Predicting Taxi Pickups in New York City

Josh Grinberg, Arzav Jain, Vivek Choksi
Final Paper for CS221, Autumn 2014

ABSTRACT

There were roughly 170 million taxi rides in New York City in 2013. Exploiting an understanding of taxi supply and demand could increase the efficiency of the city's taxi system. In this paper, we present a few different models to predict the number of taxi pickups that will occur at a specific time and location in New York City; these predictions could inform taxi dispatchers and drivers on where to position their taxis. We implemented and evaluated three different regression models: linear least-squares regression, support vector regression, and decision tree regression. Experimenting with various feature sets and performing grid-search to tune hyperparameters, we were able to achieve positive results. Our best-performing model, decision tree regression, achieved a root-mean-square deviation of 33.5 and coefficient of determination (R^2) of 0.99, a significant improvement over our baseline model's root-mean-square-deviation of 146 and R^2 of 0.73.

I. INTRODUCTION AND MOTIVATION

The ability to predict taxi ridership could present valuable insights to city planners and taxi dispatchers in answering questions such as how to position cabs where they are most needed, how many taxis to dispatch, and how ridership varies over time. Our project focuses on predicting the number of taxi pickups given a one-hour time window and a location within New York City. This project concept is inspired by the MIT 2013-2014 Big Data Challenge, which proposed the same problem for taxicabs in Boston. The problem is formulated in terms of the following inputs and outputs:

Input.
Date, one-hour time window, and latitude and longitude coordinates within New York City.
e.g. "17 March 2013, from 5 PM to 6 PM, at coordinates (40.75, -73.97)"

Output.
Predicted number of taxi pickups at the input time and location.
e.g. "561.88 pickups"

II. METHODOLOGY

A. Data Management

We use a dataset detailing all ~170 million taxi trips in New York City in 2013, as provided by the Freedom of Information Law and hosted on the website of Chris Whong [1]. The data associates each taxi ride with information including date, time, and location of pickup and drop-off.

A small number of taxi pickups in this dataset originate from well outside the New York City area. In order to constrain our problem to New York City as well as to reduce the size of our data given our limited computational resources, we only consider taxi trips that originate somewhere within the 28-square-mile rectangular region that encloses Manhattan as shown and defined in Figure 1 below. To further limit the size of our data, we only consider taxi rides in the months of January through April.

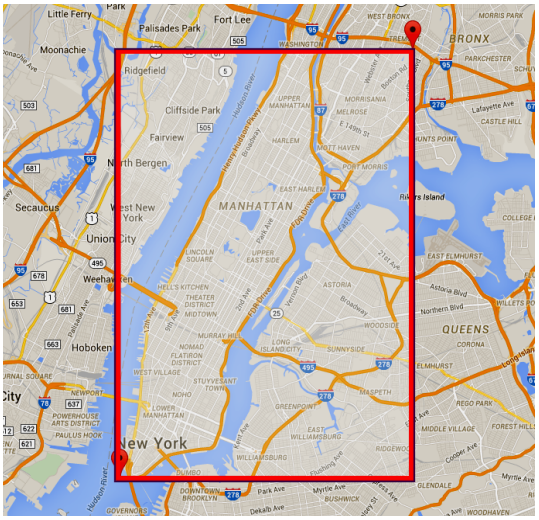

Figure 1. We only consider taxi rides in the portion of

[1] All taxi data is accessible at: chriswhong.com/open-data/foil_nyc_taxi.

New York City between latitudes 40.70° to 40.84° and longitudes -74.02° to -73.89°. This rectangular region encloses the city's densest areas: Manhattan, part of Queens, and part of the Bronx.

We divide our rectangular region of New York City into a grid of 0.01° x 0.01° squares called *zones*. Each zone roughly corresponds to 1km x 1km region.

For ease of querying and aggregation, we store the data in a MySQL database hosted on Amazon RDS.
In order to put the raw data into the same form as our input to the problem, we group the raw taxi data by time (at the granularity of an hour) and zone, count the total number of pickups for each time-zone combination, and store these aggregated values as data points to be used for training and testing. For instance, one row in our aggregated pickups table is "2013-03-05 15:00:00, 15704, 811", representing 811 pickups in Zone #15704 on March 5, 2013 between 3 PM and 4 PM local time. In total, our data set consists of 482,000 such data points.

