Capstone Project

Machine Learning Engineer Nanodegree

Vishakh Rayapeta
Sep 11, 2016

Taxicab Demand Supply Forecasting using Spatiotemporal Data

1. Definition

Project Overview

Spatial data is represented by numerical values in a geographical coordinate system. Typically spatial data includes an input feature that is an areal unit or a point in space. *Temporal data* (also called time series data) refers to a sequence of data points in time. Typically temporal data includes an input feature that is a time stamp. In this project, we will focus on the problem of using spatiotemporal data for *demand & supply forecasting for taxicabs*.

Viability and quality of service in the taxicab business is significantly influenced by the implementation of an effective dispatch operation that avoids problems of unmet demand or oversupply[3]. City planners also want to improve transportation options to efficiently route vehicles and reduce congestion[4].

The dataset used in this project was provided by Didi Chuxing for an algorithm competition[1]. Didi Chuxing is a ride hailing company in China which processes over 11 million trips per day.

Problem Statement

Taxicab demand varies based on location and time. For example, demand surges are observed in the residential districts during the morning commute hours and in the business districts during the evening commute hours. The goal is to predict the gap between volume of drivers (supply) and riders (demand) in a certain time period within a specific geographic area (district).

The dataset includes taxicab data for one city. The city is divided into 'n' non-overlapping districts $\{d_1, d_2, \ldots, d_n\}$. Each day is divided into 144 time slots $t_1, t_2, \ldots, t_{144}$, where each time slot is 10 minutes long.

For district d_i and time slot t_j the following terms are defined:

- r_{ij}: Number of passenger requests (Demand)
- a_{ij}: Number of driver answers (Supply)
- gap_{ij} : $gap_{ij} = r_{ij} - a_{ij}$ (Demand Supply Gap)

We will define the train, validation and test sets for this project as follows:

- Training data: First two weeks of data
- Validation data: Use k-fold cross-validation (reuse training data)

- Test data: Third week data.

The above split results in a roughly 67%, 33% train, test split.
Test data provides information for the 30 minutes (3 time slots) before the 10 minute time slot that needs to be predicted. Given the data for every district and time slot t_j,t_{j-1},t_{j-2}, we need to predict $gap_{i,j+1}$, $\forall d_i \in D$.

The following task list provides an overview of the work involved in this project:

1. Load and process data files [data_processing.ipynb]
    a. Load data sets from files
    b. Perform common processing tasks:
        i. Handle missing data, categorical data
        ii. Perform feature reduction if needed
    c. Save processed data to a pickle file
2. Generate statistics, visualizations to analyze the data [data_exploration.ipynb]
3. Train regression models [implement_basic.ipynb]
    a. Split data into train & test sets
    b. Scale features if needed
    c. Choose relevant learning algorithms
        i. Linear models: Linear regression, SVM (linear kernel)
        ii. Non-linear models: Random Forests, Gradient Boosting, K-Nearest Neighbors, Neural Nets (feed forward with ReLU activation, with and without hidden layers)
    d. Perform the following tasks for selected algorithms:
        i. Estimate feature importance or ranking
        ii. Use learning curves to select a subset of features that delivers good performance without overfitting
        iii. Use validation curves to select a good range/value for each hyper parameter
        iv. Select hyper parameters using grid search. Generate algorithm scores for the train and test data sets
4. Review results and attempt methods to improve performance

Metrics

We will use three metrics (MSE, R^2, MAPE) to measure model performance. Using multiple metrics allows a more comprehensive comparison of the models. The metrics are defined for district d_i at time slot t_j having an actual supply-demand gap gap_{ij} and model prediction $gapX_{ij}$.

Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{d_i,t_j}(gap_{ij} - gapX_{ij})^2$$

MSE provides the expected value of the quadratic loss. It incorporates both the variance and bias of the model. This metric leads to a minimum variance unbiased estimator. One drawback of this metric is that it weighs larger errors more heavily than smaller ones. In this project, using this metric alone will cause the model to focus primarily on districts with higher taxicab volumes.

Coefficient of determination (R^2)

$$SS_{tot} = \sum_{d_i,t_j}(gap_{ij} - \overline{gap})^2$$

$$SS_{res} = \sum_{d_i,t_j}(gap_{ij} - gapX_{ij})^2$$

$$R^2 \equiv 1 - \frac{SS_{res}}{SS_{tot}}$$

R^2 provides a measure of how well the observed outcomes are replicated by the model. An R^2 of 1 indicates that the regression line perfectly fits the data.

Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n}\sum_{d_i}\left(\frac{1}{q}\sum_{t_j}\left|\frac{gap_{ij} - gapX_{ij}}{gap_{ij}}\right|\right) \quad \forall gap_{ij} > 0$$

n, q are constants and are set to number of districts (66), total number of time slots - 1 respectively. This was the metric used during the Didi competition.
MAPE is used as a measure of prediction accuracy in trend estimation and attempts to normalize the error value. Typically lower MAPE scores indicate a better fit.
There are two drawbacks with using MAPE:

- It cannot be used with zero values. We will use metrics only for $gap_{ij} > 0$.
- The metric has a bias towards models whose forecasts are too low.

