Expedia Hotel Searches Data Science Project

Data Set Description

Expedia has provided a dataset that includes shopping and purchases data. The data are organized around a set of “search result impressions”, or the ordered list of hotels that the user sees *after they search* for a hotel on the Expedia website. The data contain impressions where the hotels are randomly sorted to avoid the position bias of the existing algorithm. The user response is provided as a click on a hotel and/or a purchase of a hotel room.

Learning to rank hotels to maximize purchases using time series analysis and machine learning models. Load and filter dataset using the highest correlated variables to hotel prices which includes

- prop_id (hotel id)
- visitor_location_country_id (country id the customer is located in)
- srch_room_count (number of hotel rooms specified in search)
- date_time (date and time of the search)
- price_usd (the displayed price of the hotel for the given search)
- srch_booking_window (number of future days the hotel stay started from the search date)
- srch_length_of_stay (number of nights stay that was searched)
- srch_saturday_night_bool (+1 if the hotel stay includes a Saturday night with a length of stay is less than or equal to 4 nights; otherwise 0)

Data Preparations

Evaluate the relationships between the highest correlated variables

- The ID of the hotel property and the displayed price of the hotel
- The country ID where the customer for the hotel property is located
- The number of hotel rooms specified in the search given hotel ID and country ID
- Number of future days the hotel stay and displayed the price of the hotel
- Search where hotel stay includes a Saturday night and displayed price of the hotel
- The number of night stays that were searched by the hotel client.

Data Cleaning

When looking at which variables of the dataset contain missing values, we can see that the price competitiveness variables contain by far the most missing variables. The magnitude of the missing values led me to focus on appending this dataset in the following columns:

1. Hotel property characteristics
2. Location attractiveness of hotels
3. User’s aggregate purchase history

We see that the majority of values in some columns are null, to prevent us from making some very bad models it is safe to remove ‘comp1_rate’, ‘comp1_inv’, and ‘comp1_rate_percent_diff’ to the 8th competitive level.

Once we evaluate the missing value after dropping the price competitiveness variables, we can see that the hotel search variables based on the user's aggregate purchase history suffer from no missing data.

Matrix Analysis

Finding the most predictive variables for maximizing hotel purchases starts with printing a correlation matrix. Sorting the highest positive and negative search and hotel variables will give an idea of which variables to select for the time series analysis.

Two portions of the correlation matrix that does not include the diagonal elements are:

1. Upper Triangular (positive correlations)

   To observe the relationship of hotel prices against the country ID where the hotel and customer are located, we can see a strong positive correlation between 'prop_country_id' (ID of the country the hotel is located in) and 'visitor_location_country_id' (ID of the country the customer is located in).

   In addition, the positive correlation between 'srch_room_count' (number of nights stay that was searched) and 'srch_adults_count' (number of adults specified in the hotel room) focuses on the main demographic effect on user aggregate purchasing history. This could be beneficial in strengthening the relationship between hotel prices against search preferences.

2. Lower Triangular (negative correlations)
   To observe the relationship of hotel prices against explanatory hotel search variables, we can see a strong negative correlation between 'srch_length_of_stay' (number of nights stay that was searched) and 'srch_saturday_night_bool' (boolean value of Saturday night with a length of stay less than or equal to 4 nights). This could be beneficial in predicting the trend in hotel prices based on consumer preferences.

Note: test.csv does not contain the columns position, click_bool, gross_bookings_usd, nor booking_bool. Visit Kaggle.com

From this matrix analysis, some variables that I have selected include the property, visitor location, country, and the number of hotel rooms specified in the search with the most data points.

Subset the Dataset

Selected 'prop_id' / 'visitor_location_country_id' / 'srch_room_count' with the most data points, which means these three columns have the maximum weight on the data to simplify my data science exploration.

Load a new dataset that filters these columns: 'prop_id' / 'visitor_location_country_id' / 'srch_room_count':

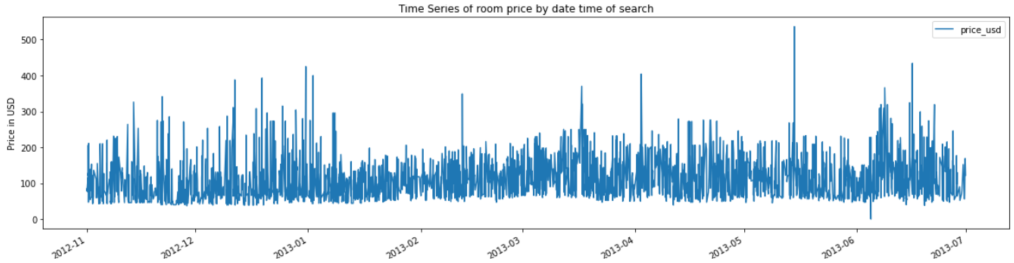
- prop_id (hotel id)
- visitor_location_country_id (country id the customer is located in)
- srch_room_count (number of hotel rooms specified in search)

Define a list of the time series forecasting variables 'date_time' / 'price_usd' / 'srch_length_of_stay' / 'srch_booking window' / 'srch_saturday_night_bool':

- date_time (date and time of the search)
- price_usd (the displayed price of the hotel for the given search)
- srch_length_of_stay (number of nights stay that was searched)
- srch_booking_window (number of days in the future the hotel stay started from the search date)
- srch_saturday_night_bool (+1 if the hotel stay includes a Saturday night with a length of stay is less than or equal to 4 nights; otherwise 0)

In my subset dataset, I decided to randomly generate the hotel id to be 104517 and the country ID of 219, and the number of hotel rooms specified in the search being equal to 1 since we are answering for each individual's search.