B. Evaluation

In order to evaluate the performance of our model, we split the data into a training set and testing set, where the training examples are all ordered chronologically before the testing examples. This configuration mimics the task of predicting future numbers of taxi pickups using only past data.

We considered using a few different error metrics to evaluate our predictions: RMSD, mean absolute error, and a root-mean-square percent deviation. We ultimately chose RMSD because it favors consistency and heavily penalizes predictions with a high deviation from the true number of pickups.

From the point of view of a taxi dispatcher, any large mistake in gauging taxi demand for a particular zone could be costly – imagine sending 600 taxis to a zone that only truly requires 400. This misallocation results in many unutilized taxis crowded in the same place, and should be penalized more heavily than dispatching 6 taxis to a zone that only requires 4, or even dispatching 6 taxis to 100 different zones that only require 4 taxis each. RMSD most heavily penalizes such large misallocations and best represents the quality of our models' predictions.

In comparing the results between our different models, we also report the R^2 value (coefficient of determination) in order to evaluate how well the models perform relative to the variance of the data set.

C. Feature Extraction

Below is a list of feature templates we use to extract features from each data point, along with a rough intuition as to why these features might be predictive.

1. *Zone ID*. We expect location to be highly predictive of taxi traffic.
2. *Hour of day* $\in [0, 23]$. We expect overall NYC taxi ridership to follow a daily cycle.
3. *Day of week* $\in [0, 6]$. We expect day of week to correlate with taxi traffic.
4. *Zone and hour of day combined*. Daily patterns in ridership may be different in different zones. For example, at 12 PM, taxi traffic may drop in residential zones (because people are at work) but increase in commercial zones (because workers go out to lunch). Combining zone and hour of day would capture such an inter-feature dependency.
5. *Zone, day of week, and hour of day combined*. Even within a specific zone, the same hour of day may have a different effect during different days of the week.
6. *Hourly precipitation*, measured in hundredths of an inch, provided by the National Climatic Data Center [2], and discretized into 3 buckets representing no rain, less than 0.1 inches of rain in an hour, and at least 0.1 inches of rain in an hour. We expect precipitation to increase taxi ridership, since in rainy weather people may prefer taking taxis to walking or taking public transportation.
7. *Zone, day of week and hourly precipitation combined*. Rainfall may have a different impact on different zones at different times of the day.

All features defined by the feature templates above are binary. For example, the feature "ZONE=15403_DAY=5_HOUR=13", derived from feature template #5 in the list above, has a value of 1 only if the data point represents a taxi pickup in Zone #15403 on Saturday (day 5) between 1 PM and 2 PM.

We did not experiment with quadratic or any other polynomial operations on our numerical features because we did not expect any polynomial relationship between our features and the number of taxi pickups.

[2] All weather data is available at: ncdc.noaa.gov. The weather data we use in our project was observed from the New York Belvedere Observation Tower in Central Park.

D. Regression Models

Baseline Model.

Our baseline model predicts the number of pickups on a test data point at a given zone as the average number of pickups for all training data points in that zone.

To improve upon this baseline, we experiment with three different regression models described below. We use the Python module scikit-learn to apply our regression models.

Linear least-squares regression.

The linear regression model allows us to exploit linear patterns in the data set. This model is an appealing first choice because feature weights are easily interpretable and because stochastic gradient descent runs efficiently on large datasets. We choose squared loss (square of the residual) as our objective function in stochastic gradient descent because minimizing it directly relates to minimizing our error metric, root-mean-square-deviation.

Epsilon Support Vector Regression.

Our feature templates produce a large number of features; for example, the feature template "Zone_DayOfWeek_HourOfDay" alone produces 179 zones × 7 days per week × 24 hours per day → 30,072 binary features. We choose to include support vector regression since support vector machines perform well with many sparse features and can derive complex non-linear boundaries depending on the choice of the kernel.

Decision Tree Regression.

The decision tree regression model is both easy-to-interpret and capable of representing complex decision boundaries, thus complementing our other chosen models. We run the decision tree regression model on a reduced feature set (Feature Set #1 as defined in Table 2) that excludes feature templates containing combination features. We do this for two reasons: (1) training the decision tree model using all 36,649 binary features contained in Feature Set #2 (defined in Table 2) is prohibitively computationally expensive, since each new node in the decision tree must decide on which feature to split, and (2) combination features essentially represent the AND operation applied to multiple binary features; since this AND operation is naturally represented by paths in the decision tree, including combination features would be redundant.