2. Analysis

Data Exploration

The dataset is organized into the following 5 categories:

- Order
- Cluster Map
- POI (Points Of Interest)
- Traffic
- Weather

Data for each of these categories is available as a separate directory with one or more files.

Feature Discussion [data_processing.ipynb]

Order

Field	Type	Meaning	Example

order_id	string	order ID	70fc7c2bd2caf386bb50f8fd5dfef0cf
driver_id	string	driver ID	56018323b921dd2c5444f98fb45509de
passenger_id	string	user ID	238de35f44bbe8a67bdea86a5b0f4719
start_district_hash	string	departure	d4ec2125aff74eded207d2d915ef682f
dest_district_hash	string	destination	929ec6c160e6f52c20a4217c7978f681
Price	double	Price	37.5
Time	string	Timestamp of the order	2016-01-15 00:35:11

The Order data provides information of a taxicab request as described in the table above. When a request is unanswered, driver_id field has a NULL value. This indicates an unfilled request. Some of the fields have been anonymized to hide sensitive information.

Cluster Map

Field	Type	Meaning	Example
district_hash	string	District hash	90c5a34f06ac86aee0fd70e2adce7d8a
district_id	string	District ID	1

Cluster map data maps the district hash value to a numerical district ID (1 to 66).

POI (Points Of Interest)

Field	Type	Meaning	Example
district_hash	string	District hash	74c1c25f4b283fa74a5514307b0d0278
poi_class	string	POI class and its number	1#1:41 2#1:22 2#2:32

POI (Points Of Interest) data attempts to identify the different points of interest in each district. For example, 2#1:22 indicates that there are 22 points of interest of category 2#1. 2#1 indicates a point of interest that belongs to sub-category #1 within category #2. Examples could include entertainment#theater, shopping#home appliance, sports#others.

Traffic

Field	Type	Meaning	Example
district_hash	string	Hash value of the district	1ecbb52d73c522f184a6fc53128b1ea1

tj_level	string	Number of road sections at different congestion levels	1:231 2:33 3:13 4:10
tj_time	string	Timestamp	2016-01-15 00:35:11

Traffic data identifies the traffic levels for road intersections in a district. The traffic level identifies the number of roads at different traffic levels. Higher values indicate heavier traffic compared to lower values which indicate lighter traffic.

Weather

Field	Type	Meaning	Example
Time	string	Timestamp	2016-01-15 00:35:11
Weather	int	Weather	7
temperature	double	Temperature	-9
PM2.5	double	pm25	66

Weather data is updated every 10 minutes and represents the conditions for the entire city. The weather field uses a numerical value to identify conditions such as sunny, rainy, snowy etc. The temperature reading provides the reading in the Celsius scale. PM2.5 is the level of air pollution.

Statistics [data_exploration.ipynb]

We will employ two methods to identify features that are going to be more useful for gap prediction.

Correlation (Gap vs Input Features)

The Pearson standard correlation coefficient identifies linear relationships between target and input features. The following features have the highest linear correlation with gap prediction: demand(65+%), supply(45+%), POI(36%) & traffic(33+%).

Feature Importances (Random Forest Regressor)

For the remaining features, we attempt to identify non-linear relationships using the Random Forest Regressor. The following features were identified as having high feature importances: time slot, pollution, temperature & weather.

Abnormalities [data_processing.ipynb]

Dimensionality Reduction

The POI data provide 176 POI types which is a very large number of input features for just one category. Investigate dimensionality reduction for this feature.

Categorical Data

The following features are available as categorical data:
district_id, num_day, time_slot, week_day, poi_cluster, weather.
Investigate binary encoding for these features.

Missing Data

Traffic & Weather data have missing information for some time slots.

Exploratory Visualization [data_exploration.ipynb]