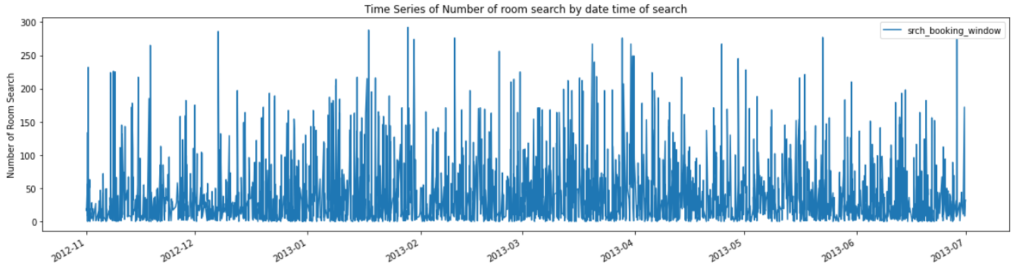
Simple Data Visualization
From the data description of the subset dataset, we see the majority of values in price_usd are < 5584 categorical benchmarks so what you can now do is see your stats for data below this. This also seems like a time series problem, but I want to perform supervised learning, so I decided to convert the timestamps to numerical entities to help me provide a good inference. Now, let's plot the new time series of room prices by date time of hotel search:



I can see the trend and/or seasonality in your dataset, and determined there is too much noise to discover insights on dates against USD price. I can see that my dataset has some auto-correlation features. I see the same trends, repeating at a different slope at the start of the plot. It goes on to become bad at the end, but now I have some hope that I can figure things out.

Time Series Analysis

Line Plot
We can also plot a time series of the number of rooms searched by date and time of hotel search:

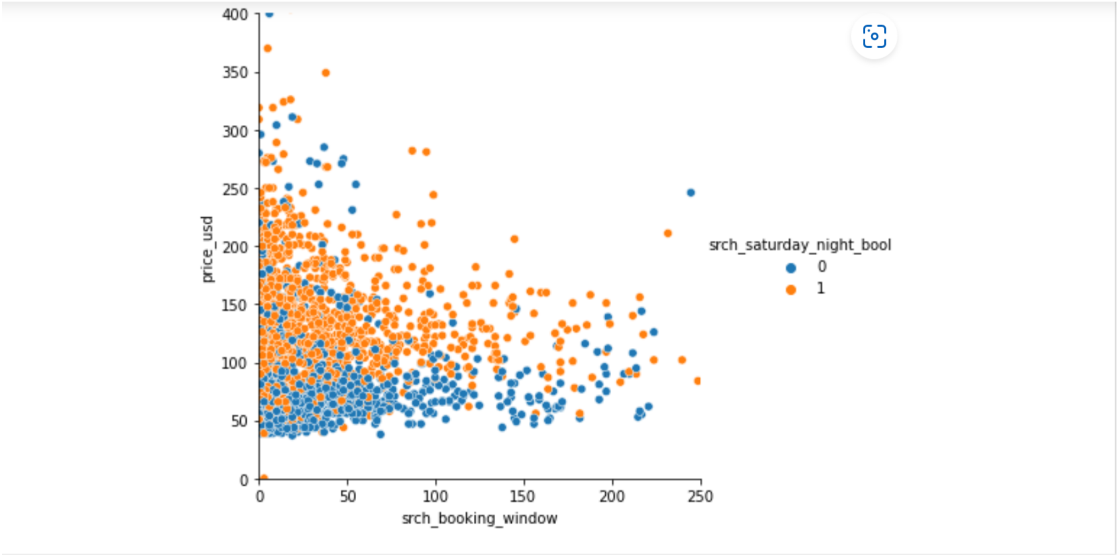


I can see here that there is even more noise when trying to discover insights on dates against the number of room searches. There are many outliers in the hotel search window and it is tougher to find the same trends/seasonalities than dates against hotel prices. The search book windows may repeat but no obvious trends are suggesting that booking windows are consistent year-round.

Scatter Plot

Plotting the relationship between search booking window and hotel price, there is a negative relationship between the two. This means that an increase in the number of days away from the hotel stay by one decreases the hotel price.

I can also set `hue='srch_saturday_night_bool'` to color our points by whether the hotel stay includes a Saturday night stay. This hue argument is very useful because it allows you to express the third dimension of information between the search booking window and hotel price.

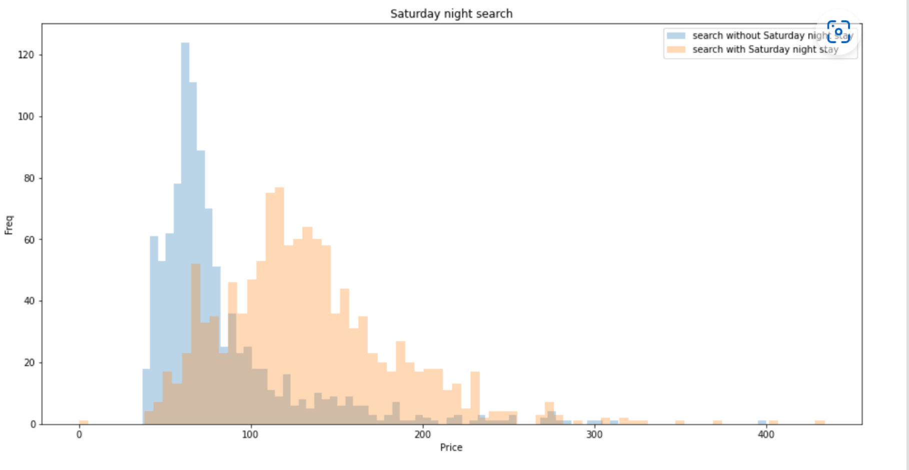


Hotel searches that include a Saturday night stay are generally priced higher than hotel searches that do not include a Saturday night stay. This means customers spend more on a hotel stay if it includes weekend nights, and pay less if the booking window from the hotel stay is larger. Now that I see the relationship between our two numerical variables, we can assess their distributions and their relation to the 'srch_saturday_night_bool'.

