Exploring Different Regression Models to Forecast Hotel Room Price

By

Dong Chen
UalbanyID:001266361

A Project Report submitted to
University at Albany
in partial fulfillment of the requirements
for the degree of
MASTER OF SCIENCE
in
Computer Science and Engineering

Advisor: Prof. Feng Ch

05/15/20

AUTHORIZATION FOR REPRODUCTION OF PROJECT

I grant permission for the reproduction of this project in its entirety, without further authorization from me, on the condition that the person or agency requesting reproduction, absorb the cost and provide proper acknowledgment of authorship.

Date______05/15/2017_____ ______________________________

Dong Chen
1400 Washington Ave
Albany, NY, 122222

TABLE OF CONTENTS

Exploring Different Regression Models To Forecast Hotel Room Price

Dong Chen
Department of Computer Science, University at Albany - SUNY
dchen@albany.edu

Abstract

Regression is one of the most important and broadly used machine learning and statistics tools out there. It allows us to make predictions from data by learning the relationship between features of the data and some observed, continuous-valued response. Regression is used in a massive number of applications ranging from predicting stock prices to understanding gene regulatory networks. In this project, I applied different models to predict the prices of hotels and compared how good the models are by calculating the root-mean-square error (RMSE) of those models.

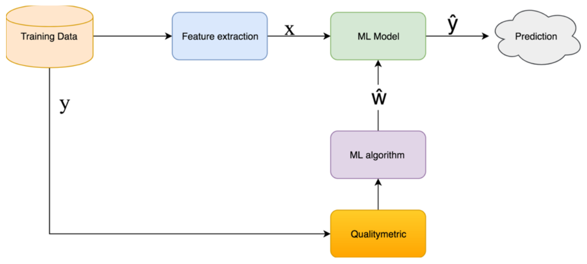
I. Introduction

Linear Regression is one of the most basic and important techniques for predicting a value of an attribute (y). It is used to fit values in a forecasting or predictive model. The attributes are usually fitted using the least squares approach. The cost function which it involves for minimising the error can be minimized using many mathematical tricks and algorithms (Gradient Descent, Derivative test, Newton's Method). Using the derivative method on the least squares approach and with the help of properties of matrices, it reduces the problem of linear regression to a consolidated equation.

There are many extensions to Linear regression as certain problems involve dependence of attributes not linearly but in more of a higher degree form. This sort of regression is Multivariate Linear Regression. Though the way it works is computationally more expensive than Linear Regression, it works very well in maintaining a balance between bias and variance.

The simplest linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is an explanatory variable, and the other is a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

Fig. 1. Machine Learning flowchart.



However, linear regression analysis consists of more than just fitting a linear line through a cloud of data points. It consists of 3 stages (1) analyzing the correlation and directionality of the data, (2) estimating the model, i.e., fitting the line, and (3) evaluating the validity and usefulness of the model. There are 3 major uses for regression analysis 1) causal analysis 2) forecasting an effect 3) trend forecasting. In this project, our goal is to focus on forcasting the accurate hotel price by comparing different regression models.

1. Simple linear regression one dependent variable one independent variable

2. Multiple linear regression one dependent variable and more than two independent.

3. The ridge regression(L2) uses L2 norm for regularization. The ridge regression gives an estimate which minimise the sum of square error as well as

satisfy the constraint that the L2 norm must be less than a certain value.

4. The Lasso regression(L1) uses L1 norm for regularization. The main difference between ridge and lasso regression is a shape of the constraint region. The Lasso estimate is an estimate **which** minimizes the sum of square as well as satisfy the below constraint.

5. K-Nearest Neighbor(KNN) regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

6. Multitask Learing(MLT) is an effective method when dealing with multiple related tasks. In MTL the related tasks are learnt simultaneously by utilizing appropriate shared information. By this way, MTL is beneficial when training data size is small and improves the performance of prediction.

II. Related Work

A. *Site and Situation facotrs affect room prices*

Insufficient attention has been given to hotel-room-price attributions and its mechanism in the lodging research field till now. A goup of researchershas developed a comparative analysis of four hedonic price models to investigate how these attributions influence room price of Beijing's hotels above star three. Spatial autocorrelation in hotel prices and in hedonic room price equation residuals were analyzed in this research too. According to the estimated results, for specific locales, the results expressed in a global model might be inaccurate.[1]

B. *Predicting hotel prices using quantile regression*

Due to the skewed distribution of hotel prices, quantile regression provides a more flexible and complete characterization of the determinants of the hotel prices at the higher and lower tail of the distribution. That study applies quantile regression approach to investigate the major determinants of hotel room pricing strategies. The ordinary least square regression is also used for comparative purposes. The data are drawn from 58 international tourist hotels in Taiwan and average room rate (ARR) is used as the proxy of hotel room price. The results of OLS and quantile regression share common characteristics but also have differences in some aspects. The OLS results reveal that number of rooms, hotel age, market conditions and number of housekeeping staff per room are the main attributes of hotel room rate.[2] The quantile regression results further demonstrate that room number and the number of housekeeping staff per guest room do not significantly influence hotel price at the low price quantile. Hotel age and market conditions are only significant determinants in high-price category. Additionally, for the high-priced quantile hotels, the proportion of foreign individual travellers positively and significantly influences room price.

C. *Hedonic priceing models*

The focus of this section is hedonic pricing models and their use estimating hotel room rates. This review concludes with an examination of more recent applications of hedonic pricing of consumer goods. Espinet et al. (2003) develop a hedonic pricing model for tourist resorts in the southern Costa Brava area. Data is collected from the tour operator Travelmar for the 1991 to 1998 period. Prices are based on monthly average of daily prices for a full-board arrangement per person per day. Among the key attributes, the authors find the star rating to have a significant positive impact on the price of an accommodation. They go on to state that there is a significant increase in revenue from 3-star to 4-star hotels, but not from 1-star to 2-star hotels. Espinet et al. (2003) also find a negative relationship between hotel capacity and hotel price.[3] However, they find positive relationships between the proximity to the sea as well as with the presence of a parking place to the price of a hotel room. Other characteristics, such as the presence of a garden, swimming pool and sports facility, are found to have insignificant effects on the price of hotel rooms in this area. Other notable results show that hotels located in certain towns tend to have higher premiums, upwards of 17%.

Andersson (2010) extends the literature on hedonic hotel pricing to analyze hotel room prices in Singapore using internet-based transactions. Using data collected from an internet-based hotel-booking agent, the author examines 563 hotel rooms from the period of January 2006 to March 2007. The author finds 4-star and 5-star hotels are associated with higher premiums. Other structural attributes associated with higher prices include in-room safes, premium rooms, and standard rooms.[4] Micro-neighborhood attributes associated with higher premiums include fitness centers, architectural interest, hotel facilities, and food and beverage options. Furthermore, hotels located on a popular road also see increases in price. In contrast, the authors find that the value for money (e.g., the attractiveness of price given hotel attributes) is associated with lower prices, as is distance to the city center.