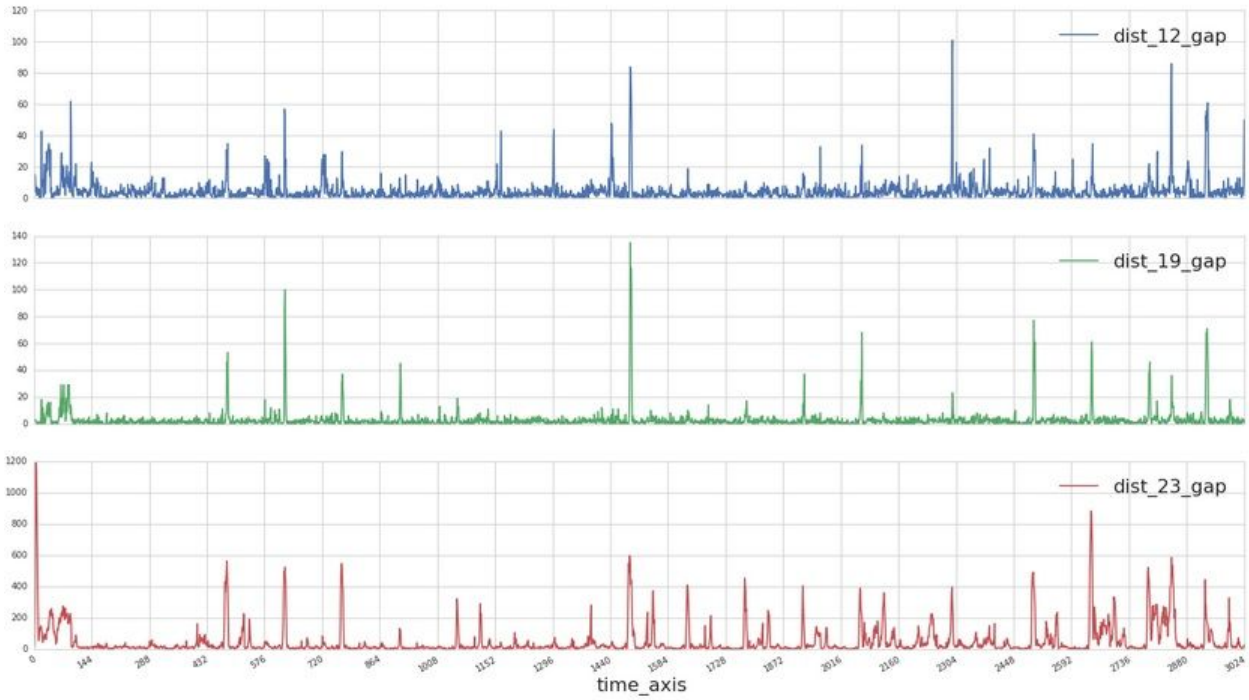

Figure 1: Time Series Plot- Gap

Figure 1 displays the time series plot for the gap value. The 3 districts (12, 19, 23) are the cluster centroids extracted from the POI data. Each tick of 144 along the time axis corresponds to one day. The plot extends to the entire 21 day period for which we have data.
We observe the following:

- Maximum observed gap value is specific to the district. Districts 12, 19 have maximum gap values in the 100-140 range. District 23 has maximum gap value in the 800 to 1000 range.
- Some of the gap values can be explained by expected commute habits. For example: Notice the longer spikes for dist_23_gap between time 432 and 864. These seem to coincide with commute hour demand during weekdays.

- However, the extent and occurrence of the spikes in gap is not easily replicated each week. Notice for district 23: While the gap plot shows some similarities between each week, there are significant differences. Clearly, this is a problem for machine learning algorithms to investigate.
- The data for the first day (0 - 144) is for New Years Day. Notice the high gap values during early morning hours. This is presumably caused by people returning from their late night celebrations.

Scatter Plot - Gap vs Input Features

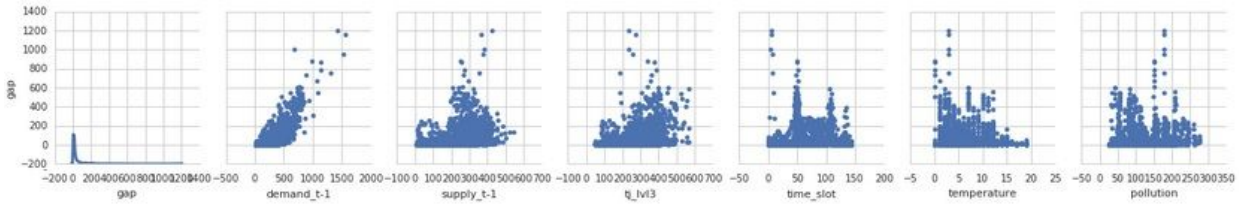

Figure 2: Scatter Plot - Gap vs Input Features

The scatter plot (figure 2) shows the relationship between the target value (gap) and selected input features for district 23.

- Gap values are distributed primarily between 0 and 600 with a tail that extends to ~1200
- Positive correlation with demand is seen.
- When demand rises above 750, gap values get sparser and display a clearer positive correlation. This indicates that spikes in demand correlates directly with spikes in gaps.
- Correlation with supply & traffic seems mostly positive but isn't as clearly linear as the demand plot. Peak gap values occur when supply & traffic are closer to mid range.
- The scatter plot of gap with time_slot shows us two things:
    - Gap spikes are observed during late night hours (time_slot = 0) - mostly due to a short supply of cabs.
    - Consistently high gap values are observed around the peak morning & evening commute hours (time_slots ~50, ~100).
- Gap correlation with temperature seems to be negative. Probably because some of the gap spikes occur during early morning hours (when the temperature is lower).
- Correlation with pollution is less clear.

Algorithms and Techniques [implement_basic.ipynb, implement_nn.ipynb]

The input features for this project exhibit linear and non-linear relationships with the target (gap) that needs to be predicted. We will employ a number of algorithms that are based on different underlying approaches to explore a possible solution. Tree based methods and KNN provide

non-linear approaches while the linear regression and SVM (linear kernel) provide linear approaches.

