Expedia Hotel Searches Data Science Project

Data Preparations

Data Cleaning

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

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

We see that most values in the price competitiveness variables are null, which helps to avoid low-performing 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

I conducted a matrix analysis to evaluate variables against hotel purchases sorting the highest positive and negative searches. Hotel variables 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, we can see a strong positive correlation between 'prop_country_id' (ID of the country the hotel is in) and 'visitor_location_country_id' (ID of the country the customer is in).

   The positive correlation between 'srch_room_count' (number of hotel rooms specified in search) and 'srch_adults_count' (number of adults specified in the hotel room) focuses on the main demographic effect on user aggregate purchasing history. 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 users searched) and 'srch_saturday_night_bool' (boolean value of Saturday night with a length of stay less than or equal to 4 nights). Beneficial in predicting the trend in hotel prices based on user preferences.

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

Some variables 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 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 users 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 less than or equal to 4 nights; 0 if otherwise)

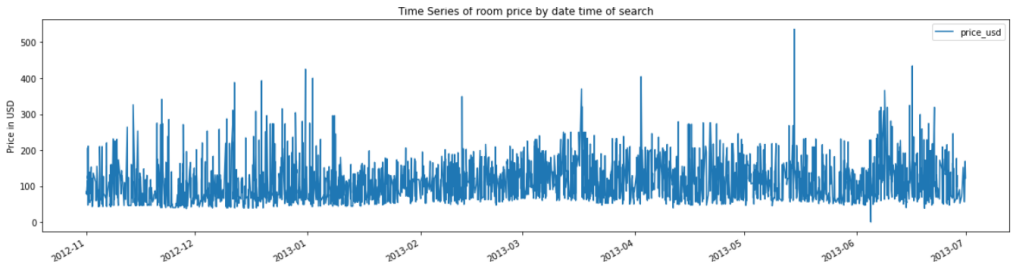
In my subset dataset, the hotel id is 104517, the country ID is 219, the number of hotel rooms specified in the search was equal to 1, and the number of adults searching for a room was equal to 2. All these values chosen have the maximum weight on the variables in the dataset.

Simple Data Visualization
We see the majority of values in `price_usd` are `< 5584` categorical benchmarks. View the data description of hotel price data.

```
df = df.loc[df['price_usd'] < 5584]
df['price_usd'].describe()
```

```
count    2252.000000
mean      109.957438
std        53.391831
min         0.120000
25%        67.000000
50%        99.075000
75%       140.000000
max       434.000000
Name: price_usd, dtype: float64
```

Convert the timestamps to numerical entities to help provide a basis for a time series analysis. Now, let's plot the new time series of room prices by date time of hotel search:

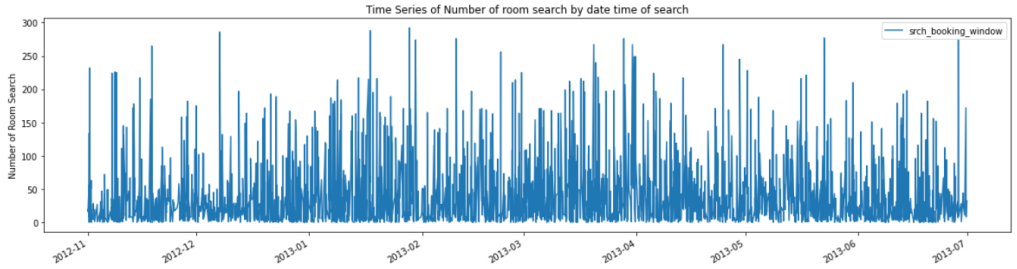


I do not see a seasonality trend in this relationship. It is partially a result of too much noise, which hinders my ability to discover insights on dates against USD prices. The dataset has some auto-correlation features and similar trends, repeating at a different slope at the start of the plot. However, the prices fluctuate less and stay relatively lower during the non-holiday months (January to May 2013).