III. Data

In this project, the hotel data was collected from Expedia. The dataset consists of hundreds of hotels in major cities like New York, Boston, Los Angeles and Chicago. The data collection strategy used here is that, everyday, for each hotel, we collected the price of all types of rooms for that a certain hotel for just that same day. To reduce the complexity of the problem, we just assume that the price would not affect by the duration of the stay for hotel price queries. For example, the table 1 showed below is part of the data we have collected from Expedia. We mainly crawled those features that can be measured numerically because it is convinient to use data with numerical to perform regression learning.

TABLE I. DATA SAMPLE (PART)

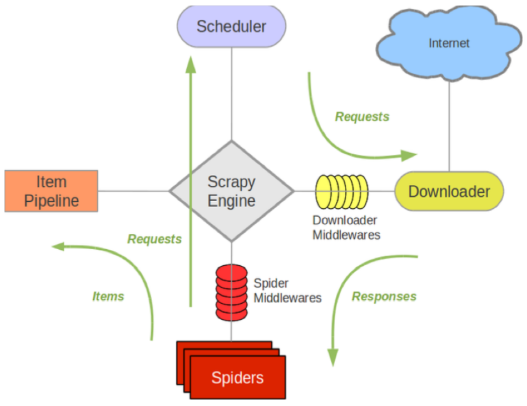
Hotel Names	**Features**		
	star	***rating***	***rates***
Sportsmen's Lodge	3.5	3.7	1175
The Bicycle Hotel & Casino	4	4.5	296
All Star Inn	2	2.8	33
Motel 6 Los Angeles LAX	2	3	1839

A. *How the data crawler works*

The following chart shows the architecture of the Scrapy, including an overview of component and occur in the system data flow (green arrow). The following for each component are introduced briefly, and gives the details of the link. Data flow are described below.

- Scrapy Engine: The engine controls the data flow between all components, and triggering events when certain action happens.
- Scheduler: The Scheduler receives requests from the engine and enqueues them for feeding them later when engine requests them.
- Downloader: The Downloader is responsible for fetching web pages and feeding them to the engine which, in turn, feeds them to spiders.
- Spiders: Spiders are custom classes written by user to parse responses and extract item from them or additional requests to follow.
- Item Pipeline: The Item Pipeline is responsible for processing the items once they have been extracted by the spiders.
- Downloader middlewares: Downloader middlewares sits between the Engine and the Downloader process requests when they pass from the Engine to the Downloader, and responses that pass from Downloader to the Engine.
- Spider middlewares sits between the Engine and the Spider and can process spider input(responses) and output (items and requests).

Fig. 2. The architecture of Scrapy.[5]



The data flow in Scrapy is controlled by the execution engine, and goes like this:

- The Engine gets the initial Requests from Spider to crawl data.

- The Engine schedules the Requests to in the Scheduler and asks for the next Request to crawl.
- The Scheduler send back the next Request to the Engine
- The Engine sends the Requests to the Downloader, passing through the Downloader Middlewares
- Once finished page downloading, the Downloader sends a response to the Engine, passing through the Downloader Middlewares
- The Engine receives the Reponses from the Downloader and sends it to the Spider for processing, passing through the Spider Middleware.
- The Spider processes the Response and returns scraped items and new Requests to the Engine, passing through the Spider Middleware.
- The Engine sends processed items to Item Pipelines, then send processed Requests to the Scheduler and asks for possible next Requests to crawl.
- The process repeats (from step 1) until there are no more request from Scheduler [5].

IV. MODELS

A. *Explore the data*

As described in the instruction, in this model, I simply just want to explore the data and see how the data would behave since there could be some weird features or mising features or even extremely abnormal data points that could affect the outcome tremendously. Therefore, the simple linear regression would do the work. I am using the least squre mesure to estimate the model. Mathematically, I employ the cost function of this simple model is as Eq. (1):

$$RRS(w_0, w_1) = \sum_{i=1}^{N}(y_i - [w_0 + w_1 x_i])^2 \quad (1)$$

w_0 and w_1 are the intercept and slope for line we are going to use to predict the hotel room rate. y_i and x_i are the data values that we have collected, in here are price of the hotel room and size of the hotel room respectively. And we can vitualize the first-hand data and the regression fit as flow:

Fig. 3. The overview of the data

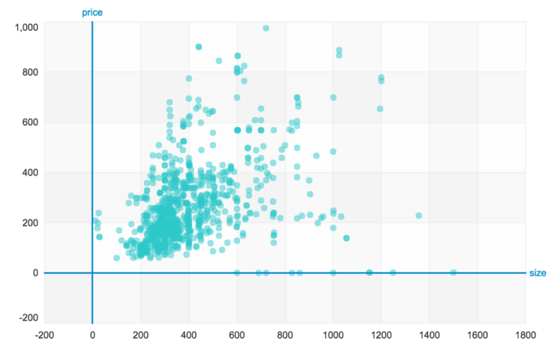
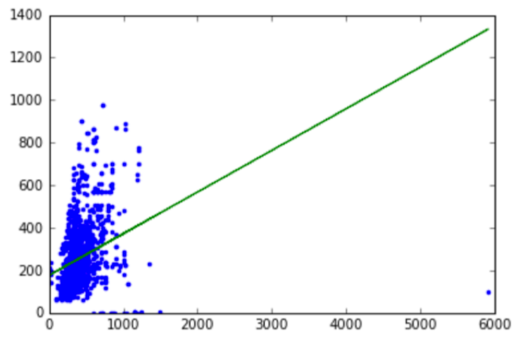


Fig. 4. The regression fit using size as the only feature



We can see that there is one data point that has the largest room size, yet the hotel price is extremely low. This point does not follow the trend of the rest of the data very well. A question is how much including this abnormal point is influencing our fit on the other data points. Let's remove this data point and see what happens.

Visually, the fit looks different, but let's quantify this by examining the estimated coefficients of our original fit and that of the modified dataset with the abnormal point removed.

Fig. 5. The regression fit after removing the abnormal point

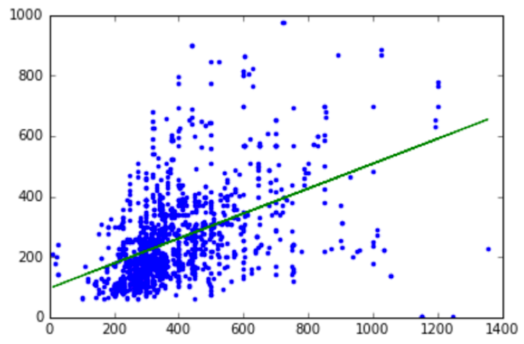


TABLE II. COEFFICIENTS FOR BOTH MODELS

coefficients	Feature values		
	value	***stderr***	***index***
Intercept_1	176.520639819	7.57629569478	None
Size_1	0.195453748709	0.016495242564	None
Intercept_2	96.1808897241	8.64899432279	None
Size_2	0.411889907028	0.0203156316421	None

We see that for the version without abnormal data points, per unit increase in size, the predicted increase in hotel prices is 0.412. In contrast, for the original dataset, the growth is only 0.195 per unit increase in hotel price. So those tree points could affect the outcome hugely. As a result, I would always eliminate those high leverage points influential points before we could do any learning for the future models.