Random Forest Regressor

Random Forests implement techniques to average many noisy but approximately unbiased models to reduce the variance. Trees are ideal candidates since they can capture complex interactions in the data. Each tree in the ensemble is built using a bootstrap sample from the training set. When splitting a node during the construction of the tree, the best split is picked among a random subset of the features.

Hyper Parameters

- Number of estimators: The number of trees in the forest.
- Maximum features: Percentage of features to consider at each split.
- Maximum depth: The maximum depth to which a tree can be grown.
- Minimum samples per split: Minimum number of samples to consider before splitting a node.
- Minimum samples per leaf: Minimum number of samples that a leaf must contain.

Gradient Boosted Trees

Boosting methods build base estimators (such as trees) sequentially in order to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble. Gradient boosting employs arbitrary differentiable loss functions.

Hyper Parameters

- Learning rate: Contribution of each tree is shrunk (multiplied) by the learning rate.
- Number of estimators: Number of boosting stages.
- Maximum depth: The maximum depth to which a tree can be grown.
- Minimum samples per split: Minimum number of samples to consider before splitting a node.
- Minimum samples per leaf: Minimum number of samples that a leaf must contain.
- Subsample: The fraction of samples to be used for individual trees. Values less than 1.0 indicate the use of stochastic gradient boosting.

K-Nearest Neighbors (KNN)

Neighbor based methods use a predefined number of training samples closest in distance to the query point that needs to be predicted. They are often successful in situations where the decision boundary is very irregular. The label assigned to a query point is computed based the mean of the labels of its nearest neighbors.

Hyper Parameters

- Number of neighbors: The number of neighbors to use.

- Weights: The weight function use (uniform weight for all neighbors or distance based)
- Metric: The distance metric used by the tree.

Support Vector Machines (SVM)

SVMs construct one or more hyperplanes in high dimension space to perform classification, regression or other tasks. In order to lower generalization error, the hyperplane that has the largest distance to the nearest training-data point of any class is selected. With the help of an appropriate kernel, it may be possible to map the original finite-dimensional space into a much higher-dimensional space. This may make the separation easier in that space.

Hyper Parameters

- Kernel: Kernel type to be used by the algorithm.
- C: Penalty parameter.
- Epsilon: Epsilon parameter for the epsilon-SVR model.
- Cache Size: Specify the size of the kernel cache in memory (MB). Higher values enable faster runtime performance.

Linear Regression

This method fits a linear model with coefficients. The cost function attempts to minimize the residual sum of squares between the true and predicted responses.

Hyper Parameters

- Intercept: Select whether to calculate an intercept for the model or not.
- Normalize: Select whether to normalize input features or not.

Neural Networks

Feed forward neural networks with a Rectified Linear Unit (ReLU) activation function will be employed. I will use TensorFlow to build networks with zero, one or more hidden layers. Since the complexity & cost of training the network increases with the number of hidden layers, I will stop adding layers when additional layers do not lead to a significant increase in performance. The cost function minimizes the residual sum of squares between the true and predicted responses.

Hyper Parameters

- Learning Rate: Rate at which the network should adapt to the calculated errors.
- Number of Steps: Number of passes through the entire training set.
- Dropout: Fraction of units to drop - used to create a more robust network and reduce overfitting.
- Batch Size: The size of individual batches for batch gradient descent.

Benchmark [implement_basic.ipynb]

Algorithm	MEAN^2	R2	MAPE
Linear Regression (Benchmark)	331.6	0.864	0.541

Table A: Scores from using a linear regression with 4 variables. This will be the benchmark for this project.

We will use a linear regression with 4 variables (demand_t-1, supply_t-1, demand_t-2, supply_t-2) as our benchmark. This algorithm provides a reasonable result and runs extremely fast. We will use this benchmark to estimate the effort, runtime incurred when adding features or using more complex methods to obtain improvements in gap prediction. To be considered better than the benchmark, the new model must improve the MAPE score and atleast one other score (Mean Squared Error or R2).

3. Methodology

Data Preprocessing [data_processing.ipynb]

POI Dimensionality Reduction

The POI data provide 176 POI types which is a very large number of input features for just one category. Using PCA, we can reduce the POI data to fewer dimensions. 72% of the POI variance can be captured using 4 PCA components.

Demand, Supply & Gap Calculation

Use order data to calculate demand, supply & gap calculations as follows:

- Demand = Number of orders received (ride requests)
- Supply = Number of orders answered (driver_id field has non-NULL value)
- Gap = Number of orders missed (driver_id field has NULL value)

Missing Data

Traffic & Weather data have missing information for some time slots. In both situations, we use the last available data until data becomes available.
Traffic data is completely missing for district 54. We substitute traffic data from district 39 to handle this. District 54 is part of the POI cluster that has district 39 as its centroid.

Create Unified Dataframe

Merge all the features (weather, traffic, demand, supply, gap, POI) so that they can be accessed by district id & time. The data is accessed for each district on the time axis (21 days each split in to 144 time slots).