III. RESULTS

Table 1 summarizes the best results for each model, obtained by training on the first 70% of the data points (January 1 through March 28) and testing on the remaining 30% (March 29 through April 30). The hyperparameters used for each model are determined using grid-search over select parameter values, and the best features to use for each model are determined through experimentation.

Model	Root-mean-square Deviation	Coef. of Determination (R^2)
Baseline	145.78	0.7318
Linear Least-Squares Regression (Feature Set #2)	40.74	0.9791
Support Vector Regression (Feature Set #2; trained on 50K randomly selected training examples)	79.77	0.9197
Decision Tree Regression (Feature Set #1)	33.47	0.9858

Table 1. Best results for each model. The feature sets are defined in the Table 2 below.

Feature Set #1	Zone HourOfDay DayOfWeek HourlyRainfall
Feature Set #2	Zone HourOfDay DayOfWeek Zone_HourOfDay Zone_DayOfWeek_HourOfDay
Feature Set #3	Zone HourOfDay DayOfWeek Zone_HourOfDay Zone_DayOfWeek_HourOfDay Zone_DayOfWeek_HourlyRainfall

Table 2. List of the feature templates that compose each feature set.

A. Hyperparameters

Below, we discuss the results of grid search for each model.

Linear

The three hyperparameters tested using grid search were the number of iterations of stochastic gradient descent, η_0, and p, where η_0 and p are parameters of the inverse-scaled learning rate $\eta = \frac{\eta_0}{t^p}$. These parameters are important to the model because they

control the extent to which the model converges to an optimal value. The model converged to an optimum R^2 value of about 0.98 using 8000 iterations of stochastic gradient descent and parameter values $\eta_0 = 0.2$, and $p = 0.4$.

Support Vector Regression

Because training support vector regression on large data set sizes is computationally expensive, we were only able to grid-search one parameter, the regularization parameter *C*. Our model performed best with a high *C* value of 1 x 10^7, indicating that lower values of *C* underfit the data and resulted in too few support vectors. We used a radial basis function kernel in order to introduce nonlinearity; using a nonlinear further increases the computation time with LibSVM. In order to lower computation time, we ran support vector regression on a reduced training set size of 50,000 data points (as opposed to ~337,000 data points for the other models). For reference, training the support vector regression using the full training set did not complete in even 8 hours of running on a Stanford Barley machine using 4 cores. It is likely that support vector regression performed much worse than the other models because of this relatively small training set size, achieving a root-mean-square deviation value of 79.77.

Decision Tree Regression

The two hyperparameters tuned were the maximum depth of the tree and the minimum number of examples that a leaf node must represent. These two parameters are important to the model because they balance overfitting and underfitting. Of the values we swept, our model performed best with a minimum of 2 examples per leaf and a maximum tree-depth of 100. With greater tree-depths, the model achieved the same performance on the test set, suggesting that tree-depths greater than 100 contribute to overfitting.

B. Feature Analysis

Experiments with different feature sets

In order to determine which feature sets produce the best results, we define three feature sets (see Table 2 above). Below are the results obtained running the linear regression model on each feature set.

We observe that feature combinations significantly improve results, as expected. Also, counter to our intuitions, weather features do not improve results (Table 3).

Feature Set	Root-mean-square Deviation	Coef. of Determination (R^2)
Feature Set #1 (Basic features)	138.05	0.7595
Feature Set #2 (Basic + feature combinations)	40.07	0.9797
Feature Set #3 (Basic + feature combinations + precipitation)	40.74	0.9791

Table 3. Results of linear regression using different feature sets.

Analysis of feature weights

In order to better understand the relative importance of our feature templates, we use the linear regression model to generate stacked bar charts of all feature weights that are used to make predictions over the course of a week. That is, at each hour interval in a given zone, we stack the weights of all the features whose values are 1 for that data point. The predicted number of taxi pickups at a given hour can be visualized by adding the heights of all the positive feature weights and subtracting the heights of all the negative feature weights. Since all features are binary, the feature weights map directly to the number of taxi pickups. In other words, if the weight of feature "ZONE=16304" is 126, then this feature contributes +126 pickups to the total predicted number of pickups for this data point.