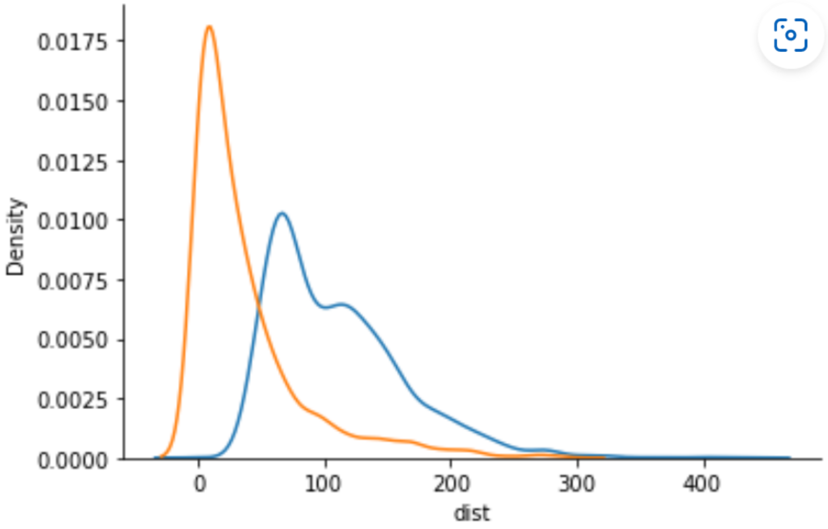
Histogram Plot

Let's plot the histograms of hotel prices searched and search the booking window. Both histograms show the distribution that suffers from left skewness, and that prices and search booking windows suffer from outliers of hotels with higher prices and had more search traffic.

To see how this time series data has been skewed, I can use a highly correlated variable like 'srch_saturday_night_bool' to distinguish the frequency of prices if the hotel search included a Saturday night stay or not.



Plotting 'srch_saturday_night_bool' and the prices by creating an overlapping bar graph. There seems to be some strong relationship between price and search without a Saturday night stay, which can be seen by the peak.



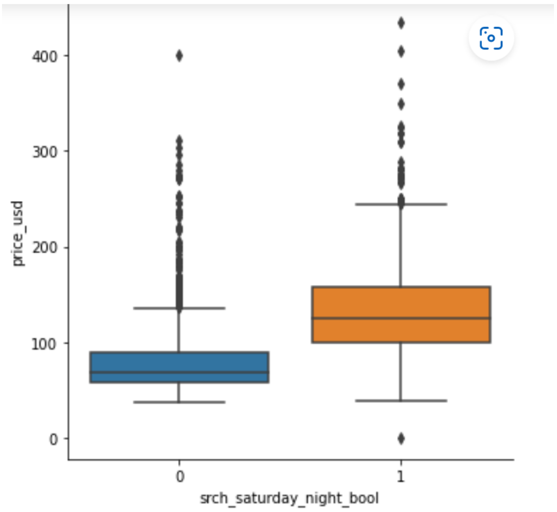
Seeing it in a distribution plot similar to the histogram plot, there seems to be a strong relationship between the distribution of hotel prices and the booking window in the lower bin [0, 100].

To get a better grasp of some potential outliers in our time series dataset, I can use a box and whisker plot to see a five-number summary of the set of data. Seeing this five-number summary will help to distinguish outliers.

Box and Whisker Plot

Plot 'price_usd' and 'srch_booking_window' on a box and whisker plot visualizing columns in the data frame on a single figure. The distribution of search booking windows has a smaller range of time series data than the spread of hotel prices. There are also a large number of outliers from both box and whisker plots suggesting that exorbitant hotel prices and the number of searches happen frequently.

Let's plot to see the distribution of hotel prices concerning 'srch_saturday_night_bool' as a box plot.



The distribution of hotel booking searches with a Saturday night stay has a wider spread and higher average hotel price. A large number of outliers remain, but the hotel booking search without a Saturday night stay had a greater number of outlier prices. This suggests that hotel searches without a weekend stay have a wider distribution of prices.

Pairwise Distribution Plots

From these visualizations above, it is clear that I must transform the data to lower noise and improve the time series forecasting overall. However, it can be difficult to select the best transform for the given prediction problems.

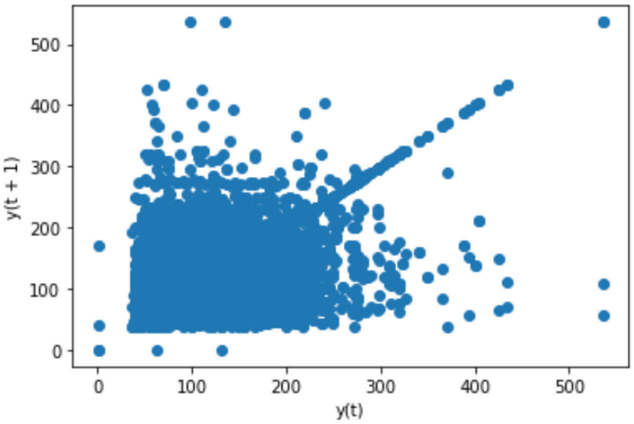
- Predicting future hotel prices based on the number of hotel booking searches.
- Predicting future hotel prices based on the number of days a customer stays.
- Predicting future hotel prices based on whether the hotel booking included a weekend stay.