Visualize 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:

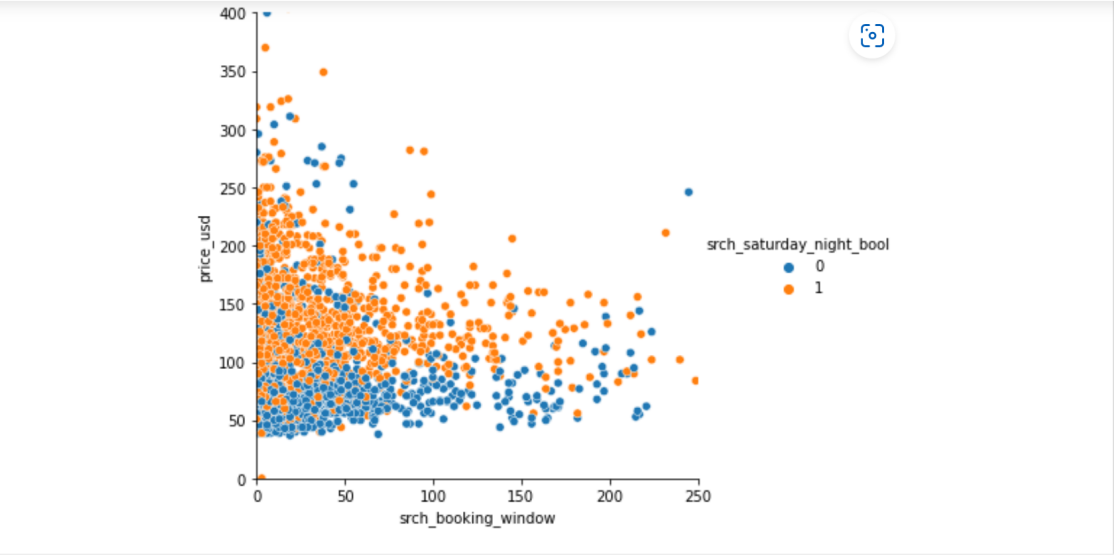


There is even more noise when trying to discover insights on dates against the number of room searches. There are many outliers in the hotel search window, which makes it tougher to find any seasonality trends. There are no trends that suggest booking windows rise or fall year-round.

Scatter Plot

There is a slightly negative relationship between search booking windows and hotel prices. 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 beneficial because it allows you to express the third dimension of information between the search booking window and hotel price.

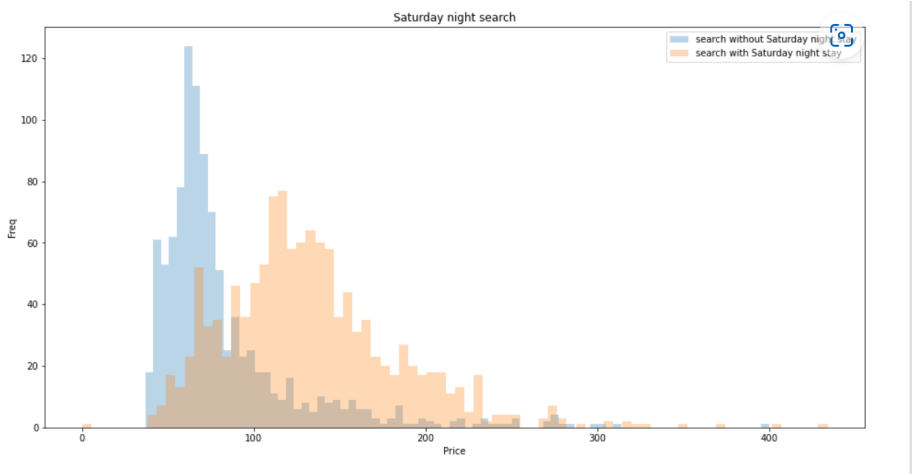


Hotel searches that include a Saturday night stay are priced higher than hotel searches without a Saturday night. Customers spend more on a hotel stay if it includes weekend nights, and pay less if the booking window from the hotel stay is larger. Now that I see the relationship between our two numerical variables, we can assess their distributions and their relation to the boolean variable.

