Background

The purpose of this project is to predict the quantitative target variable, price, of an Uber or Lyft ride based on Boston Uber/Lyft ride data and weather data taken from a real-time API, ranging from November 26 to December 18, 2018. To achieve this, supervised regression methods will be employed on a clean dataset containing the weather data appended to the ride data, along with a dummy variable column representing the occurrence of a sports game/event that occurred within one hour of the game's start time and end time. This section details the methodology involved in the statistical analysis process, performing tests to determine unique relationships between features and feature groups.

Dataset Description

Real time ride data, collected using Uber and Lyft API queries, containing 10 features including: ride application used (Uber, Lyft), distance between pickup and dropoff, timestamp of ride, pickup location of ride, destination of ride, price of ride, surge multiplier of ride (over much price was increased), specific type of ride (Uber Black, Lyft Lux XL), and corresponding id is included. There are over 690000 ride data instances recorded for this dataset.

Weather data, containing shared features with the ride dataset such as the location of the ride pickup, at timestamps ranging between November - December, contains unique features such as rain, clouds, humidity, wind, and pressure measurements.

Sports data, which was manually created, contains features for NBA Celtics and NHL Bruins games occurring between November 26 to December 18, 2018. Features include location of the game's stadium (e.g. North Station), start time, stop time. This data was collected from NBA and NHL 2018-2019 schedule.

Ride and Weather data provided from:
https://www.kaggle.com/ravi72munde/uber-lyft-cab-prices and is relatively clean.

Regression Modeling

Because the target variable, price, is a continuous/quantitative feature, three linear regression algorithms were implemented to build a predictive model: Multivariate Linear Regression, Lasso Regression, and Ridge Regression. To prepare the ride data for these methods, the data is preprocessed in a variety of ways: a new dummy column is created to represent rides occuring during different time ranges (1/0 if ride did or did not occur within the time range, e.g 8:00 A.M.-11:59 P.M., 12:00-3:59 P.M.), features such as 'id' and 'product_id' were dropped

(redundant features for predictive modeling), dataframe was checked for any missing values, and the categorical features were converted to dummy columns using pandas get_dummies() method, in which one of the dummy columns for each categorical feature is dropped to prevent multicollinearity.

Two arrays are created, X and y, in which X represents a 2D matrix of all independent variables in the dataset, and y, an array of the target variable, price, for each row in the X matrix. The X and y data is split into a training and test set, or hold-out set, in which the training data holds 80% of the total data, and test containing 20%. The training data is now ready to be fitted into each of the regression models, and test data ready to evaluate the model's accuracy in predicting the price. The metrics used to evaluate the test data were:

1. R-squared: the proportion of the variance in the dependent variable that is predictable from the independent variable.
2. Mean Absolute Error: the average magnitude of the errors in a set of predictions
3. Mean Absolute Percentage Error: measure of prediction accuracy
4. Root Mean Square Error: quadratic scoring rule that also measures the average magnitude of the error

Linear Regression

The target variable's distribution is shown below:

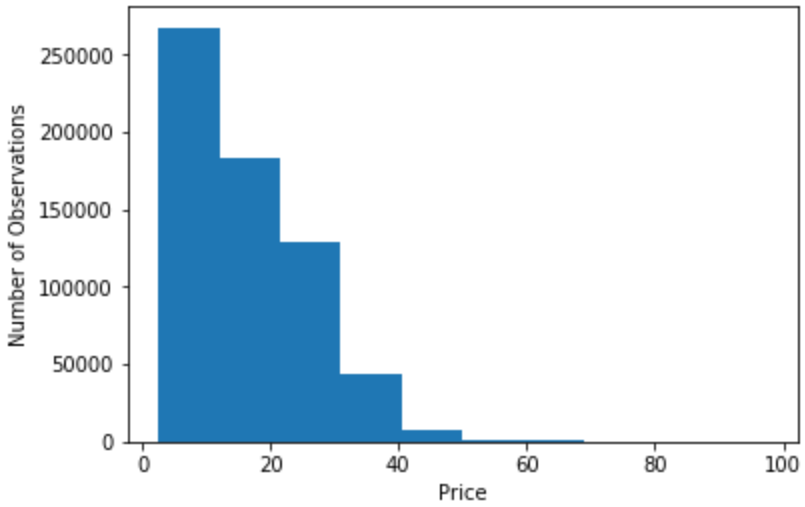

Figure 10. Distribution of Prices

The distribution of prices is heavily skewed to the left, indicating the presence of more smaller priced rides than larger priced rides. The linear regression model trained with this

distribution of y values will not be precise due to the non-linearity. To handle this, the target variable is log-transformed to linearize it and make more precise predictions.