The hotel with 0 review score is said to be a "high leverage" point because it is at an extreme x value say 0 where there are no other observations. As a result, recalling the closed-form solution for simple regression, this point has the potential to dramatically change the least squares line since the center of x mass is heavily influenced by this one point and the least squares line will try to fit close to that outlying (in x) point. If a high leverage point follows the trend of the other data, this might not have much effect. On the other hand, if this point somehow differs, it can be strongly influential in the resulting fit.

An influential observation is one where the removal of the point significantly changes the fit. As discussed above, high leverage points are good candidates for being influential observations, but need not be. Other observations that are not leverage points can also be influential observations (e.g., strongly outlying in y even if x is a typical value). [6]

B. *Simple Linear Regression*

In statistics, simple linear regression is a linear regression model with a single explanatory variable. That is, it concerns two-dimensional sample points with one independent variable and one dependent variable (conventionally, the x and y coordinates in a Cartesian coordinate system) and finds a linear function (a non-vertical straight line) that, as accurately as possible, predicts the dependent variable values as a function of the independent variables. The adjective simple refers to the fact that the outcome variable is related to a single predictor. [7] To create test data for the model, first we should split the data into training and testing set where we will use 80% of the data as the training data set and 20% of them as the test set. We can use the generic function random split to do this work by the generic function random split. Armed with these SArray functions we can use the closed form solution to compute the slope and intercept for a simple linear regression on observations stored as SArrays: input feature, output.

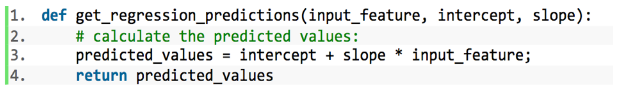
Fun. 1. Closed-Form-Solution

```
def simple_linear_regression(input_feature, output):
    input_sum = input_feature.sum()
    output_sum = output.sum()
    N = input_feature.size()
    input_mean = input_sum/N
    output_mean = output_sum/N
    in_out_prod = input_feature * output
    in_out_prod_sum = in_out_prod.sum()
    prod_sum = output_sum * input_sum
    prod_mean = prod_sum/N
    sqr_test = input_feature * input_feature
    sqr_test_sum = sqr_test.sum()
    sqr_sum = input_sum * input_sum
    sqr_mean = sqr_sum/N
    slope = (in_out_prod_sum - prod_mean)/(sqr_test_sum - sqr_mean)
    intercept = output_mean - (input_mean * slope)
    return(intercept, slope)
```

After passing all the data input feature and output value in the function we can get the Intercept w_0 = 185.261339188 and slope w_1 = 0.182229076588. Now that we have the model parameters: intercept & slope we can make predictions. Using the function to calculate the prediction. We will return the predicted output given the input feature, slope and intercept:

$$\hat{y} = w_0 + w_1 x \tag{2}$$

Fun. 2. Predict the price



```python
def get_regression_predictions(input_feature, intercept, slope):
    # calculate the predicted values:
    predicted_values = intercept + slope * input_feature;
    return predicted_values
```

We can calculate a prediction given the slope and intercept. We want to find out the estimated price for a hotel with size of 400 square feet according to the model we estimated above. Therefore, after the calculation, the estimated price for a house with 400 square feet is $258.15. Now we have the tool to make a model for predicting hotel prices using room size, but there are many other features in the data. We can also use hotel star to form a model and test which model would be better in predicting the hotel room price. Just plug-in the same function we can get the Intercept for star is: -137.582534468, the slope is: 118.030847711.

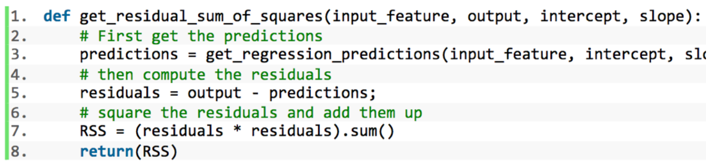
Now that we have two models that can make predictions let's evaluate our model using Residual Sum of Squares (RSS) and Root Mean Square Error (RMSE). We will create another function to compute the RSS and RMSE of a simple linear regression model given the input feature, output, intercept and slope:

$$RRS(w_0, w_1) = \sum_{i=1}^{N} (y_i - [w_0 + w_1 x_i])^2 \tag{3}$$

$$RMSE(w_0, w_1) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - [w_0 + w_1 x_i])^2} \tag{4}$$

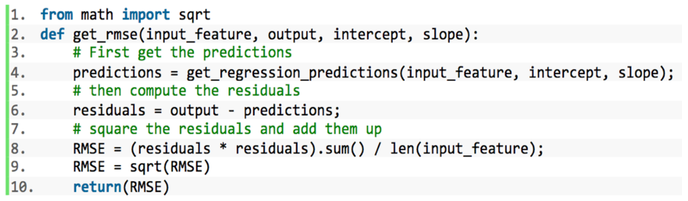
After the calculation, the RSME when using [star] on TEST data is: 112.490218939, while RSME when using [size] on TEST data is: 142.856727874. As we can see, the model using star is much better than the model using size of the room.

Fun. 3. Caculate RSS



```python
def get_residual_sum_of_squares(input_feature, output, intercept, slope):
    # First get the predictions
    predictions = get_regression_predictions(input_feature, intercept, sl
    # then compute the residuals
    residuals = output - predictions;
    # square the residuals and add them up
    RSS = (residuals * residuals).sum()
    return(RSS)
```

Fun. 4. Caculate RSME



```python
from math import sqrt
def get_rmse(input_feature, output, intercept, slope):
    # First get the predictions
    predictions = get_regression_predictions(input_feature, intercept, slope);
    # then compute the residuals
    residuals = output - predictions;
    # square the residuals and add them up
    RMSE = (residuals * residuals).sum() / len(input_feature);
    RMSE = sqrt(RMSE)
    return(RMSE)
```

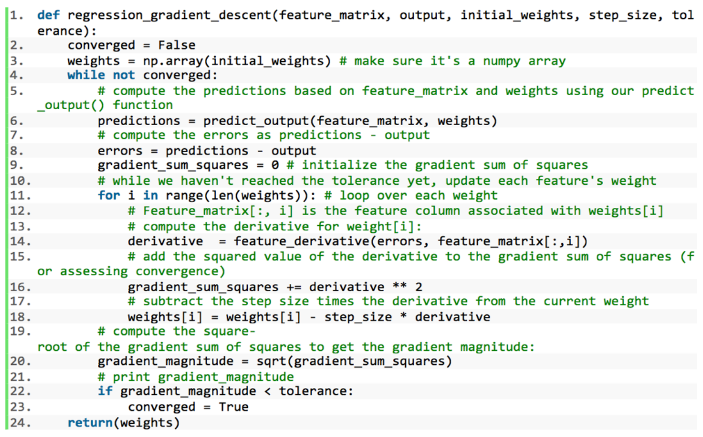
Based on common sense, the star metric is often a better way to measure the quality(price) of a certain hotel room. This tells us that the feature we choose to train the data would be crucial to predict the price. We would do more feature selection in the Lasso Regression in the later chapter.