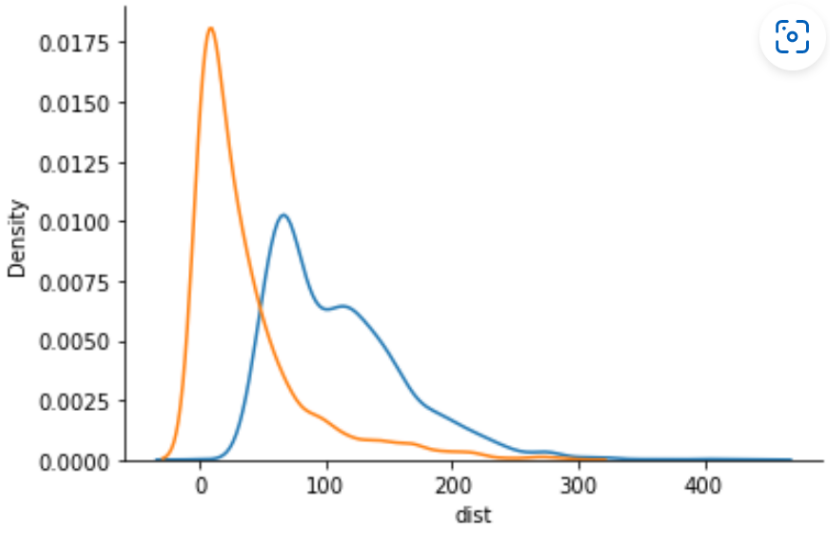
Histogram Plot

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

To see how this time series data compares to 'srch_saturday_night_bool' distinguishes the frequency of prices if the hotel search included a Saturday night stay or not.



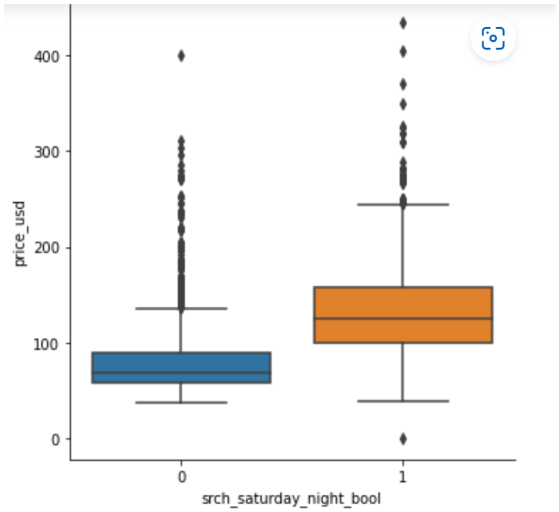
In the density plot, we can see a similar trend:



The distribution and histogram plot display a strong relationship between the distribution of hotel prices and the booking window in the lower bin [0, 100]. I can use a box and whisker plot to see a five-number summary, which helps to distinguish outliers in our time series dataset.

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. Outliers from both box and whisker plots suggest that exorbitant hotel prices and the number of searches happen frequently.

Let's plot 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 higher mean prices. The hotel booking search without a Saturday night stay has more outlier prices, which 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. The best transform for the given prediction problems is:

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

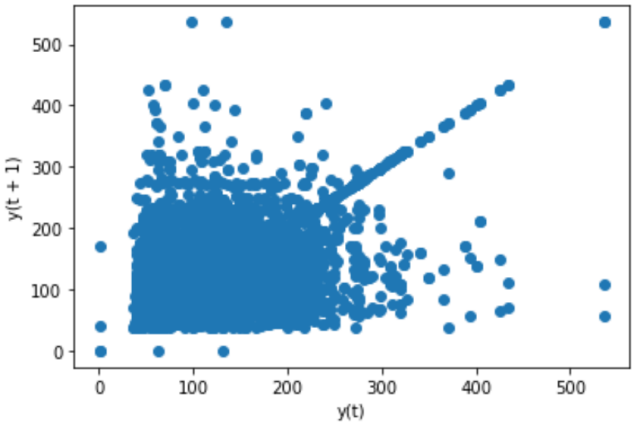
Time Series Analysis - Feature Engineering

Lag Features
Using lag features, we can predict future hotel prices across time stamps by predicting the hotel price at the next time (t+1) given the price 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.

	min	mean	max	t+1
3152	NaN	NaN	NaN	186.0
10560	NaN	NaN	NaN	246.0
11127	NaN	NaN	NaN	52.0
19171	NaN	NaN	NaN	68.0
25358	52.0	161.333333	246.0	46.0