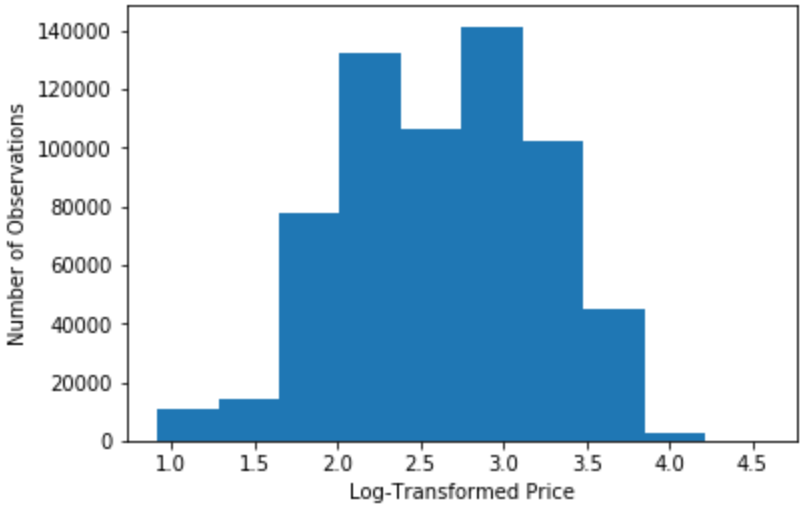

Figure 11. Log-Transformed Distribution of Prices

Using both scikit-learn's LinearRegression() method and statsmodel's OLS() method, the training data was fit into a linear regression for both the original y variable and the log-transformed y variable, applying the inverse function on the predicted values to generate real price values. This is how both linear regression models compare on predicting training data:

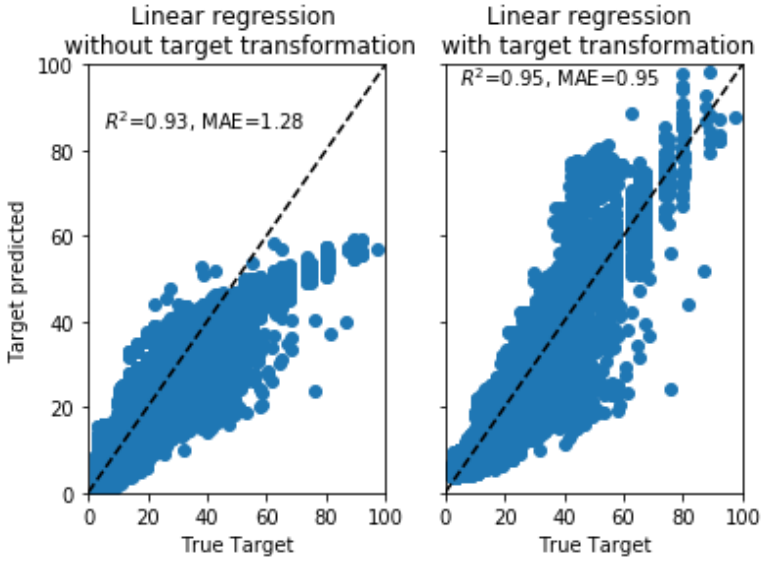

Figure 12. Linear regression with and without target variable transformation.

Clearly the log transformation on the target variable benefits the model to not underpredict higher prices rides, as seen from the non-transformed model, which significantly

does. The R-squared value improves from 0.93 to 0.95, and the mean absolute error is reduced significantly from 1.28 to 0.95.