Column Name	Function
district_id, weather, temperature, pollution	Self explanatory
num_day	Day (1 to 21)
week_day	Week day (0 to 6 for Monday to Sunday)
Time_slot	Time Slot (1 to 144)
demand_t-1, demand_t-2, demand_t-3	Demand values for three previous time slots
supply_t-1, supply_t-2, supply_t-3	Supply values for three previous time slots
poi_pc1, poi_pc2, poi_pc3, poi_pc4, poi_cluster	First 4 PCA components, POI cluster id
tj_lvl1, tj_lvl2, tj_lvl3, tj_lvl4	Number of roads at traffic levels 1 to 4
gap	**Gap value for current time slot (Target value needs to be predicted)**
demand, supply	*Demand, supply for current time slot (not provided as a feature - used for gap calculation)*

Table1: Explanation of the columns in the unified dataframe

Handle Categorical Data

The following features are available as categorical data:
district_id, num_day, time_slot, week_day, poi_cluster, weather.
We will encode these features using binary encoding (NOT one hot) to ensure that input feature count stays low. For example: time_slot = 5 is encoded as binary 101.

In hindsight, I could have encoded features with fewer categories in both binary and one-hot versions. Feature selection methods can then be used to determine which encoding performs better for each individual algorithm. This alternate approach would allow the algorithm to pick the encoding scheme.

Train / Test Split [implement_basic.ipynb]

We will use weeks 1 & 2 of the data for training. Week 3 data is reserved for test. This results in 67% / 33% train / test split.

Implementation [implement_basic.ipynb]

Overview

For each selected algorithm, the following steps are performed:

- Estimate feature importance or ranking
- Use learning curves to select a subset of the highest ranked features
- Use validation curves to identify values/ranges for hyper parameters
- Perform grid search for final hyper parameter selection using ranges identified by validation curves
- Report scores for both train & test sets using the best estimator

Function Name	Purpose
gap_estimate print_score mape_score	Compute and print scores for training & test data. The following scores are reported: Mean Square Error, R2 and MAPE.
generate_learningcurves plot_learning_curve	Plot performance (training vs CV score) of algorithm as a different percentage of the training data set is used.
plot_validation_curve	Plot performance (training vs CV score) of algorithm as a different value for the hyper parameter is used.
select_hyperparams	Perform hyper parameter selection using grid search with 5x2 cross validation. Generate learning curves and scores for the best estimator.

Table 2: Functions used for implementation

Algorithm Selection [implement_basic.ipynb]

- Linear regression is a simple algorithm with fast runtimes that was used as a baseline / benchmark.
- However, gap plots indicate nonlinearity. So I evaluated non-linear models (random forests, gradient boosting, k-nearest neighbors, neural networks).
- Tree based methods like random forests and gradient boosting are relatively simple to train. I used validation curves with cross validation scores to ensure that they did not overfit the training data. However, overfitting was still observed and could not be fully eliminated.

- SVMs support both linear and non-linear kernels. However, my initial runs showed that the linear kernel outperformed the radial basis function (rbf) kernel. This meant that SVMs could not be used to capture nonlinear behavior which limited their final performance.
- K-nearest neighbors delivered reasonable performance. However, during the refinement stage they were outperformed by tree based and neural net algorithms.
- Neural networks can potentially 'engineer' or discover features by combining the original input features. I used feed forward, fully connected neural nets with zero to 3 hidden layers.

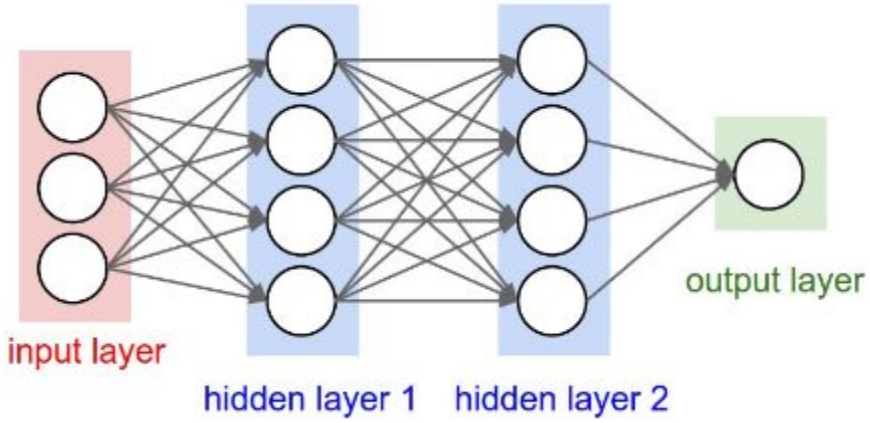

Figure 3: Fully connected neural network with two hidden layers.

- In my implementation, the size of the hidden layers matches the size of the input layer (number of features). Using fewer hidden layers reduces the complexity and training time for the network. Using more hidden layers allows the network to combine features in more complex ways and possibly engineer useful features that deliver higher performance.
- I stopped adding layers when the additional layer did not lead to a significant increase in performance. The neural network with 3 hidden layers provided the best performance and runtime trade-off.
- I attempted some experiments with hidden layers that were wider (2X size of input layer). However, these experiments did not yield significant improvements.