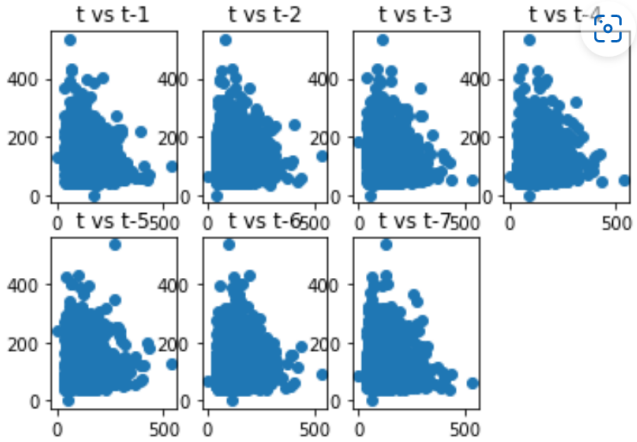
Checking the correctness of the values on the 5th row (array index 4), we can see that 52 USD is the minimum and 246 USD is the maximum of hotel prices in the window of [52, 161.33, 246] with a predicted future price of 46 USD.

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

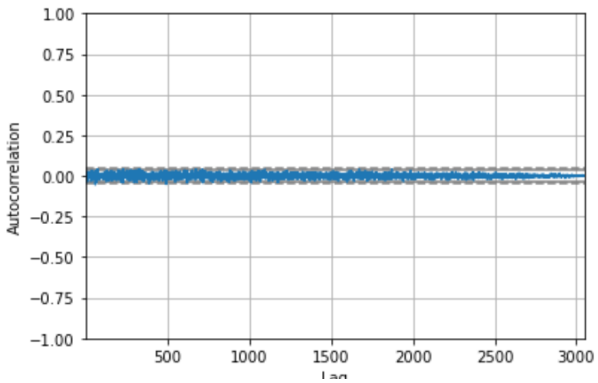


More points tighter into the diagonal line suggests a stronger relationship. However, more spread from this line means a relatively weak positive correlation between observations and their lag1 values.

Create a scatter plot of hotel prices with each value in the previous seven days using the new dataset above.



With this slightly positive relationship between the current price and its lag=1 value, a sign for seasonality trends extracted from this series does not seem reasonable.

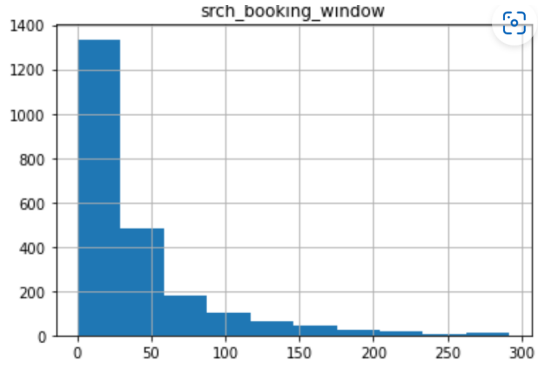


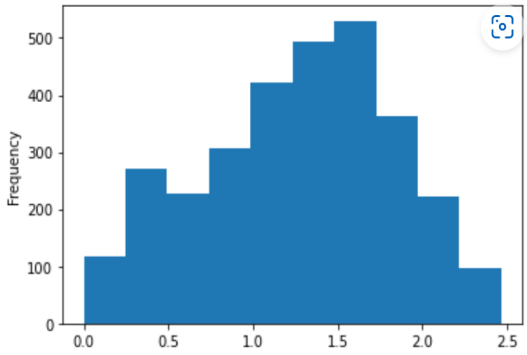
Since the autocorrelations fell to near zero, we can see cycles of no correlation, suggesting no seasonality trends in this data frame.