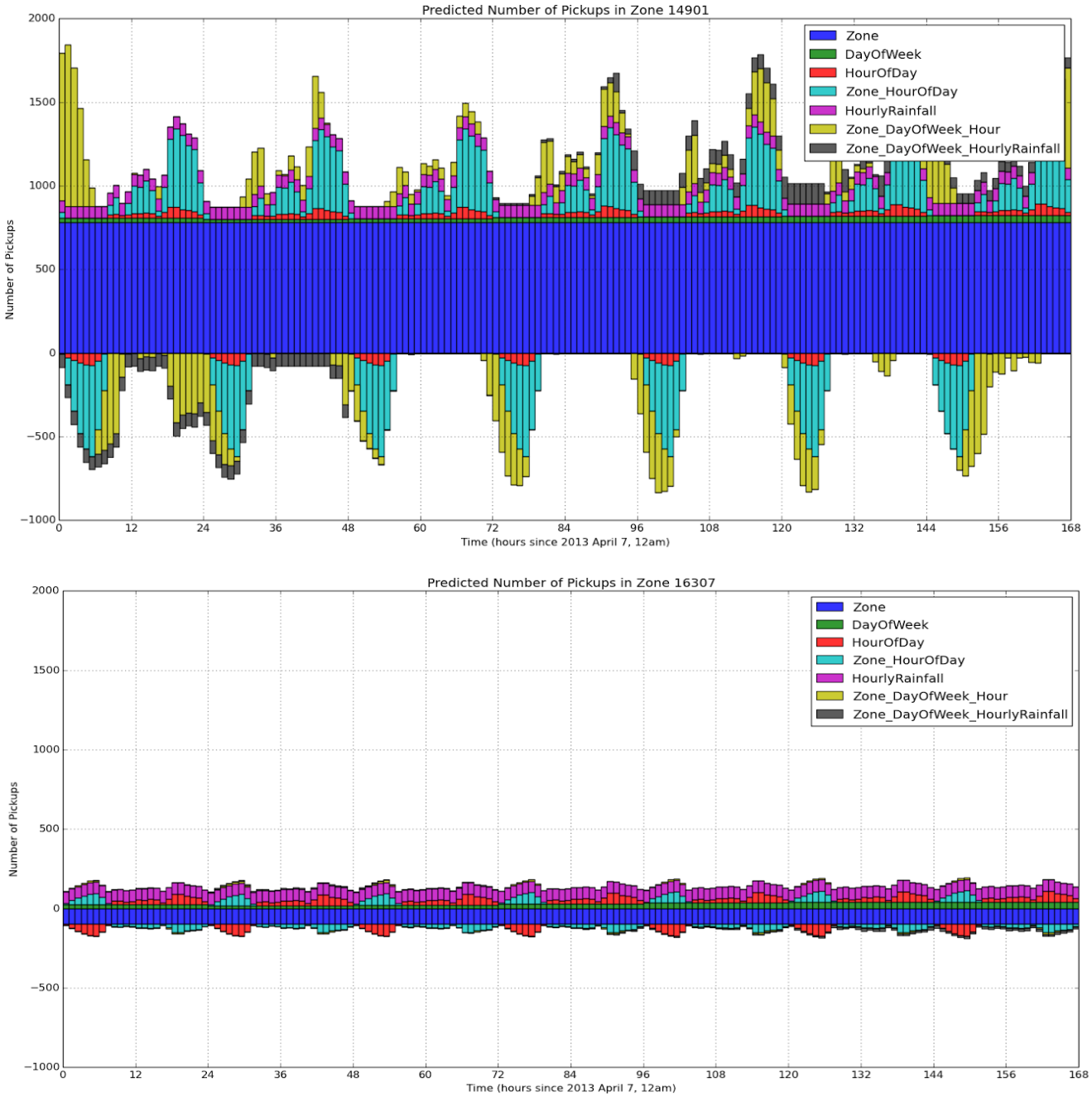

Figure 2. Feature weights used to make predictions at each hour over the course of a week (April 7, 2013 through April 13, 2013), for two different zones in New York City, one busy (Zone #14901, top) and one quiet (Zone #16307, bottom).