Challenges [data_processing.ipynb]

- The first challenge in this project was that different types of data originated from different files. Each file presented data in a different format using a different index(key). Before applying any ML algorithm, I needed to create a unified dataframe that integrated all these different features with the target values. Defining the structure for this unified data frame (using the time_slot, district_id as indices) was a significant milestone.

- The large amount of data meant that I could not simply 'eyeball' it to find issues like missing data. Each feature was reviewed to find NULL values (pd.null). Missing values for traffic & weather data were then extrapolated using the last available information.
- The original POI data had 176 categories which presented a dimensionality challenge. Using PCA, I was able to capture 72% of the original POI variance with just 4 components.

Refinement

Overview

The refinements attempted in this section are mostly based on the exploration of the initial results. The following is a summary of the refinements attempted:

- **Residual Analysis:** We will perform residual analysis for the initial results using various methods (correlation matrix, residual plots etc). The initial run did not include all the features when training the models. One goal is to determine if any information from features that were not included could help improve the residual. Once we identify features that need to be included, we will add them back in and regenerate the scores.
- **Feature engineering:** Most algorithms perform better in the presence of data that is closer to a normal distribution. However, the gap data plot shows a long tail distribution. One of the refinement attempts will use the square root of the gap values as a feature in order to reduce its long tail and bring it closer to a normal distribution.
- **Parameter Tuning + Feature reduction:** Once new features have been added, we will perform a second round of parameter tuning for the best performing algorithms. We will also check to see if overfitting can be addressed by dropping one or more features. The goal is to identify the best set of features for each algorithm that delivers good performance while minimizing overfitting.
- **Ensembling:** Finally, we will create an ensemble using the best performing algorithms. The goal is to explore the possibility of combining results from diverse algorithms to yield better results. Our goal is not just to get the highest test score - We want to reduce overfitting and demonstrate stable results during multi-fold cross validation.

Initial Results [implement_basic.ipynb]

The table below summarizes the results from using the basic algorithms after parameter tuning.

Algorithm	MEAN^2	R2	MAPE
Random Forests	315.3	0.864	0.371
Gradient Boosting	389.1	0.832	0.481
K-Nearest Neighbors	405.4	0.825	0.419

Support Vector Machines	301.1	**0.870**	0.412
Linear Regression (Benchmark)	**300.0**	**0.870**	0.546
Neural Nets (no hidden layers)	513.8	0.778	0.641
Neural Nets (one hidden layer)	336.9	0.854	0.371
Neural Nets (two hidden layers)	321.2	0.861	0.328
Neural Nets (three hidden layers)	311.2	0.865	**0.318**

Table 3: Test Scores for Basic Algorithms. Best scores are in bold & underlined.

Residual Analysis [residual_analysis.ipynb]

Four methods were used to understand the residual:

- Correlation matrix
- Residual & Scatter plot
- Facet plots
- Feature importances (Random Forests)

The analysis resulted in the following discoveries:

- The original (non-binary) time_slot, week_day features exhibit a significant non-linear relationship with the residual. Facet plots and feature importances identified this behavior.
- Neural nets have a residual that has significant linear correlation with POI and traffic. This was indicated by the correlation matrix.
- Gradient Boosted Trees display the best scatter plot with a Pearson correlation of 0.98.

Refinement Attempts

(A) Feature Engineering (Include Gap) [refine_addgap.ipynb]

Since gap is the target variable, we will add the gap values for previous time slots as new features.

Algorithm	MEAN^2	R2	MAPE
Random Forests	303.5	**0.869**	0.347
Gradient Boosting	352.3	0.848	0.372
K-Nearest Neighbors	344.5	0.851	0.384
Support Vector Machines	**302.2**	**0.869**	0.412

Linear Regression	331.6	0.864	0.541
Neural Nets (three hidden layers)	372.5	0.839	**0.302**

Table 4: Scores in green show improvement from previous run. Best scores are in bold & underlined.

Results for all algorithms (except linear regression, SVM) show improvement.

(B) Feature Engineering (Square Root) [refine_sqrt.ipynb]

The gap distribution plot indicated a long tail to the right side. We will apply the square root function to the gap target and demand & supply features in order to bring their distributions closer to a normal gaussian distribution.
The results from using the square root version of gap, demand & supply features do not improve on the run that used the non-square root gap values.

(C) Include original time_slot, week_day features [refine_addtimeweek.ipynb]

Residual analysis indicated that the influence for time_slot, week_day features was not included.
We will use the original time_slot, week_day features and rerun all the algorithms.
This experiment will be performed along with including the gap values for previous time slots (A).

Algorithm	MEAN^2	R2	MAPE
Random Forests	**281.3**	**0.878**	0.340
Gradient Boosting	282.4	**0.878**	0.359
K-Nearest Neighbors	339.2	0.853	0.377
Support Vector Machines	302.1	0.869	0.414
Linear Regression	313.4	0.864	0.546
Neural Nets (three hidden layers)	343.5	0.851	**0.301**

Table 5: Scores in green indicate improvement from previous run (A). Best scores are in bold & underlined.