Feature Engineering

Lag Features

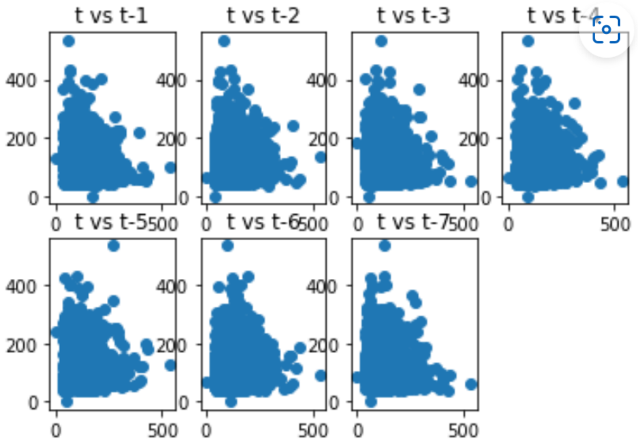
Using lag features, we can predict future hotel prices across time stamps by predicting the price of the hotel at the next time (t+1) given the value at the current time (t). I can also use the concat() function to construct a new dataset with just our new columns and a window size of 3. Check the correctness of the values on the 5th row (array index 4), we can see that 61 USD is the minimum and 246 USD is the maximum of hotel prices in the window of [206.0, 186.0, 61.0].

The scatterplot of the data frame shows how accurate lag features predict hotel prices after one period.

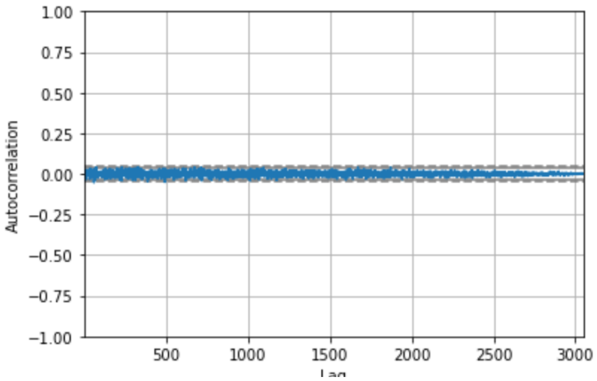


More points tighter into the diagonal line suggests a stronger relationship and more spread from the line suggests a weaker relationship. This shows a relatively weak positive correlation between observations and their lag1 values.

We can create a scatter plot for the observation with each value in the previous seven days using our price data frame.



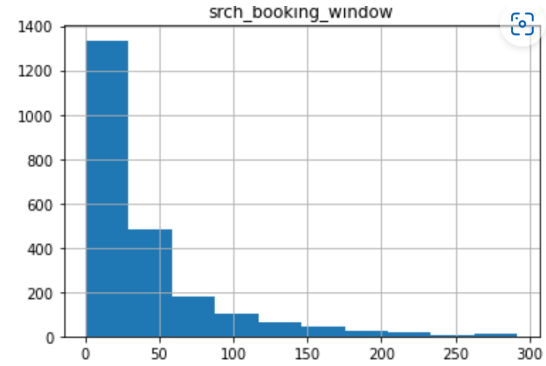
With this slightly positive relationship between an observation with its lag=1 value, we can see that the trend and seasonality information extracted from the series does not seem reasonable.

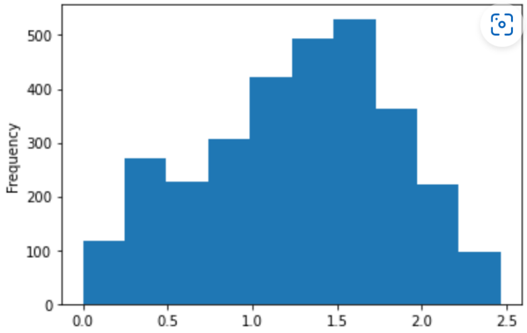


To check for the seasonality trend of the data frame, we can use an autocorrelation plot. Since the correlations fell to near zero, we can see cycles of no correlation. This captures the relationship of observation with past observations in the same and opposite seasons or times of the year. This suggests no seasonality trends in this data frame.

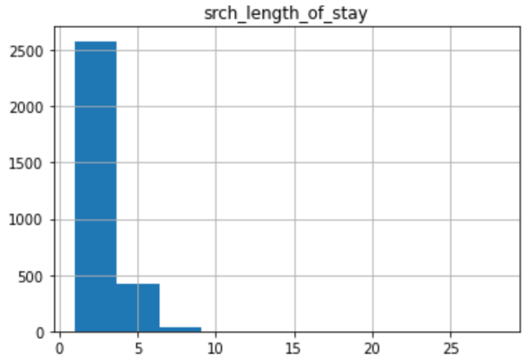
Log Transformation

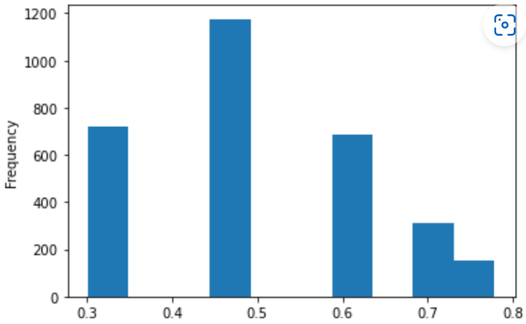
Time series variables with an exponential distribution can be made linear by taking the logarithm of the values. There is an extreme increase on the bar graph and an equally extreme long tail distribution on the histogram.





In the ‘srch_booking_window’ variable, the dataset is non-stationary. This means that the mean and the variance of the observations change over time. This is caused by what appears to be a seasonality component. We see that the logarithmic transform changes the distribution of observations to be perhaps nearly Gaussian.