Looking at the fitted vs. residuals plot for each model:

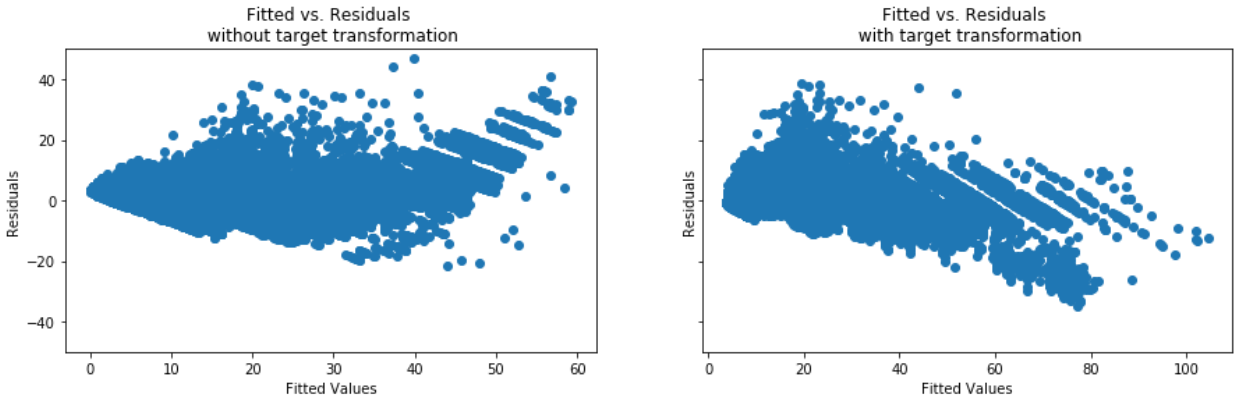

Figure 13. Fitted values vs. Residuals plot with and without transformed target variable.

The residuals for both plots do form a horizontal band around a residual value of 0, and no clear pattern is observed. This suggests that the relationship is linear and the variance of residuals is equal. However, the non-transformed model underpredicts significantly, capping the price predictions at 60$, where their residuals are greater than 20, up to 40. The transformed model handles this issue, precisely predicting prices greater than 60 with residuals near 0, or the regression line. Looking at the distribution of residuals plot, or QQ plot, for the transformed model:

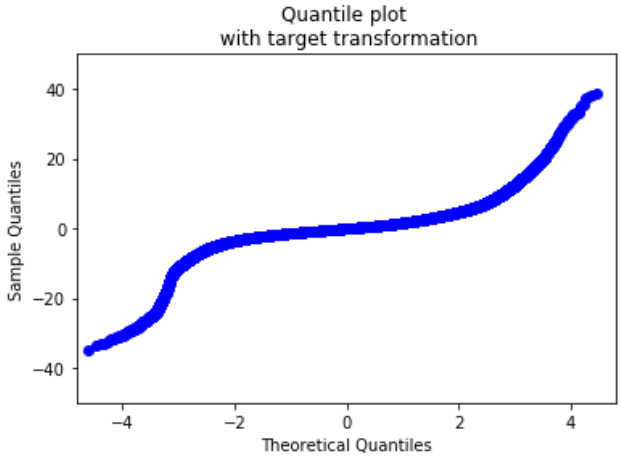

Figure 14. QQ plot of distribution of residuals

The QQ plot suggests, due to the 'S' shape of the distribution, the presence of heavy tails in the distribution, with residuals as high as approximately 40 and as low as approximately -40. These residuals are further analyzed, looking at a leverage vs. standardized residuals plot, to

determine if any data-points in the training data are highly influential, disrupting the regression line and lowering precision by the model:

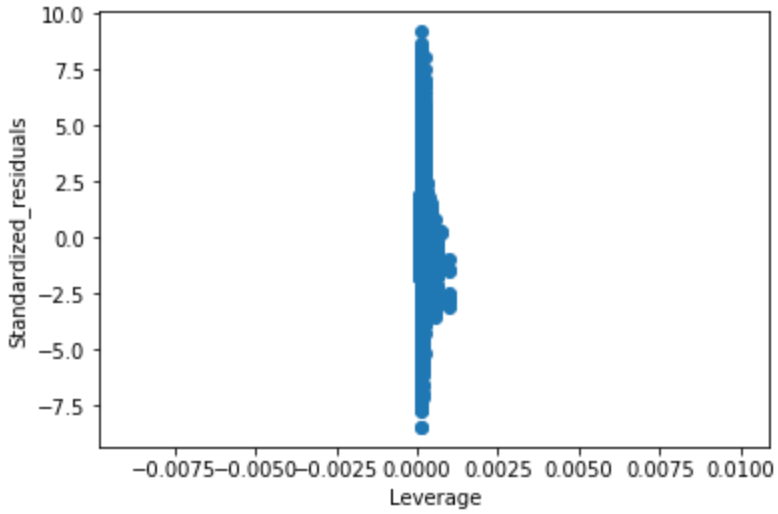

Figure 15. Standardized residual vs. leverage plot.