Results for most non-linear algorithms show improvement. Rest have results similar to (A).

(D) Final Improvements [refine_paramtune.ipynb, refine_featred.ipynb, refine_nn3.ipynb]

We apply multiple methods to improve the best performing algorithms:

- More parameter tuning using validation curves, grid search
- Experiments to determine an optimal feature set. We will drop or include unused features iteratively to find the set of features that improves results.

- Dropping supply, traffic, week_day improved random forests.
- Dropping supply, POI and adding district_id improved gradient boosting.
- Dropping supply and reducing dropout to 0.01 improved neural nets.

Algorithm	MEAN^2	R2	MAPE
Random Forests	274.6	0.881	**0.338**
Gradient Boosting	**250.0**	**0.892**	0.339
K-Nearest Neighbors	338.0	0.854	0.381
Neural Nets (three hidden layers)	301.5	0.870	0.346

Table 6: Scores in green indicate improvement from previous run (A). Best scores are in bold & underlined.

(E) Ensembling [refine_ensemble.ipynb]

We will build two ensembles that average the results from the best performing models: Random Forests, Gradient Boosting and Neural Nets.

Algorithm	MEAN^2	R2	MAPE
Ensemble (RF,GB,NN)	259.0	0.888	**0.327**
Ensemble (RF,GB)	253.0	0.891	0.336

Table 7: Scores in green indicate improvement from previous run (A). Best scores are in bold & underlined.

We find that the performance of both ensembles is very close to the best performing gradient boosting model.

4. Results

Model Evaluation and Validation [model_eval.ipynb]

Two types of sensitivity analysis were performed on the best performing models.

Cross Validation Curves [model_eval.ipynb]

10 fold CV was used to measure the sensitivity of the algorithm to the subset of the data used for training / CV. All scores listed below are MSE.

Algorithm	Test Scores (Mean, Std)	CV Scores (Mean, Std)

Random Forests	138.7, 7.8	277.4, 134.5
Gradient Boosting	36.5, 1.0	163.4, 44.9
Neural Nets	155.4, 2.6	**157.6, 10.4**
Ensemble (RF,GB,NN)	83.7, 1.8	169.3, 38.0

Table 8: Mean and Standard Deviation for CV scores. Best scores are in bold & underlined.

We notice that the standard deviation in CV scores is smallest for neural nets which indicates the highest stability and best fit to the data.
Random Forests, Gradient Boosting display a significant divergence between the test and CV scores. This indicates the presence of overfitting with these algorithms.

Perturbation Analysis [model_eval.ipynb]

Random normal noise (mean=0.0, std=0.25) was added to the input data during training (test data was not touched). The scores with and without noise were compared.
Once again the neural net algorithm showed the least sensitivity to noise and highest stability.

Justification

Algorithm	MEAN^2	R2	MAPE
Linear Regression (Benchmark)	331.6	0.864	0.541
Ensemble (RF,GB,NN)	**259.0**	**0.888**	**0.327**

Table 9: Ensemble model vs Linear Regression(benchmark). Best scores are in bold & underlined.

We will select the ensemble of the three best models as the final model since it provides good performance with reasonable stability. This model delivers the best results on the test data and improves on all 3 metrics compared to the linear regression benchmark (Table 9).
Our goal was to improve the MAPE metric and atleast one other metric (Mean Squared Error or R2) to be considered better than the benchmark. The ensemble beats the benchmark on all 3 metrics.

5. Conclusion

Free-Form Visualization [model_eval.ipynb]