C. *Multiple Linear Regression*

Multiple regression is an extension of simple linear regression as we have listed in the Introduction part. It is used when we want to predict the value of a variable based on the value of two or more other features. The variable we want to predict is called the target where we defined the target as the price of the hotel room. The variables we are using to predict the value of the dependent variable are called the independent variables. In this mode, we are going to use more than single feature as we did in the last model. Here we intuitively chose the 'star', 'size', and 'ratings' as the feature to learn the multiple linear regression model. As usual we are going to split the data into 80% and 20% for the training data and test data respectively. This time, instead of using the Closed-Form-Solution to compute the weight of those features, we will use gradient decent algorithm to do so. What we need is a function that performs a gradient descent. The basic premise is simple. Given a starting point we update the current weights by moving in the negative gradient direction. The gradient is the direction of increase and therefore the negative gradient is the direction of decrease and we're trying to minimize a cost function. The amount by which we move in the negative gradient direction is called the 'step size'. We stop when we are 'sufficiently close' to the optimum. We define this by requiring that the magnitude (length) of the gradient vector to be smaller than a fixed 'tolerance'. The step size will be much smaller than we might expect but this is because the gradient has such large values. Below is the pseudocode and the real code for the gradient decent algorithm.

```
init $\boldsymbol{w}^{(\mathbf{1})} = 0$ (or randomly, or smartly), t = 1
while || $\nabla$ RSS $\boldsymbol{w}^{(\boldsymbol{t})}$ || > ε
    for j = 0, …, D
    partial[j] = $-2 \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\boldsymbol{w}^{(\boldsymbol{t})}))$
    ${w_j}^{(t+1)} \leftarrow {w_j}^{(t)} - \eta$ partial[j]
    $t \leftarrow t+1$
```

Fun. 5. Gradient decent for multiple regression



```
def regression_gradient_descent(feature_matrix, output, initial_weights, step_size, tolerance):
    converged = False
    weights = np.array(initial_weights) # make sure it's a numpy array
    while not converged:
        # compute the predictions based on feature_matrix and weights using our predict_output() function
        predictions = predict_output(feature_matrix, weights)
        # compute the errors as predictions - output
        errors = predictions - output
        gradient_sum_squares = 0 # initialize the gradient sum of squares
        # while we haven't reached the tolerance yet, update each feature's weight
        for i in range(len(weights)): # loop over each weight
            # Feature_matrix[:, i] is the feature column associated with weights[i]
            # compute the derivative for weight[i]:
            derivative  = feature_derivative(errors, feature_matrix[:,i])
            # add the squared value of the derivative to the gradient sum of squares (for assessing convergence)
            gradient_sum_squares += derivative ** 2
            # subtract the step size times the derivative from the current weight
            weights[i] = weights[i] - step_size * derivative
        # compute the square-root of the gradient sum of squares to get the gradient magnitude:
        gradient_magnitude = sqrt(gradient_sum_squares)
        # print gradient_magnitude
        if gradient_magnitude < tolerance:
            converged = True
    return(weights)
```

Output: y (scalar)
Inputs: $\mathbf{x} = (\mathbf{x}[1],\mathbf{x}[2],\ldots,\mathbf{x}[d])$ (d-dim vector)
$\mathbf{x}[j]$ = jth input (scalar)
$h_j(\mathbf{x})$ = jth feature (scalar)
$\mathbf{x}_i$ = input of ith data point (vector)
$\mathbf{x}_i[j]$ = jth input of ith data point (scalar)

To find the best feature, we consider the transformations of existing features e.g. the log of the size or even "interaction" features such as the product of rating and rates. 'star_squared = star * star', 'star_rating = star * rating', 'log_size = log(size)'.

The reason why we set the new features is that squaring star will increase the separation between not fancy hotels (e.g. 2 star) and fancy hotel (e.g. 5) since 2^2 = 4 but 5^2 = 25. Consequently, this feature will mostly affect big-star luxury hotels. The feature star times rating gives what's called an "interaction" feature. It is large when both are large. Taking the log of size of the hotel room has the effect of bringing large values closer together and spreading out small values. The difference between rates and rating is totally non-scenical but we will do it just for testing and comparison.

TABLE III. MODEL1 COEFFICIENTS

coefficients	**Feature values**		
	value	***stderr***	***index***
intercept	-129.244772102	23.7829617109	None
size	0.0808452500304	0.0142100330904	None
star	116.550058842	5.31826612178	None
rating	-9.03680177658	8.14256748473	None

TABLE IV. MODEL2 COEFFICIENTS

coefficients	**Feature values**		
	value	***stderr***	***index***
intercept	44.7513118174	79.7804756611	None
size	0.079088697748	0.014204322029	None
star	49.0818241099	29.9971649747	None
rating	-51.2732109322	20.1963107268	None
star_rating	16.2172232403	7.0962843183	None

TABLE V. MODEL3 COEFFICIENTS

coefficients	**Feature values**		
	value	***stderr***	***index***
intercept	-389.245307082	103.794553221	None
size	0.0032073110041	0.0217214751786	None
star	-1.54875235654	30.3383464326	None
rating	77.3984967902	31.5805324912	None
star_rating	-29.493545194	11.0228408461	None
star_squared	34.4068605547	6.20052335183	None
log_size	56.5395795933	12.5090974673	None

- Model1 Feature: [size, star, rating]
- Model2 Feature: [size, star, rating, star_rating]
- Model3 Feature: [size, star, rating, star_rating, star_squared, log_size]

After the learning process, we need to analyze models by using the RSS and RMSE. Recall that Function (3) and Function (4) employed for the simple regression and we could not apply those functions to the multiple regression, therefore we should use the matrix forms to calculate the RSS and RMSE.

$$RSS(\boldsymbol{w}) = \sum_{i=1}^{N}(y_i - h(\mathbf{x}_i)^T\boldsymbol{w})^2 \quad (5)$$

$$RMSE(\boldsymbol{w}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - h(\mathbf{x}_i)^T\boldsymbol{w})^2} \quad (6)$$

This time, we will repeat the process of calculating the RSS and RMSE by plugging in the Function (3) and Function (4) although we must modify those two functions into the matrix notation. With the help of the Frame that we use to manipulate the dataset, we can easily revise the functions and apply them to the multiple regression task.