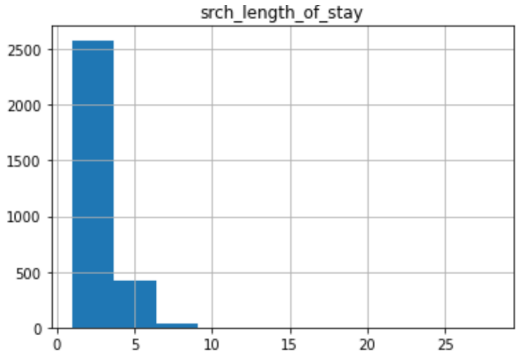
Log Transformation

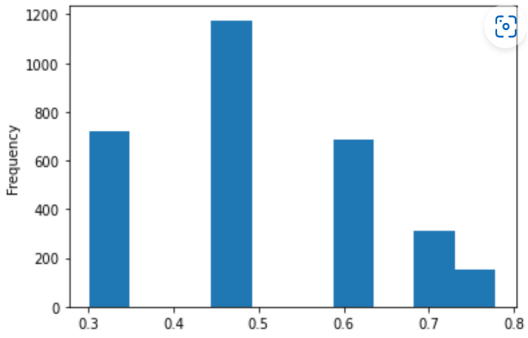
Time series variables with an exponential distribution can be made linear by taking the logarithm of the values. There is an extreme increase on the bar graph and an equally extreme extended tail distribution on the histogram.

In the 'srch_booking_window' and 'srch_length_of_stay' variables, both are non-stationary. There does seem to be a seasonality component. We see that the logarithmic transform changes the distribution of observations to be perhaps nearly Gaussian.









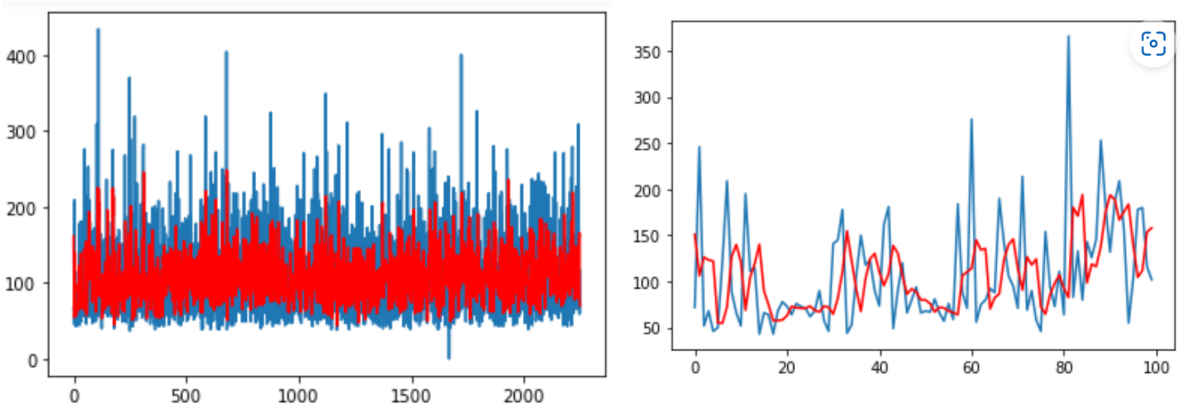
The 'srch_length_of_stay' histogram is a Gaussian distribution, but still contains slight right-skewness suggesting that there are more occurrences of the lower number of days spent in a hotel than the numbers occurrences of the higher number of days spent in a hotel.

Moving Average Smoothing

Smoothing the time series data to remove the fine-grained variation between time steps seen in the lag features reduces noise and improves signal in time series forecasting. Moving average is a source of new information when modeling a time series forecast as a supervised learning problem.

The left line plot shows the full Expedia Hotel Search Dataset (blue) with a moving average prediction (red). These raw observations are plotted with the moving average transform overlaid.

The right line plot shows the lag in the transformed dataset of the first 100 observations from the Expedia Hotel Search Dataset (blue) with a moving average prediction (red).



The model forecasts prices for the next day as new prices are made available (e.g., daily, monthly, yearly). This result shows a root mean squared error (RMSE) of 61.67 USD per hotel booking search. An RMSE above 60 USD indicates the predicted hotel prices have relatively high variability from the mean hotel price.

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 apply this time series approach to a supervised learning problem, we need to set the DatetimeIndex for the subset training set using the set_index() function.