The feature weight plots in Figure 2 validate many of the intuitions we used to pick features. For example, we notice small daily cycles common to both zones; this justifies the importance of the 'HourOfDay' feature template (red series). These feature weights represent the daily taxi traffic cycles in New York City across all zones. Furthermore, the 'Zone_HourOfDay' feature weights (cyan series) validate our intuition that daily taxi traffic patterns may vary by zone: notice that 2 PM in Zone #14901 has a higher feature weight relative to other times of day than does 2PM in Zone #16307.

The weights of the weather feature templates present interesting and nuanced results. It rained heavily between April 11 and April 13 (hours 96 through 168). Despite this period of intense rain, the weights of the 'HourlyRainfall' features (magenta series) are roughly constant throughout the week—this suggests that rainfall has little effect on taxi ridership in New York City overall. Now, consider zone-specific effects of rain. From the plots, we see that 'Zone_DayOfWeek_HourlyRainfall' features predict

higher taxi ridership during heavy rain in Zone #14901, but slightly *lower* taxi ridership during heavy rain in Zone #16307. As depicted in the maps in Figure 3 below, Zone #14901 is much more densely populated than Zone #16307 and contains multiple subway stops. Perhaps people in Zone #14901 opt for taxis as opposed to public transportation when it is raining more than people in Zone #16307. As we had suspected, rain appears to have different effects in different zones, validating the usefulness of our combination feature template.

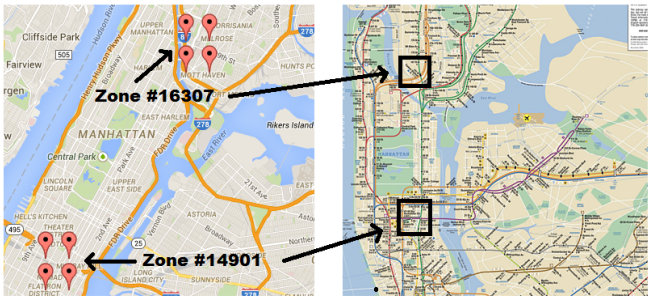

Figure 3. Zones #14901 and #16307 on a road map and a NYC Subway map.

C. Model Analysis

In order to visualize how well the models perform, we plot the true versus predicted number of pickups for each data point in the test set in Figure 4.

The scatter plots in Figure 4 suggest that the linear regression and decision tree regression models perform well on the test set. Most predictions lie close to the true values. The data points straddle the unit-slope line evenly, signifying that the models do not systematically underestimate or overestimate the number of taxi pickups. For both models, as expected, absolute prediction error increases as the true number of pickups increases. This effect can be visualized as a cone-shaped region extending outward from the origin within which the data points fall. The error plot for support vector regression (not shown) looks roughly the same, but with more dispersion of data points.

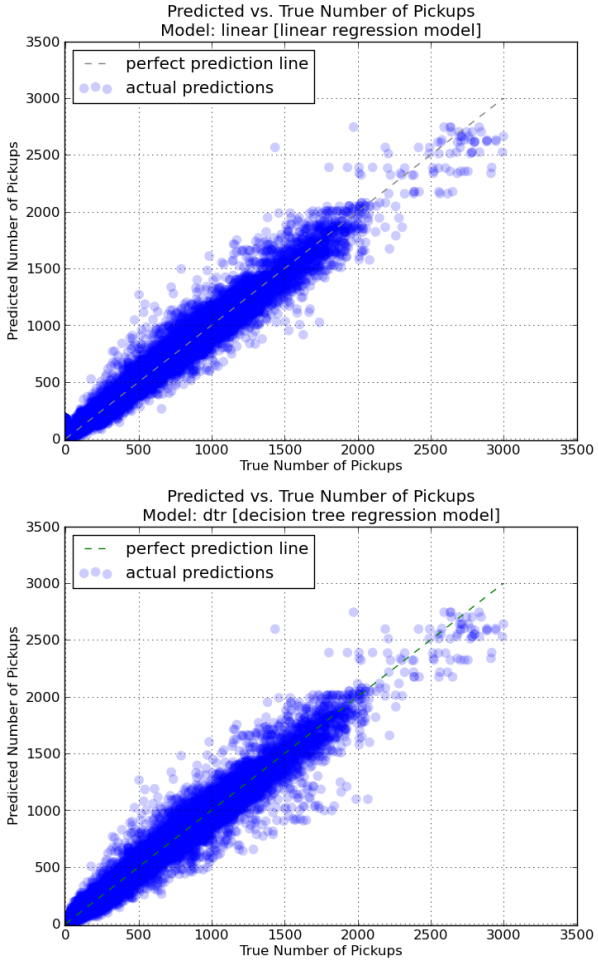

Figure 4. Predicted versus true number of pickups using least-squares linear regression (top) and decision tree regression (bottom).