TABLE VI. RSS & RMSE FOR THREE MODELS

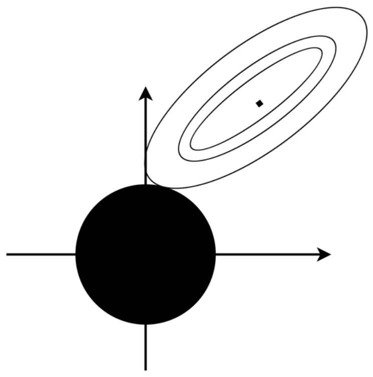
Metrics	**Models**		
	Model_1	***Model_2***	***Model_3***
RSS	14308109.193	14242865.809	13666716.3833
RMSE	110.965521591	110.712237316	108.449867791

The numbers tell us that the feature manipulation would magnify the importance of the critical features while alleviating the effectiveness of the less-important features. And we can also conclude the fact that the three models using multiple regression are all better than the simple regression model because of the RMSE we have collected.

D. *Ridge Regression*

In statistics, the method is known as ridge regression, in machine learning it is known as weight decay, and with multiple independent discoveries, it is also variously known as the Tikhonov–Miller method, the Phillips–Twomey method, the constrained linear inversion method, and the method of linear regularization. It is related to the Levenberg–Marquardt algorithm for non-linear least-squares problems. we will run ridge regression multiple times with different L_2 penalties to see which one produces the best fit. In many cases, this matrix is chosen as a multiple of the identity matrix, giving preference to solutions with smaller norms; this is known as L_2 regularization. [8] In other cases, low pass operators (e.g., a difference operator or a weighted Fourier operator) may be used to enforce smoothness if the underlying vector is believed to be mostly continuous. This regularization improves the conditioning of the problem, thus enabling a direct numerical solution.

Fig. 6. Ridge Regression Overview



In this model, we will run ridge regression multiple times with different L_2 penalties to see which one produces the best fit. To achieve the best fit, first we will use the cross-validation to see the effect of L2 regularization. Also, we should assess the final model using the test data and compute the RMSE for the model. We will not use the gradient descent algorithm this time because it would be too hard to choose the right L_2 value.

$$PRSS(\boldsymbol{w})_{l_2} = RSS(\boldsymbol{w}) + \lambda||\boldsymbol{w}||_2^2 \quad (7)$$

$$PRSS(\boldsymbol{w})_{l_2} = (\boldsymbol{y} - \boldsymbol{Hw})^{\boldsymbol{T}}(\boldsymbol{y} - \boldsymbol{Hw}) + \lambda\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} \quad (8)$$

The L_2 penalty is a "magic" parameter we need to select. We will implement a kind of cross-validation called k-fold cross-validation. The method gets its name because it involves dividing the training set into k segments of roughly equal size. Like the validation set method, we measure the validation error with one of the segments designated as the validation set. The major difference is that we repeat the process k times as follows:

- Set aside segment 0 as the validation set, and fit a model on rest of data, and evaluate it on this validation set

- Set aside segment 1 as the validation set, and fit a model on rest of data, and evaluate it on this validation set...
- Set aside segment k-1 as the validation set, and fit a model on rest of data, and evaluate it on this validation set

After this process, we compute the average of the k validation errors, and use it as an estimate of the generalization error. Notice that all observations are used for both training and validation, as we iterate over segments of data. To estimate the generalization error well, it is crucial to shuffle the training data before dividing them into segments. The tool we are using is GraphLab. GraphLab Create has a utility function for shuffling a given SFrame. We reserve 10% of the data as the test set and shuffle the remainder. Compute average error:

$$\mathrm{CV}(\lambda) = \frac{1}{K}\sum_{1}^{K} error_k(\lambda) \qquad (9)$$

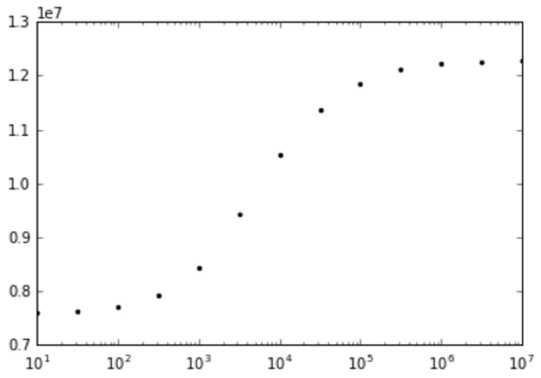
Fun. 6. K-fold cross validation algorithm

```
def k_fold_cross_validation(k, l2_penalty, data, output_name, features_list):
    vali_error_sum = 0
    n = len(data)
    for i in xrange(k):
        start = (n * i) / k
        end = (n * (i + 1))/ k - 1
        validation_set = data[start : end + 1]
        train_set = data[0 : start].append(data[end + 1 : n])
        model = graphlab.linear_regression.create(train_set, target = output_name, featu
res = features_list, l2_penalty=l2_penalty,
                                                       verbose=False,validation_set=None)
        predicitons = model.predict(validation_set)
        residuals = predicitons - validation_set[output_name]
        rss =  sum(residuals**2)
        vali_error_sum += rss

    ave =  vali_error_sum / k
    return ave
```

Once we have a function to compute the average validation error for a model, we can write a loop to find the model that minimizes the average validation error. We will again be aiming to fit a 15th-order polynomial model using the size input for l2_penalty in [10^1, 10^1.5, 10^2, 10^2.5, ..., 10^7] (Numpy function: np.logspace(1, 7, num=13).) Since the degree of the polynomial is now fixed to 15, to make things faster, we should generate polynomial features in advance and re-use them throughout the loop. We will plot the l2_penalty values in the x axis and the cross-validation error in the y axis. This time we will Use plt.xscale('log') will make the plot more intuitive.

Fig. 7. L_2 Selection



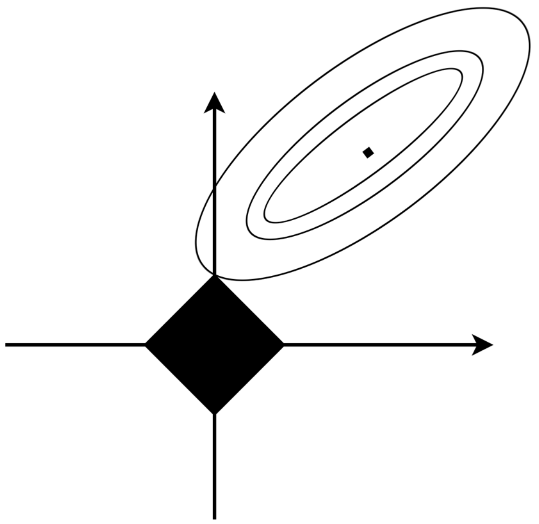
Now that we have selected the best L_2 penalty, and all we need to do is to plug in the multiply regression model and relearn the training data from last model and compute the RSS and the RMSE which is 18423054.6709 and 96.194108304 respectively. This figure shows that the regulated learning(L_2) performs better than the model without it in multiple regression learning.