I extracted the feature matrix *(X)* containing the user's aggregate purchase history - the number of future days before the hotel stay, the number of hotel stays that include a Saturday night, and the number of night stays searched. I also extracted the target array *(y)* of hotel prices.

date_time	srch_saturday_night_bool	log_srch_booking_window	log_srch_length_of_stay
2013-01-03 20:48:24	0	0.845098	0.602060
2013-04-29 09:39:50	0	2.390935	0.477121
2013-03-25 19:43:44	0	0.903090	0.301030
2013-05-19 19:52:42	0	1.633468	0.778151
2012-11-14 17:18:51	0	0.698970	0.301030

I split the data into training (70%) and testing (30%) datasets for machine learning models.

```
date_time
2013-01-03 20:48:24    186.0
2013-04-29 09:39:50    246.0
2013-03-25 19:43:44     52.0
2013-05-19 19:52:42     68.0
2012-11-14 17:18:51     46.0
Name: price_usd, dtype: float64
```

```
The shape of X_train is: (1576, 3)
The shape of X_test is: (676, 3)
The shape of y_train is: (1576,)
The shape of y_test is: (676,)
```

Model Building

To assess the selected features from this subset dataset, we can run a linear regression model to evaluate the predictive capabilities of users' aggregate purchasing history on hotel prices.

Linear Regression

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 in the summary statistics at 112 USD. This higher average price means a wider distribution with a (strong) positive skewness. The rmse at 45.77 is a large distance between the predicted and actual hotel prices. Therefore, linear regression is not effective at predicting prices.

```
count    2252.000000
mean      109.957438
std        53.391831
min         0.120000
25%        67.000000
50%        99.075000
75%       140.000000
max       434.000000
Name: price_usd, dtype: float64
```

The R2 score can measure the variation in the hotel prices 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
```

We can see here that the linear regression fits worse than a constant function that always predicts the mean of the data at a -2.33 score.

Comparing Classifiers' Performance

I have determined that linear regression is not good at predicting a numerical value like price.

I changed my original problem by predicting future occupancy with a Saturday night or more than four nights based on hotel price, search length of stay, and search booking window.

First, set the DatetimeIndex for the subset dataset using the set_index() function for classification models predicting the boolean variable 'srch_saturday_night_bool'

```
                     price_usd  srch_saturday_night_bool  log_srch_booking_window  log_srch_length_of_stay
date_time
2013-01-03 20:48:24      186.0                         0                 0.845098                 0.602060
2013-04-29 09:39:50      246.0                         0                 2.390935                 0.477121
2013-03-25 19:43:44       52.0                         0                 0.903090                 0.301030
2013-05-19 19:52:42       68.0                         0                 1.633468                 0.778151
2012-11-14 17:18:51       46.0                         0                 0.698970                 0.301030
...                        ...                       ...                      ...                      ...
2013-01-02 19:36:56      107.0                         1                 1.491362                 0.602060
2013-02-23 12:27:10       81.0                         0                 1.869232                 0.301030
2012-11-17 17:53:21       59.0                         0                 1.531479                 0.845098
2013-04-26 18:52:31       71.0                         0                 1.778151                 0.477121
2013-02-23 11:03:49      117.0                         1                 1.681241                 0.477121
```

Feature matrix *(X)*

date_time	price_usd	log_srch_booking_window	log_srch_length_of_stay
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

Target Array *(y)*

```
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
```

Separating the data into a training set of 70% and a test set size of 30%.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Importing several classifiers included in the sklearn library

```
import sklearn as sk
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

clfs = [sk.ensemble.RandomForestClassifier(n_jobs=-1),
        sk.naive_bayes.GaussianNB(),
        sk.linear_model.LogisticRegression(n_jobs=-1),
        sk.tree.DecisionTreeClassifier(),sk.ensemble.AdaBoostClassifier(),
        QuadraticDiscriminantAnalysis(),MLPClassifier(),SVC()]
```

I divided the dataset into ten subsets. One of the ten subsets is a test set, and the other nine are in a training set.

```
nfolds = 10
```

```
from sklearn.model_selection import KFold
kf = KFold(n_splits=nfolds, random_state=0, shuffle=True)
```

Compare the classifiers' performances in average Area Under the Curve (AUC) score:

```
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.