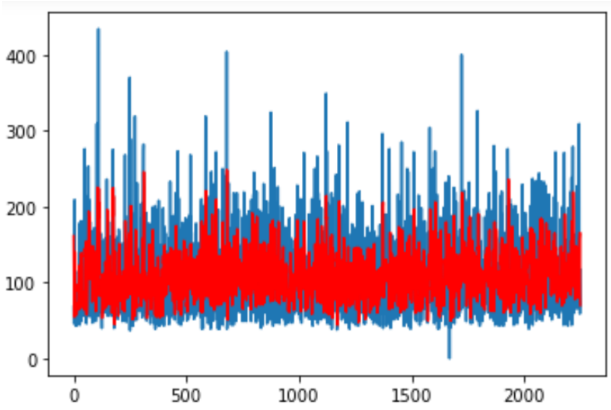


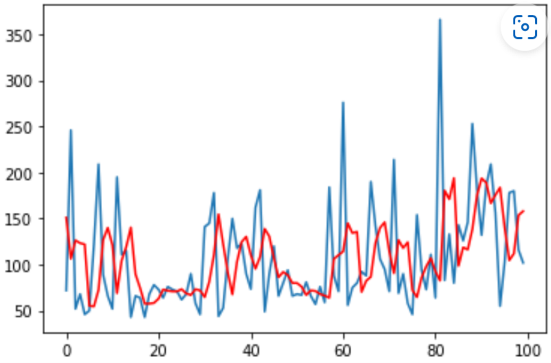


Using another numerical variable in the ‘srch_length_of_stay’ variable, the dataset is non-stationary. Similarly, the mean and the variance of the observations change over time as well as what appears to be a seasonality component. We see that the logarithmic transform changes the distribution of observations to be perhaps nearly Gaussian, but the histogram still contains slight right-skewness suggesting that there is a larger number of occurrences of the lower number of days spent in a hotel and only a few numbers occurrences of the higher number of days spent in a hotel. Finally, we use the clipping method to handle the outliers seen in the 'srch_length_of_stay' variable and reduce bias when predicting hotel prices using this search variable.

Moving Average Smoothing

Smoothing the time series data to remove the fine-grained variation between time steps seen in the lag features. Hope is to reduce noise and improve the signal in time series forecasting. In this case, we use the moving average as a source of new information when modeling a time series forecast as a supervised learning problem. When using this new information when modeling a time series forecast as a supervised learning problem:





As new observations are made available (e.g. daily), the model can be updated and a prediction made for the next day. This result shows a root mean squared error (RMSE) of 61.67 USD per hotel booking search.

The first line plot of the Expedia Hotel Search dataset (blue) with a moving average prediction (red). These raw observations are plotted with the moving average transform overlaid.

In the second line plot, you can see the lag in the transformed dataset plotting the first 100 observations from the Expedia Hotel Search Dataset (blue) with a moving average prediction (red).

Feature Selection

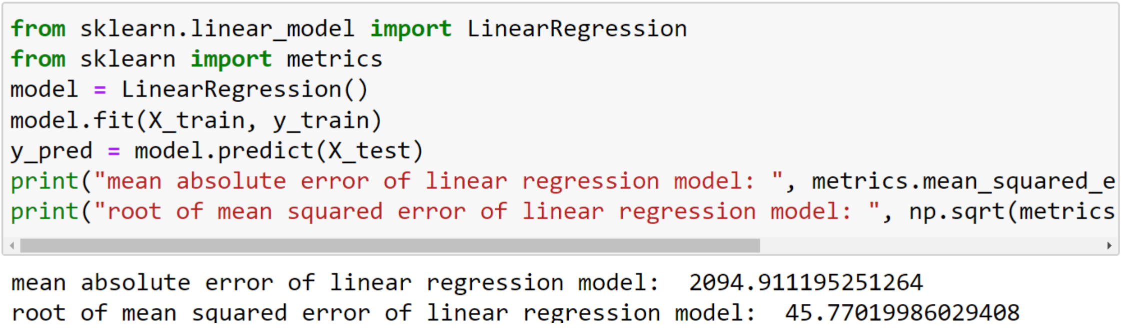
When separating the data into train and test involves randomness, we use the subset of the training set as the dataset for this model. To conform with the time series approach to supervised learning, we set the DatetimeIndex for the subset training set using the set_index() function. Once we extract the feature matrix of the user's aggregate purchase history and target array of hotel prices, we can split the data into training (70%) and testing (30%) data.

Model Building

To determine the type of classification and regression models to utilize, it is necessary to restate our predictive problem and whether it requires a supervised or unsupervised learning model.

The objectives behind our machine learning model include the:

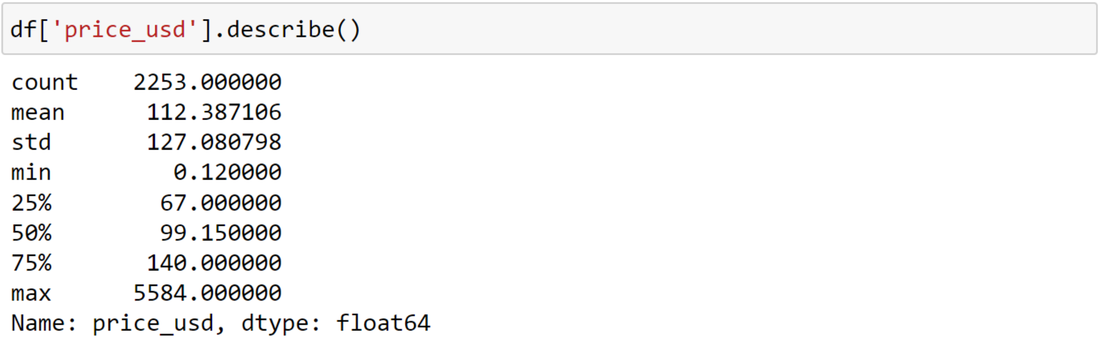
1. Minimize the difference between the actual price and the price estimated using the mean squared error (mse) and the root mean squared error (rmse).



```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("mean absolute error of linear regression model: ", metrics.mean_squared_e
print("root of mean squared error of linear regression model: ", np.sqrt(metrics
```

```
mean absolute error of linear regression model:  2094.911195251264
root of mean squared error of linear regression model:  45.77019986029408
```