E. *Lasso Regression*

In statistics and machine learning, lasso (least absolute shrinkage and selection operator) (also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the statistical model it produces. It was introduced by Robert Tibshirani in 1996 based on Leo Breiman's Nonnegative Garrote. [8] Lasso was originally formulated for least squares models and this simple case reveals a substantial amount about the behavior of the estimator, including its relationship to ridge regression and best subset selection and the connections between lasso coefficient estimates and so-called soft thresholding. It also reveals that (like standard linear regression) the coefficient estimates need not be unique if covariates are collinear.

Though originally defined for least squares, lasso regularization is easily extended to a wide variety of statistical models including generalized linear models, generalized estimating equations, proportional hazards models, and M-estimators, in a straightforward fashion. [1][3] Lasso's ability to perform subset selection relies on the form of the constraint and has a variety of interpretations including in terms of geometry, Bayesian statistics, and convex analysis.

Fig. 8. Lasso Overview



In this model, we mainly use Lasso regression to select the most influential feature for the task. We can come up with a quite tricky situation where there are 100 billion feature that we can use to train the data, but each prediction is expensive. To make it computationally feasible, we can manipulate the input feature and make it sparse. If the input feature is sparse, the computation only depends on the number of the non-zero features. So, which features are the most relevant to the task for the prediction? Instead of searching over a discrete set of solutions, can we use regularization? We can start with full model (all possible features) and then "Shrink" some coefficients exactly to 0. Then, knock out certain features, as a result, the non-zero coefficients would stand out to indicate "selected" features.

L_1 regularized regression Just like ridge regression, solution is governed by a continuous parameter λ. For convex problems, will start to take smaller and smaller steps Measure size of steps taken in a full loop over all features - stop when max step < ε. Blow is the function for penalized RSS (PRSS):

$$PRSS(\boldsymbol{w})_{l_2} = RSS(\boldsymbol{w}) + \lambda||\boldsymbol{w}||_{\mathbf{1}} \quad (10)$$

Coordinate descent algorithm for lasso with normalized features, blow is the pseudo for the coordinate descent:

```
Initialize $\hat{w} = 0$ (or smartly)
  while not converged
  for j = 0, 1, …, D

    compute:  $\rho_j = \sum_1^N h_j(\boldsymbol{x_i})(y_i - \hat{y}_i(\boldsymbol{\hat{w}_{-j}}))$

    set: $\boldsymbol{\hat{w}_j} = \begin{cases} \boldsymbol{\rho_j} + \frac{\lambda}{2} & \boldsymbol{if\ \rho_j} < -\frac{\lambda}{2} \\ \mathbf{0} & \boldsymbol{if\ \rho_j\ in}\,[-\frac{\lambda}{2}, \frac{\lambda}{2}] \\ \boldsymbol{\rho_j} - \frac{\lambda}{2} & \boldsymbol{if\ \rho_j} > \frac{\lambda}{2} \end{cases}$
```

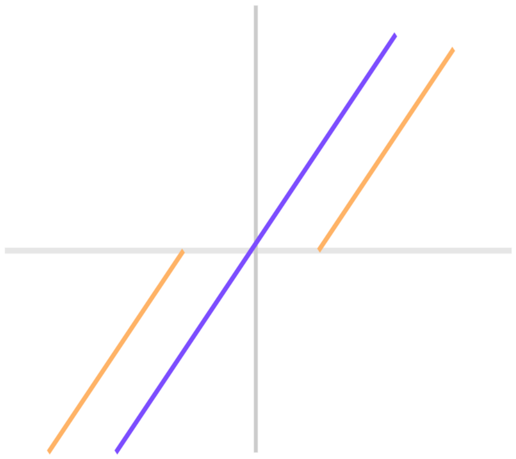
We seek to obtain a sparse set of weights by minimizing the LASSO cost function, Function (10). (By convention, we do not include w [0] in the L1 penalty term. We never want to push the intercept to zero.)

The absolute value sign makes the cost function non-differentiable, so simple gradient descent is not viable (we would need to implement a method called sub gradient descent). Instead, we will use coordinate descent: at each iteration, we will fix all weights but weight i and find the value of weight i that minimizes the objective. Here are the steps:

1. Pick a coordinate i
2. Compute w[i] that minimizes the cost function SUM [(prediction - output) ^2] + lambda*(|w [1] | + ... + |w[k]|)
3. 3.Repeat Steps 1 and 2 for all coordinates, multiple times

We will use cyclical coordinate descent with normalized features, where we cycle through coordinates 0 to (d-1) in order, and assume the features were normalized as discussed above. When do we know to stop? Each time we scan all the coordinates (features) once, we measure the change in weight for each coordinate. If no coordinate changes by more than a specified threshold (ε), we stop. In this process, we also can manually set how many feature we would like to keep, say the non-zero features. In this model, we set the max_nonzeros = 6; Some techniques can be applied during this process for example when we are choosing the L_1 on the validation set, we can further narrow down the bestRSS score and bestl1 by performing the cyclical coordinate:

Fig. 9. Soft thresholding



L1-----------------RSS---------------max_nonzeros

```
{1438449.8882876630: (261230622.88899678, 7),
 1485674.7083298976: (263804976.22194180, 7),
 1532899.5283721322: (265755287.60718963, 6),
 1580124.3484143671: (267305174.79435593, 6),
 1627349.1684566017: (268868705.99515540, 6),
 1674573.9884988363: (271290798.39221954, 6),
 1721798.8085410709: (274142245.43534270, 6),
 1769023.6285833055: (277033085.90482290, 6),
 1816248.4486255401: (279985624.84062790, 6),
 1863473.2686677747: (284277060.55391914, 6),
 1910698.0887100096: (288316477.72309760, 5),
 1957922.9087522442: (291243627.50947690, 5),
 2005147.7287944788: (294205169.79052890, 5),
 2052372.5488367134: (297201088.88838510, 5),
 2099597.3688789480: (300278070.08631410, 5),
 2146822.1889211829: (303465152.03691715, 5),
 2194047.0089634173: (306699043.32456770, 5),
 2241271.8290056521: (310091561.12653154, 5),
 2288496.6490478870: (314052305.56794480, 5),
 2335721.4690901213: (319262720.23849720, 5)}
```

We can eyeball that the best RSS for the best L_1 value is 265755287.607, 1532899.52837 respectively. With these values we have computed, we can learn another model using the multiple regression using all the numeric feature we can get with L1 regulation to select the right features we need to use for the next round learning.