Classifier training and evaluation in classification prediction using cross-validation
Calculate the average AUC score across all ten trials. Make sure to define 'n_job=-1' to use all processors and avoid the machine learning model getting stuck.

```
import sklearn as sk
avgCV_AUC = sk.model_selection.cross_val_score(cl, X, y, cv=kf, n_jobs=-1,
                                               scoring='roc_auc').mean()
```

```
avgCV_AUC
```

```
0.8260215305544684
```

The average AUC score across all ten subsets is 0.826. It ranks a random positive outcome (the ability to determine a person's hotel stay includes a Saturday night based on the test set) more than 80% of the time. This score suggests a better-performing model and distinguishes between the positive and negative classes well.

Random Forest Classification Model

Create a Random Forest Classifier that has 100 trees, a maximum depth of 2 leaves, and is in a random state.

```
# creating a RF classifier
model = RandomForestClassifier(n_estimators = 100, max_depth=2, random_state=0)

# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
model.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = model.predict(X_test)
```

Summarize the subset dataset and calculate the accuracy

Summarize the training sets for any missing values and the shape of the dataset.

```
# Shape of training data (num_rows, num_columns)
print(X_train.shape)

# Number of missing values in each column of training data
missing_val_count_by_column = (X_train.isnull().sum())
print(missing_val_count_by_column[missing_val_count_by_column > 0])
```

```
(1576, 3)
Series([], dtype: int64)
```

The Random Forest Classifier is now ready to predict a person's hotel stay with a Saturday night and more than four nights.

```
# using metrics module for accuracy calculation
from sklearn import metrics
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
```

```
ACCURACY OF THE MODEL:  0.8076923076923077
```

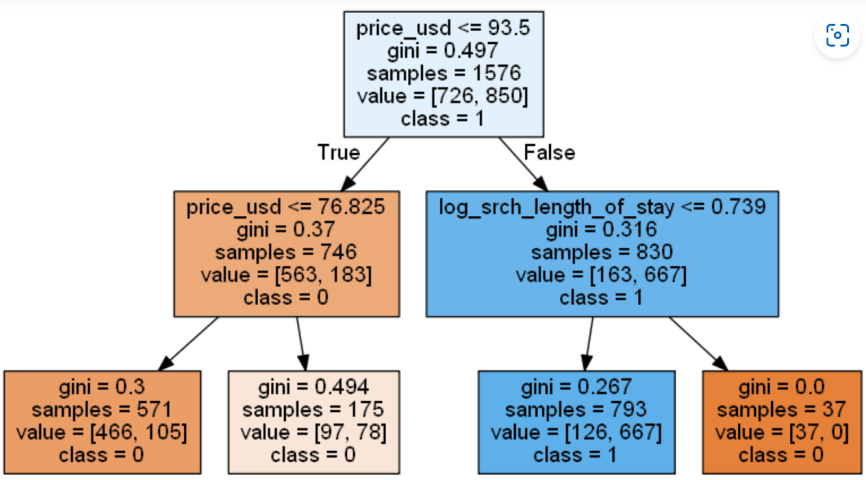
The accuracy of the Random Forest Classifier is slightly above 80%, which means the model performs well.

Decision Tree Classification Model

The goal is to build a decision-tree classification model that returns the parameters that satisfy the conditions that a hotel stay includes a Saturday night stay to predict accuracy.

The decision-tree classifier defines a maximum depth of two nodes and a random state. The tree predicts the number of people whose hotel stay includes a Saturday night and more than four nights.

Out[52]:



Hotel price that falls below and is equal to 93.5 USD for 1576 samples in the first node. The hotel prices that satisfy the condition were 850 hotel prices, while 726 hotel prices did not.

The first node splits the hotel prices that were less than and equal to 76.82 USD with a sample of 725 instances in the left second node. The hotel prices are below and equal to 76.82 USD for 563 samples with a Saturday night stay. The hotel prices above 76.82 USD have a 183 sample size with a Saturday night stay.

Hotel prices fell below or equal to 76.82 USD for a 571 sample including a Saturday night. Hotel prices rose above 76.82 USD for 175 samples not including a Saturday night.

