.274	1347.056
unit_R276	1109.3287	282.426	3.928	0.000	555.757	1662.900
unit_R277	244.8509	297.265	0.824	0.410	-337.805	827.506
unit_R278	136.5007	299.641	0.456	0.649	-450.813	723.815
unit_R279	674.6319	283.415	2.380	0.017	119.123	1230.141
unit_R280	359.0968	296.353	1.212	0.226	-221.772	939.966
unit_R281	987.1788	285.896	3.453	0.001	426.806	1547.552
unit_R282	1223.5555	285.376	4.288	0.000	664.202	1782.909
unit_R284	414.0286	287.747	1.439	0.150	-149.973	978.030
unit_R285	549.7894	304.392	1.806	0.071	-46.837	1146.416
unit_R287	205.4215	301.566	0.681	0.496	-385.665	796.508
unit_R291	1222.6187	283.095	4.319	0.000	667.737	1777.500
unit_R294	621.3606	284.060	2.187	0.029	64.587	1178.134
unit_R295	460.5662	302.389	1.523	0.128	-132.134	1053.266

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
unit_R300	1966.5409	283.550	6.935	0.000	1410.767	2522.315
unit_R303	860.6935	293.349	2.934	0.003	285.713	1435.674
unit_R304	782.8743	294.064	2.662	0.008	206.491	1359.257
unit_R307	173.4779	290.804	0.597	0.551	-396.515	743.470
unit_R308	616.6505	294.582	2.093	0.036	39.253	1194.048
unit_R309	719.0169	284.045	2.531	0.011	162.273	1275.761
unit_R310	1213.4060	287.954	4.214	0.000	649.000	1777.812
unit_R311	80.1136	297.235	0.270	0.788	-502.485	662.712
unit_R312	-57.3178	283.462	-0.202	0.840	-612.919	498.284
unit_R313	-38.1015	294.598	-0.129	0.897	-615.530	539.327
unit_R318	238.6733	291.928	0.818	0.414	-333.523	810.870
unit_R319	983.1575	286.462	3.432	0.001	421.675	1544.640
unit_R321	725.2055	283.627	2.557	0.011	169.280	1281.131
unit_R322	1521.6853	289.064	5.264	0.000	955.103	2088.267
unit_R323	961.0219	291.904	3.292	0.001	388.873	1533.171
unit_R325	223.3824	285.308	0.783	0.434	-335.838	782.603
unit_R330	640.2951	287.898	2.224	0.026	75.998	1204.592
unit_R335	379.1425	299.927	1.264	0.206	-208.732	967.017
unit_R336	24.3370	294.794	0.083	0.934	-553.476	602.150
unit_R337	-76.9667	292.432	-0.263	0.792	-650.151	496.217
unit_R338	-98.7951	288.832	-0.342	0.732	-664.923	467.333
unit_R341	477.1573	286.761	1.664	0.096	-84.911	1039.225
unit_R344	199.3412	293.390	0.679	0.497	-375.721	774.403
unit_R345	62.5111	288.528	0.217	0.828	-503.021	628.043
unit_R346	896.6019	289.148	3.101	0.002	329.854	1463.349
unit_R348	72.9974	284.661	0.256	0.798	-484.955	630.950
unit_R354	154.5193	285.499	0.541	0.588	-405.074	714.113
unit_R356	824.7241	290.919	2.835	0.005	254.507	1394.941
unit_R358	211.6330	290.118	0.729	0.466	-357.014	780.280
unit_R370	149.3546	291.117	0.513	0.608	-421.251	719.961

Feature	Coefficient	STD	ERR	t	P > | t |	[95.0% Conf. Int.]
unit_R371	339.7832	294.723	1.153	0.249	-237.892	917.458
unit_R372	433.7426	296.156	1.465	0.143	-146.741	1014.226
unit_R373	286.7111	299.642	0.957	0.339	-300.604	874.027
unit_R382	873.7440	281.793	3.101	0.002	321.414	1426.074
unit_R424	147.1765	295.664	0.498	0.619	-432.342	726.695
unit_R429	969.9395	282.213	3.437	0.001	416.786	1523.093
unit_R453	1261.1499	302.364	4.171	0.000	668.499	1853.801
unit_R454	-210.7897	295.593	-0.713	0.476	-790.169	368.589
unit_R455	-367.3221	294.935	-1.245	0.213	-945.413	210.768
unit_R456	-121.7382	293.392	-0.415	0.678	-696.804	453.327
unit_R459	-98.9040	339.161	-0.292	0.771	-763.679	565.871
unit_R464	-175.0081	284.797	-0.615	0.539	-733.227	383.211
conds_Fog	12.4251	534.012	0.023	0.981	-1034.270	1059.120
conds_Haze	81.3800	78.773	1.033	0.302	-73.020	235.780
conds_Heavy Rain	-1134.8119	309.032	-3.672	0.000	-1740.532	-529.092
conds_Light Drizzle	-1037.3001	158.035	-6.564	0.000	-1347.058	-727.542
conds_Light Rain	196.7151	85.362	2.304	0.021	29.401	364.030
conds_Mist	697.7834	530.048	1.316	0.188	-341.142	1736.709
conds_Mostly Cloudy	-519.8329	47.149	-11.025	0.000	-612.248	-427.418
conds_Overcast	-448.5743	41.116	-10.910	0.000	-529.164	-367.985
conds_Partly Cloudy	-261.3683	65.219	-4.008	0.000	-389.202	-133.535
conds_Rain	-1225.3503	176.085	-6.959	0.000	-1570.487	-880.213
conds_Scattered Clouds	-44.9748	57.565	-0.781	0.435	-157.805	67.855