TABLE VII. LASSO FEATURE SELECTION

Coefficient Names	Features		
	index	*value*	*stderr*
intercept	None	153.531245227	None
star	None	6.443666566	None
star_squared	None	2.93345739643	None
size_sqrt	None	1.29300746878	None
rates_rating	None	0.0	None
zipcode	None	0.0	None
rating	None	1.86944571357	None
rates	None	0.0	None
size	None	0.120075102605	None
guests	None	0.0	None

As we perform another multiple regression learning only using the feature we have selected for based on the data above, we can get a RSS and RSME of 36056230.3335 and 118.102716225 respectively.

F. K-Nearest Neighbors (k-NN) regression

The k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

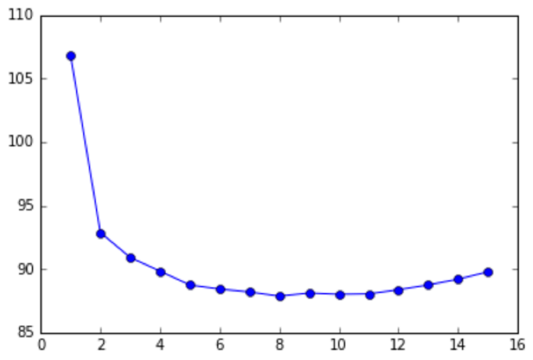
- In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. [10]

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. There remains a question of choosing the value of k to use in making predictions. Here, we use a validation set to choose this value.

We will need a loop to finish this task. For k in [1, 2, ..., 15]: Makes predictions for each hotel in the VALIDATION set using the k-nearest neighbors from the TRAINING set. Then we could compute the RSS and RMSE for these predictions on the VALIDATION. To visualize the performance as a function of k, we will plot the RMSE and on the VALIDATION set for each considered k value:

Fig. 10. Chosing the k value

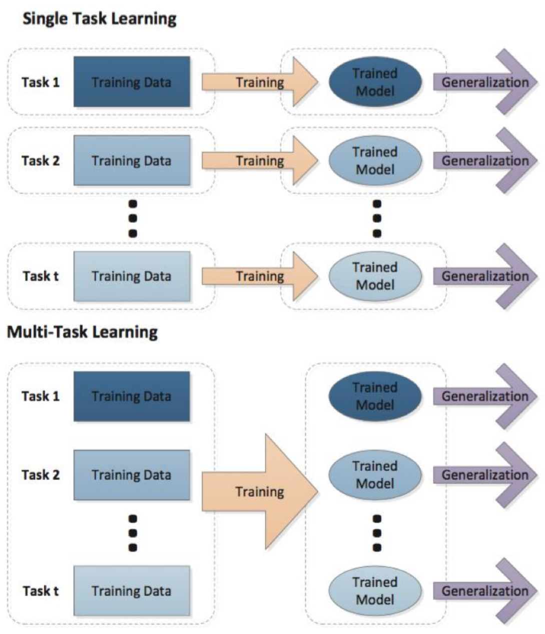


As we can see from the chart that, when k = 8, we can achieve the smallest RMSE ann at the same time get the smallest RSS. The final results are RSS = 39329030.7143 and RMSE = 88.005592927. By far, this k-NN performs the best among all the models we have discovered and I believe if we can conclude that if have enough data say 99% of all the hotel room price out there in the market, this model would be performing the best to predict values given the 'similarity' of the two hotel rooms.

G. Multitask-Learnig (MTL)

The traditional way of dealing with classification/regression/clustering tasks is single task learning (STL). In STL, the relatedness is ignored, each task is independently and learnt independently. In MTL, these tasks are learnt simultaneously by extracting and utilizing appropriate shared information across tasks. Learning multiple related tasks simultaneously effectively increases the sample size for each task, and improves the prediction performance. Thus, multi-task learning is especially beneficial when training size is small for each task [11]. Figure 2 shows the difference between STL and MTL.

Fig. 11. Difference between STL and MTL[14]



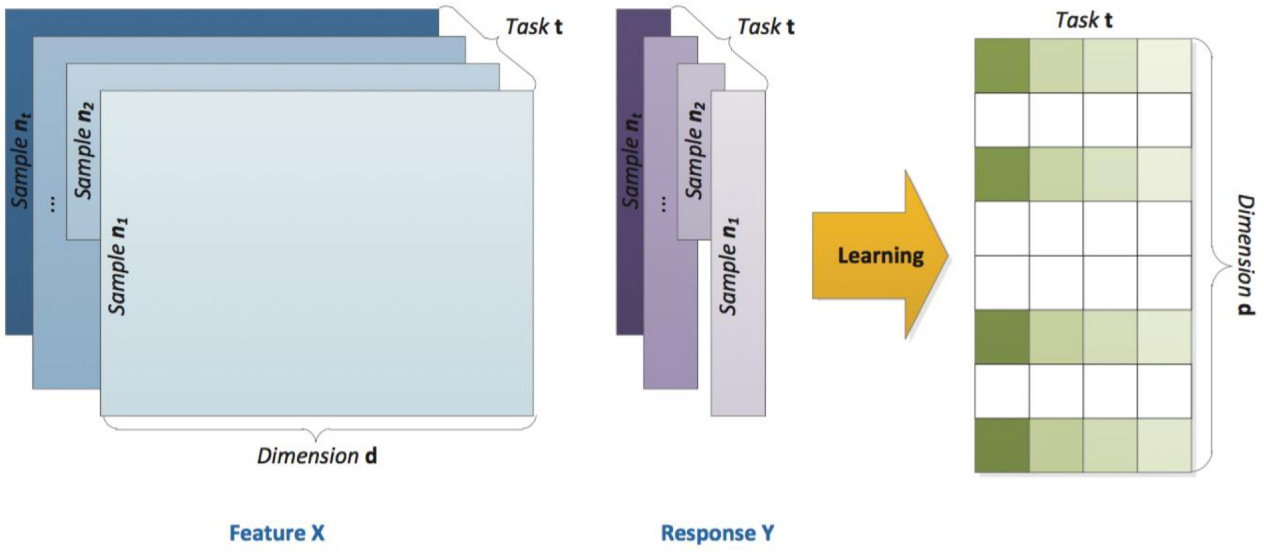
In this model, we are going to use Joint Feature Selection: $\ell 2,1$ - norm Regularization with Least Square Loss(Least_L21) One way to capture the task relatedness from multiple related tasks is to constrain all models to share a common set of features. This motivates the group sparsity:[12]

$$\min_{\boldsymbol{w}} \; RSS(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_{1,2} \tag{11}$$

The Least_L21 model solves the $\boldsymbol{\ell_{2,1}}$ -norm regularized multi-task least squares problem: [13]