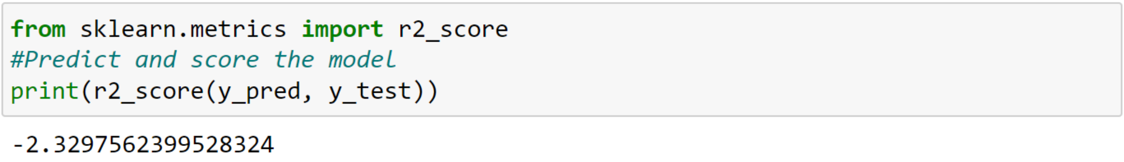
In this linear regression, the mean absolute error is much greater than the average hotel prices that were seen in the summary statistics at 112 USD. This suggests that the hotel price distribution is very widely distributed with a (strong) positive skewness. The root means square error at 45.77 is a huge distance between values predicted by the model and actual values. This indicates that linear regression is not a great option for predicting hotel prices. To find these summary statistics, refer to the data described below:



```
df['price_usd'].describe()
```

```
count    2253.000000
mean      112.387106
std       127.080798
min         0.120000
25%        67.000000
50%        99.150000
75%       140.000000
max      5584.000000
Name: price_usd, dtype: float64
```

In the linear regression, the R2 score can measure the variation in the hotel prices that can be explained by the explanatory hotel variables.



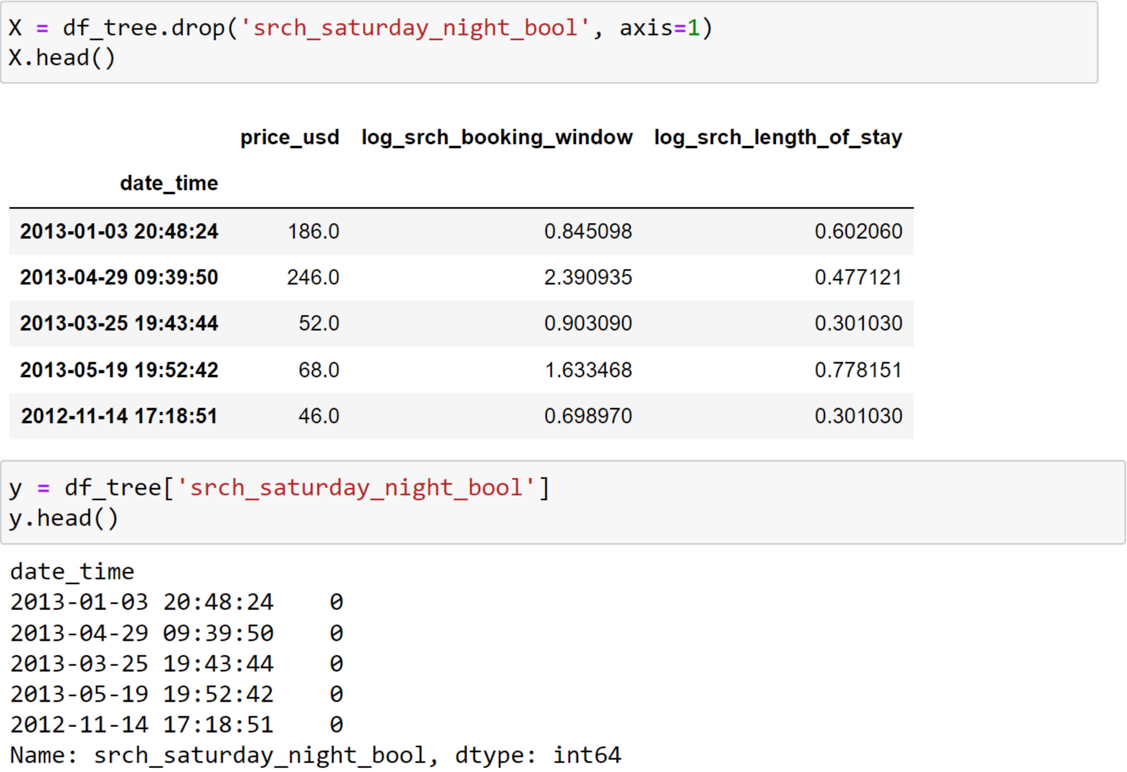
```
from sklearn.metrics import r2_score
#Predict and score the model
print(r2_score(y_pred, y_test))
```

```
-2.3297562399528324
```

This linear regression shows a fit worse than a horizontal line since R2 is negative. Note that R2 is not always the square of anything, so it can have a negative value without violating any rules of math.

2. Build a classification decision-tree model that returns the parameters that satisfy the conditions that a hotel stay includes a Saturday night stay to predict accuracy.

   When using the decision tree classifier *DecisionTreeClassifier* with a max depth of 2 levels and in a random state, we can use this predictive analysis key categorical variable 'srch_saturday_night_bool' which tells you how many hotel residents stayed past Saturday night.



```python
X = df_tree.drop('srch_saturday_night_bool', axis=1)
X.head()
```