The plot indicates none of the data-points are highly influential, as not points have both high leverage and a high residual value. Points that have a moderate amount of leverage are clustered around a range of standardized residual values between -2.5 and 2.5. These points in the training data is observed to identify which X variable is extreme compared to the mean, which happened to be surge multiplier of 2.5 and higher, corresponding to significantly higher prices than normal, compared to a mean surge multiplier of 1.0. However, because these points do not have high residuals, they are well aligned to the fitting model, and do not alter the regression line significantly, so these points are kept within the training data. Extreme standardized residuals values are observed in the training data, isolating a data-point with a standardized residual > 9.0, containing a high priced UberPool ride, 40$, going a distance of 3.42 with surge multiplier of 1, for which the average price per mile of an UberPool rides is 5.33$ a mile. This is a clear outlier in the training data, and the datapoint is removed. The log-transformed linear regression is reran with the training data and evaluated using the test data, with metrics shown below:

Table 4. Log-Transformed linear regression evaluation of test data before and after outlier removal.

	R-squared	Mean Absolute Error	Mean Absolute Percentage Error	Root Mean Squared Error

Log-Transformed Linear Model Before Outlier Removal	.945127	1.399127	9.539315	2.194692
Log-Transformed Linear Model After Outlier Removal	.945128	1.399115	9.539239	2.194679

Clearly, removal of the outlier barely improved the model, so a quick trial of removing points with high standardized residual values > 7.5 was done, however, this also barely improved the model as indicated by the metrics for the test data. To pursue better models for more precise predictions, regularization is introduced to filter out features with low importance (feature selection), and help reduce overfitting, as a means to improve the precision of the model for better predictions.

Lasso Regression

The first regularization method implemented was Lasso Regression, which performs L1 regularization, adding a penalty for non-zero coefficients, and penalizes the sum of their absolute values. Prior to fitting the lasso model, the predictors were standardized, meaning the mean of the predictors is set to 0, and standard deviation is set to 1. The scale of the X variables affects how much regularization is applied to a specific variable. Lasso requires specification of a penalty parameter, alpha, so to determine the best alpha value, cross validation on the training data using a grid of alpha values is implemented using GridSearchCV() from sklearn.

Through cross validation, to select the value of alpha that minimizes the cross-validated sum of squared residuals, the lowest alpha in the grid was chosen. This method was repeated for lower and lower alpha values, approaching an alpha of 0. Lasso regression with an alpha value of 0, is the same as normal linear regression, as implemented above. The metrics for lasso regression on the test data with lowering alpha values are calculated:

Table 5. Lasso regression evaluation of test data for lowering alpha values

Alpha	R-squared	Mean Absolute Error	Mean Absolute Percentage Error	Root Mean Squared Error
.01	.845453	2.444336	15.832045	3.683205

.001	.943908	1.443256	9.852084	2.218941
.0005	.945073	1.408761	9.643551	2.203753
0	.945128	1.399115	9.539239	2.194679

According to the evaluation of the test data for lowering alpha values, the lasso regression is not beneficial to the original linear regression model, as the regularization did not improve the model's evaluation of unseen data compared to the normal regression's evaluation of the test data, as the alpha with the best metrics for lasso's predictions on the test data is 0. Another regularization method, Ridge Regression, is implemented to see if the model benefits from optimizing the coefficients in another way.

Ridge Regression

The second regularization method used, Ridge Regression, performs L2 regularization, adding a factor of sum of squares of coefficients to the residuals sum of squares in the original ordinary least squares method from normal linear regression. Once again, the training data is scaled, and a regularization parameter, alpha, must be specified. Cross validation of the training data with a grid of alpha values is performed to determine the best alpha value. Based on the cross validation results, the best value for alpha was determined to be 2.778. Ridge Regression, with this alpha value, was fitted for the log-transformed training data, and the metrics for evaluating the test data are as follows:

Table 6. Ridge regression evaluation of test data

Alpha	R-squared	Mean Absolute Error	Mean Absolute Percentage Error	Root Mean Squared Error
2.778	.945129	1.399106	9.537433	2.194624

Clearly, this model is slightly better, compared to the original linear regression, by optimizing the coefficients, adding penalty equivalent to the square of the magnitude of coefficients. This is the model of choice for evaluating unseen data, as it provides fair weighting on each important and unimportant feature, without completely eliminating them (like in Lasso).