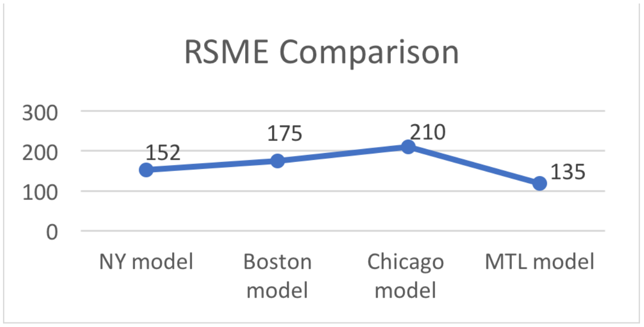
$$\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \boldsymbol{w})^2 + \rho_1 \|\boldsymbol{w}\|_{2,1} + \rho_{L2} \|\boldsymbol{w}\|^2 \tag{12}$$

Fig. 12. MTL joint feature selection on the $\boldsymbol{\ell 2,1}$ norm Regularization[15]



Not quite like what we did in the previous models, in this model, first we need to combine the data inputs features and all the related values together as a big task showed in the Fig. 12. Only with this method can we learn the data thoroughly and share those features across all the data and therefore we call it the joint feature selection $\boldsymbol{\ell 2,1}$ norm Regularization. As for the comparison, we are also going to learn each of the single task for this case. There will be four models in total the first one would be $\boldsymbol{\ell 2,1}$ norm Regularization learning only using the hotel data from NYC (NY model), second model (Boston model) using data from Boston, third using data from Chicago (Chicago model), the last model we combine all the features and response together to learn then all together (MTL model).

Fig. 13. RSME Comparison between the four values



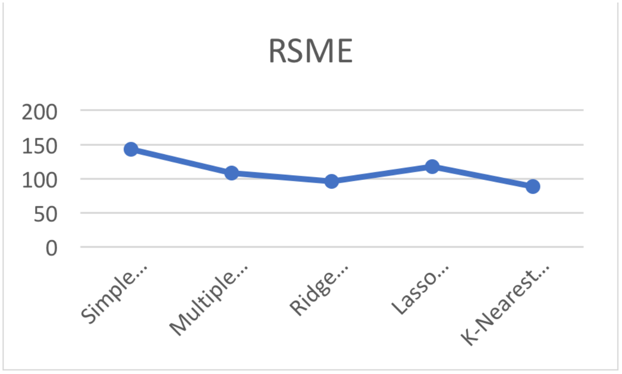
V. Conclusion & Discussion

In this project, I have compared some of the most popular regression models and have calculated their RMSE as a metric to measure the quality of those models.

First, surprisingly, the model with the least RMSE is the k-NN model we have built. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. The RMSE is way better than any of those parametric regression learning models. However, this model is just a non-parametric method, if we do not have enough data (Luckily there is enough data for that model) to support the process of the learning, I believe the RMSE would not be that good.

Second, the performance multitasking learning improves a lot down the stretch comparing to the single task learning using the $\ell 2,1$ norm Regulation. From this angle, we can conclude that when learning a model, we should always consider sharing features of those tasks together to find a deep connection between data as to increase the prediction performance.

Fig. 14. RSME Comparison between models



Third, it is well known, as explained by others, that L1 regularization helps perform feature selection in sparse feature spaces, and that is a good practical reason to use L1 in some situations. However, beyond that reason I have never seen L1 to perform better than L2 in practice. And, to be clear, I don't think I am the only one to be in this situation. Even in a situation where you might benefit from L1's sparsity to do feature selection, using L2 on the remaining variables is likely to give better results than L1 by itself. In our project, it seems L2 norm regulation solely performs better than only learning with L1 norm regulation.

Fourth, there are so many other models we can explore using the Multitask learning which would be a trend when analyzing data that has underlining connections. Within the MTL paradigm, information can be shared across some or all the tasks. Depending on the structure of task relatedness, one may want to share information selectively across the tasks. For example, tasks may be grouped or exist in a hierarchy, or be related according to some general metric. Suppose that the parameter vector modeling each task is a linear combination of some underlying basis. Similarity in terms of this basis can indicate the relatedness of the tasks. For example, with sparsity, overlap of nonzero coefficients across tasks indicates commonality. A task grouping then corresponds to those tasks lying in a subspace generated by some subset of basic elements, where tasks in different groups may be disjoint or overlap arbitrarily in terms of their bases. Task relatedness can be imposed a priory learned from the data.

VI. REFERENCE:

[1] Modeling hotel room price with geographically weighted regression(Article in International Journal of Hospitality Management 30(4):1036-1043 · December 2011)

[2] Pricing determinants in the hotel industry: Quantile regression analysis (Article in International Journal of Hospitality Management 29(3):378-384 · September 2010)

[3] Espinet, J.M., M. Saez, G. Conders, and M. Fluviá (2003) 'Effect on prices of the attributes of holiday hotels: a hedonic prices approach,' Tourism Economics 9(2), 165-177

[4] Andersson, D.E. (2010) 'Hotel attributes and hedonic prices: an analysis of internet-based transactions in Singapore's market for hotel rooms,' Annals of Regional Science 44(2), 229-240

[5] Architecture Overview of Scrapy: https://doc.scrapy.org/en/latest/topics/architecture.html

[6] Chatterjee, S., & Hadi, A. S. (1986). [Influential Observations, High Leverage Points, and Outliers in Linear Regression]: Rejoinder. Statistical Science, 1(3), 415-416.

[7] "What is Simple Linear Regression?". Pennsylvania State University.

[8] Ng, Andrew Y. (2004). Feature selection, L_1 vs. L_2 regularization, and rotational invariance. Proc. ICML.

[9] Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the lasso". Journal of the Royal Statistical Society. Series B (methodological) 58 (1). Wiley: 267–88.

[10] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[11] [Zhou 2012] J. Zhou, J. Chen and J. Ye. MALSAR: Multi-tAsk Learning via StructurAl Regularization. Arizona State University, 2012. http://www.public.asu.edu/~jye02/Software/MALSAR.

[12] [Zhou 2012] J. Zhou, J. Chen and J. Ye. MALSAR: Multi-tAsk Learning via StructurAl Regularization. Arizona State University, 2012. http://www.public.asu.edu/~jye02/Software/MALSAR.

[13] [Zhou 2012] J. Zhou, J. Chen and J. Ye. MALSAR: Multi-tAsk Learning via StructurAl Regularization. Arizona State University, 2012. http://www.public.asu.edu/~jye02/Software/MALSAR.

[14] [Zhou 2012] J. Zhou, J. Chen and J. Ye. MALSAR: Multi-tAsk Learning via StructurAl Regularization. Arizona State University, 2012. http://www.public.asu.edu/~jye02/Software/MALSAR.

[15] [Zhou 2012] J. Zhou, J. Chen and J. Ye. MALSAR: Multi-tAsk Learning via StructurAl Regularization. Arizona State University, 2012. http://www.public.asu.edu/~jye02/Software/MALSAR.

[16] Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12: 55-67

[17] Seber, G. and Lee, A. (2003). Linear Regression Analysis, 2nd Edition.