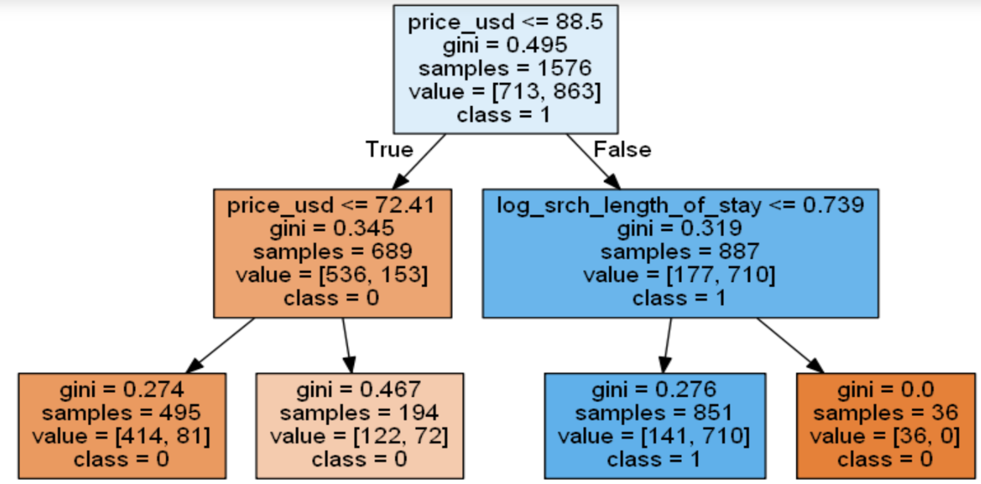
	price_usd	log_srch_booking_window	log_srch_length_of_stay
date_time			
2013-01-03 20:48:24	186.0	0.845098	0.602060
2013-04-29 09:39:50	246.0	2.390935	0.477121
2013-03-25 19:43:44	52.0	0.903090	0.301030
2013-05-19 19:52:42	68.0	1.633468	0.778151
2012-11-14 17:18:51	46.0	0.698970	0.301030

```python
y = df_tree['srch_saturday_night_bool']
y.head()
```

```
date_time
2013-01-03 20:48:24    0
2013-04-29 09:39:50    0
2013-03-25 19:43:44    0
2013-05-19 19:52:42    0
2012-11-14 17:18:51    0
Name: srch_saturday_night_bool, dtype: int64
```

Once we extract the feature matrix of user aggregate purchasing history and the target array of a customer staying at hotels longer than 4 nights (and over the weekend), we can see if the hotel prices have a significant impact on search criteria. Here, we have the decision tree model below:

Out[125]:



Interpret the decision tree to the result

In the first node, the hotel price that falls below and is equal to 88.5 USD has 1576 samples. The hotel prices that satisfy the condition were 863 hotel prices, while 713 hotel prices did not.

In the right second node, the first node splits the customer length of stay that falls below 5 nights (10^0.739 equals about 5) with a sample of 180 instances.

- The customer length of stay that rises above 5 nights has a 710 sample.
- The customer length of stay that falls below and equal to 5 nights has a 177 sample.

This means that hotel prices that were above 88.5 USD and staying at a hotel shorter than 5 nights result in 177 customers who stayed for a Saturday night stay.

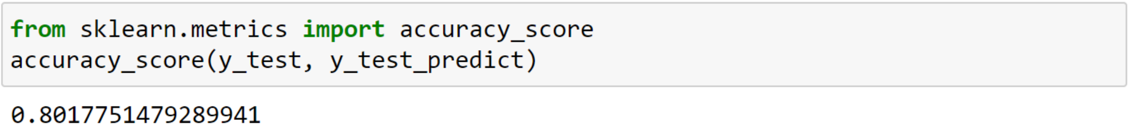
In the left second node, the first node splits the hotel prices that were less than and equal to 72 USD with a sample of 689 instances.

- The hotel prices below and equal 72 USD have a 536 sample size with a Saturday night stay.
- The hotel prices above 72 USD have a 153 sample size with a Saturday night stay.

This means that prices that fell below 72 USD had a 491 sample size that included a Saturday night stay.

Interpret Accuracy Score of Decision Tree

Given this decision tree model, we can use it to predict the values for the target variable, which is a customer staying in a hotel over the weekend and over 4 nights.

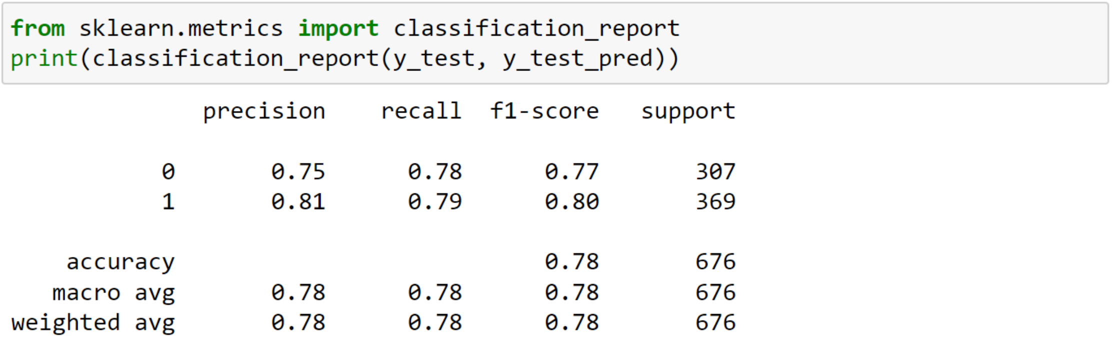


```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_test_predict)
```

```
0.8017751479289941
```

The accuracy of the decision tree model in predicting hotel prices with a Saturday night stay is above 80%. This performs much better than the linear regression model, which indicates a good-performing model for predicting hotel stays given a user's aggregate purchasing history and hotel prices.

3. Build a classification logistic regression model using the ‘liblinear’ or linear type to print out classification metrics (precision, recall, and F-1 major for both categories).



```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.78      0.77       307
           1       0.81      0.79      0.80       369

    accuracy                           0.78       676
   macro avg       0.78      0.78      0.78       676
weighted avg       0.78      0.78      0.78       676
```

Interpret Classification Metrics from Logistic Regression

It shows our precision, recall, and F-1 major for each category. Support means the sample size of 676 instances. Out of the 676 instances, 313 of them are in category 0 and 363 in category 1.

In the categories that were 0:

- The precision is 0.74 is the ratio of hotel stays with a Saturday night stay that was predicted to be accurate, which is not great.
- The recall is 0.81 is the ratio of hotel stays with a Saturday night stay identified from those that had real Saturday night stays, which is better.
- The f-1 score is 0.78 which follows the high precision and recall. The f-1 score is still far away from 1 indicating a classification model that could be improved upon.