The baseline model performs very poorly by comparison. This is unsurprising, since the baseline naively predicts the same value for all data points in a given zone, as shown by the horizontal streaks of points in Figure 5.

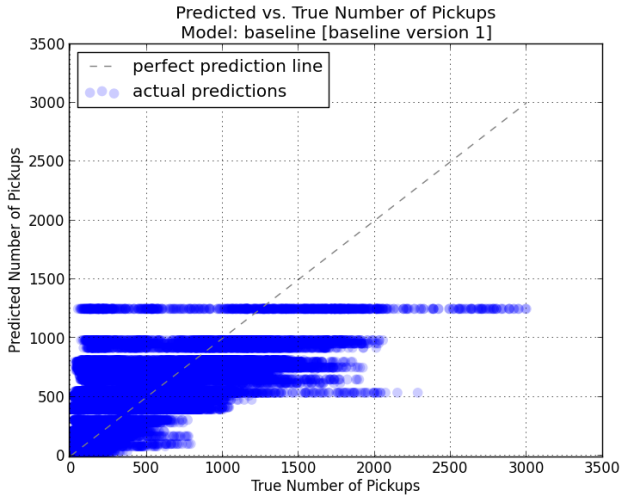

Figure 5. Predicted versus true number of pickups using the baseline model.

Analysis of Decision Tree Regression

Of all of our models, decision tree regression performed best, achieving an RMSD value of 33.47. In order to quantify bias, variance, and the degree to which the model has converged, we plot learning curves, shown below in Figure 6. The learning curves indicate that the decision tree regression model converges using a training set size of as few as 100,000 training examples. Using greater than 100,000 training examples, the test and training R^2 scores are practically identical and well above 0.97, indicating that the model achieves both low bias and low variance.

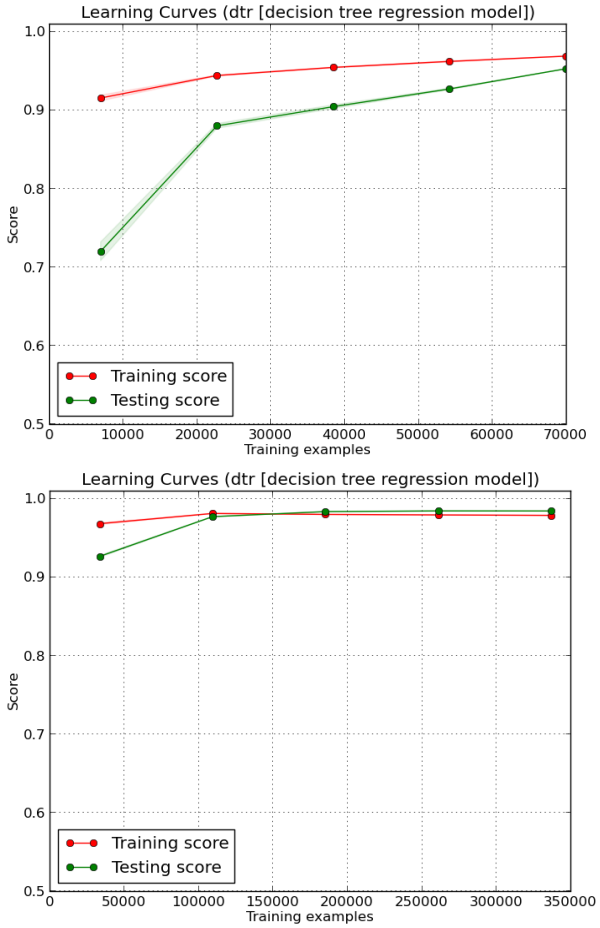

Figure 6. Learning curves for decision tree regression, showing up to 70,000 training data points (top) and up to ~337,000 training data points (bottom).