Feature Importance

The coefficients generated by normal, lasso, and ridge regression are shown in the figure below:

	Linear_reg_coef	Lasso_coef	Ridge_ceof
distance	0.175169	0.161968	0.175168
surge_multiplier	0.664398	0.000000	0.663981
temp	0.000155	0.001184	0.000154
clouds	0.014289	0.000000	0.014287
pressure	0.001618	0.002065	0.001616
rain	-0.165179	-0.000000	-0.163952
humidity	0.004713	-0.000000	0.004566
wind	0.000945	0.000000	0.000942
Sports Occurence	-0.001114	0.000000	-0.001110
day_Mon	-0.038976	-0.000000	-0.038969
day_Sat	-0.021571	-0.000000	-0.021554
day_Sun	-0.013511	0.000000	-0.013486
day_Thu	-0.016944	-0.000000	-0.016956
day_Tue	-0.017647	0.000000	-0.017689
day_Wed	-0.009021	0.000000	-0.009060
cab_type_Uber	0.305898	-0.000000	0.292215
destination_Beacon Hill	-0.005297	0.000000	-0.005293
destination_Boston University	0.002138	0.000000	-0.005750
destination_Fenway	-0.009608	-0.000000	-0.017496
destination_Financial District	-0.028673	-0.000000	-0.028665
destination_Haymarket Square	0.011162	-0.000000	0.003270
destination_North End	0.009126	-0.000000	0.001235
destination_North Station	-0.009663	-0.000000	-0.009659
destination_Northeastern University	0.013040	0.000000	0.005150
destination_South Station	0.001920	-0.000000	-0.005970
destination_Theatre District	0.019449	0.000000	0.019451
destination_West End	-0.009987	-0.000000	-0.009983
source_Beacon Hill	-0.004943	-0.000000	-0.004947
source_Boston University	-0.014258	-0.000000	-0.022151
source_Fenway	0.005568	0.000000	-0.002326
source_Financial District	-0.035374	-0.000000	-0.035378
source_Haymarket Square	0.010455	-0.000000	0.002550
source_North End	0.043091	0.000000	0.035185
source_North Station	-0.008365	-0.000000	-0.008374
source_Northeastern University	-0.002984	0.000000	-0.010877
source_South Station	0.025079	0.000000	0.017178
source_Theatre District	0.028038	0.000000	0.028036
source_West End	-0.001643	-0.000000	-0.001651
name_Black SUV	0.404289	0.532603	0.404423
name_Lux	0.122628	0.000000	0.109112
name_Lux Black	0.389719	0.230102	0.376185
name_Lux Black XL	0.741261	0.580783	0.727702
name_Lyft	-0.482561	-0.392451	-0.496036
name_Lyft XL	-0.026367	-0.000000	-0.039872
name_Shared	-0.955848	-0.891060	-0.969305
name_UberPool	-0.851360	-0.491517	-0.851147
name_UberX	-0.743514	-0.383876	-0.743308
name_UberXL	-0.279592	-0.000000	-0.279415
name_WAV	-0.743385	-0.383839	-0.743178
time_range_12:00-3:59 P.M.	0.007476	0.000000	0.007519
time_range_4:00-7:59 A.M.	-0.019724	-0.000000	-0.019670
time_range_4:00-7:59 P.M.	-0.000140	-0.000000	-0.000109
time_range_8:00-11:59 A.M.	-0.050768	-0.000000	-0.050728
time_range_8:00-11:59 P.M.	0.049374	0.000000	0.049408

Figure 16. Coefficients values for independent features from normal, lasso, and ridge regression.

The importance of features is ranked as follows (dummy columns are grouped together) according to the coefficients:

1. Ride-Type (name_'RideType')
2. Surge-Multiplier
3. Cab-Type (cab_type_'cabtype')
4. Distance
5. Rain (not for Lasso)

The rest of the features: temperature, clouds, pressure, wind, sports occurrence, day, destination, source, and time-range, have little to no effect on the target variable, price, relative to the five features featured above. Lasso regression surprisingly minimized the coefficient

associated with rain to 0, indicating that the feature is of low magnitude as compared to the rain coefficient in ridge and linear regression and is excluded from the model. Ridge regression clearly is the model of choice, as the inclusion of the feature rain benefits the model in it's precision of predicting the price in unseen data.