The first node splits the user length of stay that falls below five nights (10^0.739 equals about 5) with a sample of 163 instances in the right second node. The user length of stay rises above five nights for 667 samples. The user length of stay falls below and is equal to 5 nights for 163 samples.

The user length of stay that fell below or equal to five nights had a 793 sample, not including a Saturday night. The user length of stay rose above five nights for 37 samples including a Saturday night.

Interpret Accuracy Score of Decision Tree Model
The goal is to find how accurate this decision tree model is at predicting the number of people whose hotel stay includes a Saturday night and more than four nights.

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

```
0.8165680473372781
```

The accuracy of the decision tree model at predicting hotel prices with a Saturday night stay is above 80%. This model performs much better than the estimated average AUC score for the DecisionTreeClassifier.

Logistic Regression Classification Model

The logistic regression has a high average AUC score, which means a better-performing model and distinguishes between the positive and negative binary classes. This model determines the best-predicted weights as close as possible to all actual responses.

The goal is to 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       0.73      0.81      0.77       306
           1       0.83      0.75      0.79       370

    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. Support means the sample size of 676 instances. 306 of the samples are in category 0 and 370 are in category 1.

In the categories that were 0:

- The precision is 0.73 is the ratio of hotel stays with a Saturday night predicted to be true, which could be better.
- The recall is 0.81 is the ratio of hotel stays with a Saturday night identified from those that had real Saturday night stays, which is better.
- The f-1 score is 0.77, which is still far from 1. This f-1 score means that this classification model could be improved upon.

In the categories that were 1:

- The precision is 0.83 is the ratio of hotel stays with a Saturday night predicted to be true, which is better than in categories that were 0.
- The recall is 0.75 is the ratio of hotel stays with a Saturday night identified from those that had real Saturday night stays, which is worse than in categories that were 0.
- The f-1 score is 0.79, which is closer to 1. This f-1 score means that this classification model has slightly improved.

Despite the capabilities of the logistic regression model predicting binary classes, we can see in comparing classifiers that the logistic regression has an accuracy AUC score greater than 80%.

Partition Method (K-Mean) Clustering Model

The classification and regression models above predict a hotel stay on a Saturday night and over four nights based on the feature matrix.

In a clustering mode, it carries out the same unsupervised learning problem around user segmentation - whether their hotel stays included a Saturday night or over four nights - based on the same feature matrix.

The k-mean clustering algorithm identifies 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
```

With a silhouette score of 0.59, the data points are not compact within the cluster where it belongs to a person with either a hotel stay with or without a Saturday night stay.

Conclusion

Finding 1

Expedia searches for hotel prices that have no seasonality trends and it has to do with the noise, which hinders my ability to discover insights on dates against USD prices. Many outliers in the hotel search window compared to the discrepancy of hotel prices makes it tougher to find any seasonality trends. In the negative relationships between the hotel search explanatory variables and hotel prices, we can see that variance between the hotel prices and these variables indicate a wider distribution of prices.

Finding 2

In using classifier models, Expedia users who search for hotel stays that include a Saturday night and more than four nights are a much better target variable to predict than the listed Expedia hotel prices. The linear regression model is not good at predicting hotel prices, which fits worse than a constant function that predicts the mean of the data at a -2.33 R2 score.

When comparing the classifiers' performance in predicting binary classes, a Random Forest Classification model had the highest accuracy score of 86.32%. Cross-validating between classifiers found that across ten trials, a random positive outcome (ability to determine a person's hotel stay includes a Saturday night based on the test set) was higher than an unexpected negative outcome more than 80% of the time. These classification methods allow for predicting discrete class labels, avoid the assumption of linearity between hotel prices and explanatory hotel variables, and reduce noise and overfitting.

Finding 3

To account for a clustering model, it carries out an unsupervised learning problem to further expand on the user segmentation - whether their hotel search includes or does not include a Saturday night stay and a stay over four nights. Define the number of clusters at two (one for hotel searches that has a binary class at 1 and another at 0). This clustering method adapts to single out certain features like hotel prices that distinguish the binary classification. With a silhouette score of 0.59, the data points are not compact within the cluster where it belongs to a person with either a hotel stay with or without a Saturday night stay. Two clusters of hotel searches are far from being a perfect cluster.