In the categories that were 1:

- The precision is 0.81 is the ratio of hotel stays with a Saturday night stay that was predicted to be accurate, which is good.
- The recall is 0.79 is the ratio of hotel stays with a Saturday night stay identified from those that had real Saturday night stays, which is still good.
- The f-1 score is 0.80 which follows the high precision and recall. The f-1 score is closer to 1, which indicates the classification model is improving.

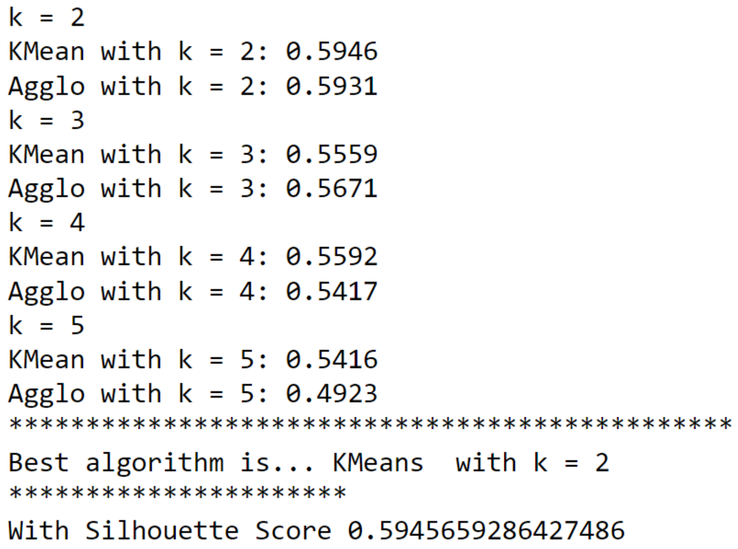
4. Build a partition method (k-mean) clustering model that groups the hotel prices that are dependent on the hotel booking window for customer searches.

Partition Method (K-Mean) Clustering Model

The classification and regression models above predict whether a customer stayed at a hotel through Saturday against hotel prices and other user aggregate purchasing history.

In a clustering mode, we can evaluate the unsupervised learning problem of customer segmentation - whether their hotel search included a Saturday night stay and over 4 nights - based on similar explanatory variables.

In applying the k-mean clustering algorithm to identify two clusters of hotel searches:



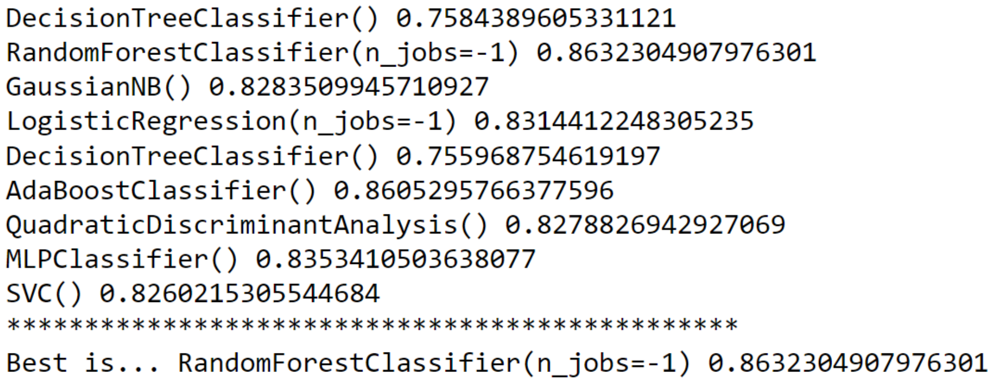
```
k = 2
KMean with k = 2: 0.5946
Agglo with k = 2: 0.5931
k = 3
KMean with k = 3: 0.5559
Agglo with k = 3: 0.5671
k = 4
KMean with k = 4: 0.5592
Agglo with k = 4: 0.5417
k = 5
KMean with k = 5: 0.5416
Agglo with k = 5: 0.4923
***********************************************
Best algorithm is... KMeans  with k = 2
**********************
With Silhouette Score 0.5945659286427486
```

A silhouette score of 0.59 is the distance between the cluster of hotel prices with and without a Saturday night stay, which indicates that our clustering model is far from being a perfect cluster.

Conclusion

Prediction Results

In the linear regression model, Expedia's search dataset does not have a linear relationship between hotel prices and explanatory hotel variables and fits worse than a horizontal line since R2 is negative. Since we have determined that a linear regression model is not good at predicting a numerical value like price, we can predict the effect of home prices on user aggregate purchasing history. Set the DatetimeIndex for the subset training set using the set_index() function for classification models. In comparing classifiers' performances in terms of average AUC:



```
DecisionTreeClassifier() 0.7584389605331121
RandomForestClassifier(n_jobs=-1) 0.8632304907976301
GaussianNB() 0.8283509945710927
LogisticRegression(n_jobs=-1) 0.8314412248305235
DecisionTreeClassifier() 0.755968754619197
AdaBoostClassifier() 0.8605295766377596
QuadraticDiscriminantAnalysis() 0.8278826942927069
MLPClassifier() 0.8353410503638077
SVC() 0.8260215305544684
***********************************************
Best is... RandomForestClassifier(n_jobs=-1) 0.8632304907976301
```

The Random Forest Classification model has the highest accuracy score at 86.32% in terms of average AUC. In creating a random forest classifier to estimate 100 trees, a maximum depth of 2 leaves, and ensuring it is in a random state, the accuracy of the classifier model is well above 80%. Using logistic regression and decision-tree classifier also did a fine job at estimating the relationship between 'srch_saturday_night_bool' and user aggregate purchasing history as well as hotel prices, but can also be used to make a prediction about a categorical variable versus a continuous one. Using the classification and regression models above predict whether a customer stayed at a hotel through Saturday against hotel prices and other user aggregate purchasing history.