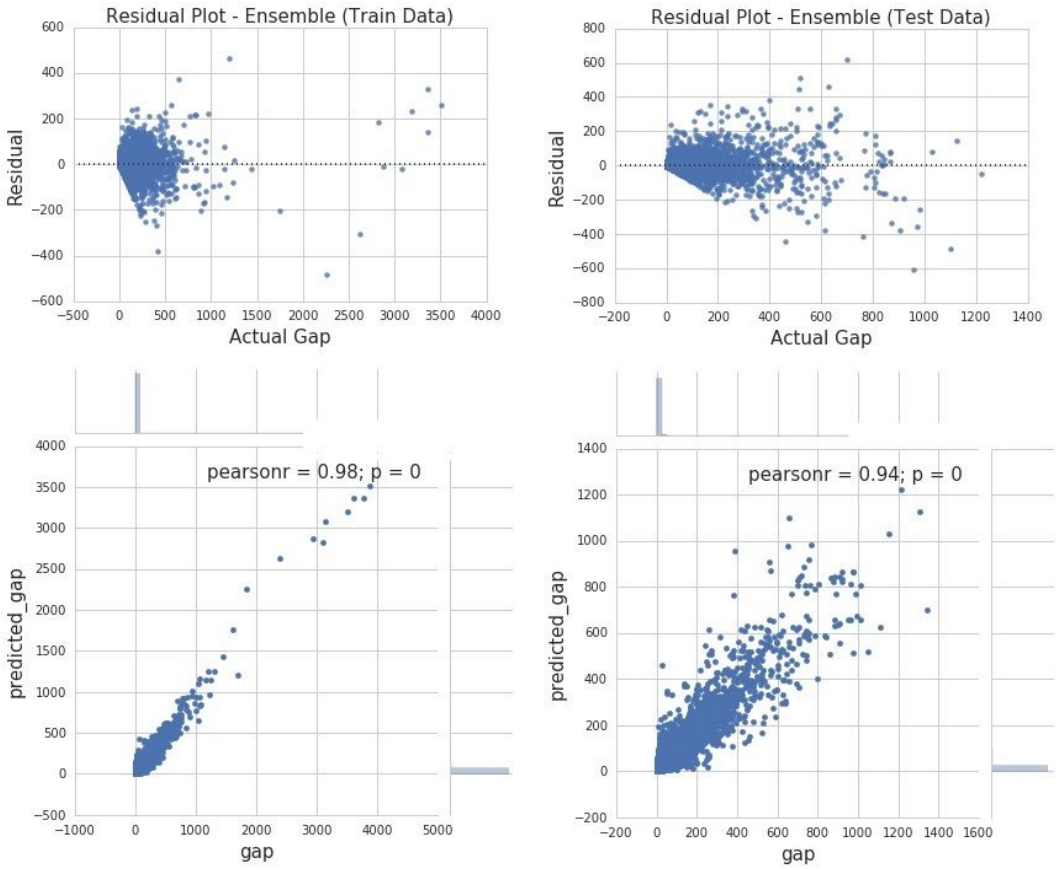

Figure 4: Residual & Scatter Plots compare the residual and predicted gap to the actual (true) gap values.

Residual and scatter plots compare performance of the ensemble model for the train (left) & test (right) data. We notice the following:

1. Residual is essentially random for both train & test data. The pearson correlation indicates strong fit with train data (0.98) and moderate fit with test data (0.94).
2. The gap range for test data is much smaller with max value ~1400. While the train data has a max value ~4000.
3. In addition to overfitting, this smaller range could be one contributing factor for the difference between train & test scores of the algorithm.
4. Residual analysis (see model_eval.ipynb) indicates that non-linear correlation with time_slot, week_day is not fully captured with the final models. Incorporating this information will most likely improve performance further.

Reflection

The following steps were performed in this project:

- Process input data to create input features
- Explore properties of the input and target data using statistics and visualizations
- Implement basic algorithms and tune parameters using validation curves, grid search
- Perform residual analysis using correlation matrix, residual & scatter plots
- Refine results by using the insights from residual analysis

Virtual Computing using Amazon EC2

Due to the size of the dataset involved in this project I made use of the virtual compute resources available from Amazon EC2. The c4.8xlarge instance (36 CPUs, 60GB RAM) enabled me to speed up results and attempt multiple experiments.

Parameter Tuning & Refinement Process

Even with the use of powerful compute resources, a brute force grid search for parameter tuning was not feasible. So I used validation curves to narrow down the parameter range for use in grid search. Residual analysis for refinement was an important insight during this project.

Improvement

Data - Quantity & Quality

- Learning curves for the tuned algorithms indicate that there is room for performance improvement with more training data. Availability of more data will most likely improve performance.
- Since the data was made public, many aspects of it were anonymized. It is possible that having the following information would improve predictions:
    - POI type labels (airport, school, shopping, residential, sports arena...)
    - Physical layout (map) of the districts and major roads
    - Special event information such as sports or other celebrations
    - Holiday calendar

Feature Engineering

- Limited amount of feature engineering was attempted due to my lack of experience. The non-linear dependencies with time_slot & week_day are not fully captured with the current model.

- Domain experts (in the taxicab industry) could provide insights regarding missing features and/or feature engineering.

Algorithm Tuning

- The neural network implementation was performed using the recently available tf.contrib.learn library from TensorFlow. This made the task of creating and tuning the networks much simpler. However, it lacked support for tuning parameters such as regularization.
- Learning rate decay and non-default options for the optimizer or activation function can be used.
- I attempted GPU based acceleration but did not notice any improvements in runtime performance. If GPU acceleration provides significant runtime improvement, a more exhaustive parameter search will become possible and could lead to improved tuning.
- Both the Random Forests and Gradient Boosting algorithms showed lower stability during sensitivity analysis caused by overfitting. Further parameter tuning may alleviate this problem.

Stacking Models

Competition winners at Kaggle typically stack models in two or more stages to achieve higher performing and stable results. Ensembles of dozens of base models are used and 'blended' with input features in multiple stages. I did not use this approach due to lack of experience.

References

1. Website: Didi Research Algorithm Competition 2016
2. Xu, R.: Machine Learning for real-time demand forecasting
3. Schaller, B.: Entry Controls in Taxi Regulation
4. Website: MIT Big Data Challenge: Transportation in the City of Boston