The tree diagram in Figure 7 shows a subsection of the trained decision tree. Since all of our features are binary, each node in the tree represents one of the 206 features in Feature Set #1 upon whose value the data can be split. Evaluating a test data point using a decision tree can be imagined as asking the data point true-or-false questions until a leaf node is reached. As the tree diagram begins to show, the first question that our tree asks is what zone the data point resides in, since the decision tree has determined that branching on zone provides the most information gain. Once a data point's zone is ascertained, it is next asked for its hour of day, then day of week, and finally amount of precipitation. This ordering of feature templates by descending information gain is consistent with the relative weights of features produced by the linear regression model, shown in Figure 2: zone-based features are most informative, and weather-based features are least informative.

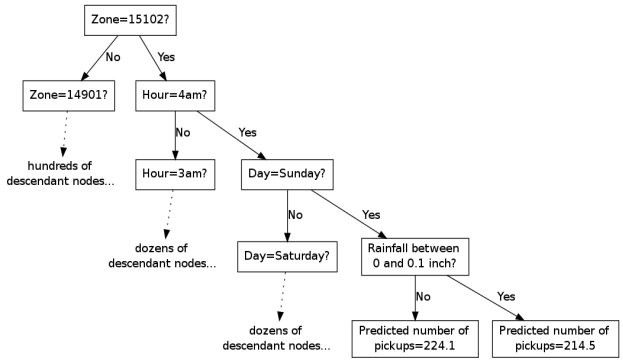

Figure 7. Subsection of the final trained decision tree.

One possible reason this model outperforms linear regression and support vector regression is that although it is run on a smaller feature set, paths within the tree are able to represent the combination of (that is, the AND operation applied to) any number of features in Feature Set #1. For example, the decision tree regression model is able to capture the effect of rainfall on taxi pickups specifically on Sundays at 4am in Zone #15102 (as shown in rightmost paths of the above tree), whereas the other models cannot easily capture such dependencies between features (short of combining all features together). Perhaps it is because these features in Feature Set #1 are highly dependent upon one another that the decision tree regression performs best of all the models we tried.

IV. CONCLUSIONS AND FUTURE WORK

Conclusion

Overall, our models for predicting taxi pickups in New York City performed well. The decision tree regression model performed best, likely due to its unique ability to capture complex feature dependencies. The decision tree regression model achieved a value of 33.47 for RMSD and 0.9858 for R^2 – a significant improvement upon the baseline's values of 145.78 for RMSD and 0.7318 for R^2. Our experiments, results, and error analysis for the most part supported our intuitions about the usefulness of our features, with the exception of the unexpected result that weather features did not improve model performance. A model

such as ours could be useful to city planners and taxi dispatchers in determining where to position taxicabs and studying patterns in ridership.

Future work

Predictions for arbitrary zones and time intervals

Currently, our model predicts pickups only for pre-defined zones and 1-hour time intervals. We could extend our model to predict the number of taxi pickups in arbitrarily sized zones and time intervals. We could do this by training our model using very small zones and time intervals. In order to predict the number of pickups for a larger region and time interval, we could sum our granular prediction values across all mini-zones in the specified region and all mini-time-intervals in the desired time interval.

Neural network regression.

We may be able to achieve good results using a neural network regression, since neural networks can automatically tune and model feature interactions. Instead of manually determining which features to combine in order to capture feature interactions, we could let the learning algorithm perform this task. One possible instance of features interacting in the real world could be that New Yorkers may take taxi rides near Central Park *or* when it is raining, but not when they are near Central Park *and* it is raining, since they may not visit the park in bad weather. Neural networks could be promising because they can learn nonlinearities automatically, such as this example of an XOR relationship between features.

In addition to our three core regression models, we implemented a neural network regression model using the Python library PyBrain. However, we would need more time to give the neural network model due consideration, so we list it here as possible future work.

Clustering feature template.

In order to find non-obvious patterns across data points, we could use unsupervised learning to cluster our training set. The clustering algorithm could use features such as the number of bars and restaurants in a given zone, or distance to the nearest subway station. The cluster in which each data point falls could then serve as an additional feature for our regression models, thereby exploiting similar characteristics between different zones for learning.

V. ACKNOWLEDGMENTS

We earnestly thank Professor Percy Liang and the entire CS221 teaching staff for equipping us with the conceptual tools to complete this project. We give special thanks to Janice Lan for her thoughtful feedback on our project as it matured over the course of the quarter.