Predicting Available Listings for Seattle Airbnb

Melissa G Ngamini

Thinkful Data Science

June 24, 2018

Why do people choose Airbnb?

Some of the reasons why people choose to stay at an Airbnb rather than a hotel.

- **Convenient location**
  They give the customer the illusion of living like a local because of their location that range from along beaches, canals, Main Streets, main attractions, or in secluded area.
- **Flexibility**
  Hotels are strict about their check-in and check-out day and time. At most Airbnb those can be accommodated to fit the clients need.
- **Household amenities**
  Amenities that are included can be priceless, like the use of a full kitchen, a living room and extra bedrooms.
- **More space for less Money**
  Depending on the customer needs, they can decide to book a location that only offers a bedroom if they are traveling alone or they can choose to book a house or an apartment if they are with their family or just want more space.
- **One-on-one interaction with the owner**
  The owners are typically very helpful with recommendations, directions, and any information they feel may benefit you on your trip.

What do Airbnb hosts want?

Some of the informations an Airbnb host would get from this research.

- **Find the right pricing**
  Since they are not a hotel, they shouldn't price themselves as one. They will want to know the prices that attract the most customers depending on what they offer and their location.
- **Find the periods of high demand**
  There are periods in the year that have the most customers. They will want to know those times so that they can make sure to have their Listing available at the time to be booked as soon as possible.
- **Find the "in demand" amenities**
  What amenities attracts the most customers. For the type of listing they have on Airbnb, are the customers interested in having easy access to a bathroom, or the kitchen. The best numbers of guests can they allow their customer to have.
- **Find the customers expect from host**
  What are the elements of hosting that seem to attract customer. What they need to write in their summary/description and implement to make sure they receive good reviews so that their listing is not available throughout the year.

How to find an available Airbnb Listing?

1. **What is an available Listing?**

   A listing is said to be available when it can be booked by an Airbnb client.

2. **What do we need to to be able to predict an available Listing?**

   1. **Features Selection for Prediction**
      After cleaning and Analyzing the data choose features that can help with the prediction.
      - Using Random Forest Classifier
      - Using SelectKbest

   2. **Models used for Prediction**
      Choose 3 different modeling techniques to find which one gave us the better result.
      - Random Forest Classifier
      - Logistic Regression
      - Gradient Boosting

   3. **Prediction of Available Airbnb Listings**
      Of the best models which one gave us the better result?
      - Prediction using Random Forest Classifier
      - Prediction using Gradient Boosting

Reviews, Calendar and Listings data

```
The shape of the Reviews data is: (84849, 6)
```

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	7202016	38917982	2015-07-19	28943674	Bianca	Cute and cozy place. Perfect location to every...
1	7202016	39087409	2015-07-20	32440555	Frank	Kelly has a great room in a very central locat...
2	7202016	39820030	2015-07-26	37722850	Ian	Very spacious apartment, and in a great neighb...
3	7202016	40813543	2015-08-02	33671805	George	Close to Seattle Center and all it has to offe...
4	7202016	41986501	2015-08-10	34959538	Ming	Kelly was a great host and very accommodating ...

(a) Reviews Data

```
The shape of the Calendar data is: (1393570, 4)
```

	listing_id	date	available	price
0	241032	2016-01-04	t	$85.00
1	241032	2016-01-05	t	$85.00
2	241032	2016-01-06	f	NaN
3	241032	2016-01-07	f	NaN
4	241032	2016-01-08	f	NaN

(b) Calendar Data

```
The shape of the Listings data is: (3818, 92)
```

	id	listing_url	scrape_id	last_scraped	name	summary	space	description
0	241032	https://www.airbnb.com/rooms/241032	20160104002432	2016-01-04	Stylish Queen Anne Apartment	NaN	Make your self at home in this charming one-be...	Make your self at home in this charming one-be...
1	953595	https://www.airbnb.com/rooms/953595	20160104002432	2016-01-04	Bright & Airy Queen Anne Apartment	Chemically sensitive? We've removed the irrita...	Beautiful, hypoallergenic apartment in an extr...	Chemically sensitive? We've removed the irrita...

(c) Listings Data

Figure: Features in Reviews, Calendar and Listings data

Features in Listings data

```
'id',
'listing_url',
'scrape_id',
'last_scraped',
'name',
'summary',
'space',
'description',
'experiences_offered',
'neighborhood_overview',
'notes',
'transit',
'thumbnail_url',
'medium_url',
'picture_url',
'xl_picture_url',
'host_id',
'host_url',
'host_name',
'host_since',
'host_location',
'host_about',
'host_response_time',
'host_response_rate',
'host_acceptance_rate',
```

(a) Part 1

```
'host_is_superhost',
'host_thumbnail_url',
'host_picture_url',
'host_neighbourhood',
'host_listings_count',
'host_total_listings_count',
'host_verifications',
'host_has_profile_pic',
'host_identity_verified',
'street',
'neighbourhood',
'neighbourhood_cleansed',
'neighbourhood_group_cleansed',
'city',
'state',
'zipcode',
'market',
'smart_location',
'country_code',
'country',
'latitude',
'longitude',
'is_location_exact',
'property_type',
'room_type',
'accommodates',
```

(b) Part 2

```
'bathrooms',
'bedrooms',
'beds',
'bed_type',
'amenities',
'square_feet',
'price',
'weekly_price',
'monthly_price',
'security_deposit',
'cleaning_fee',
'guests_included',
'extra_people',
'minimum_nights',
'maximum_nights',
'calendar_updated',
'has_availability',
'availability_30',
'availability_60',
'availability_90',
'availability_365',
'calendar_last_scraped',
'number_of_reviews',
'first_review',
'last_review',
'review_scores_rating',
```

(c) Part 3

```
'review_scores_accuracy',
'review_scores_cleanliness',
'review_scores_checkin',
'review_scores_communication',
'review_scores_location',
'review_scores_value',
'requires_license',
'license',
'jurisdiction_names',
'instant_bookable',
'cancellation_policy',
'require_guest_profile_picture',
'require_guest_phone_verification',
'calculated_host_listings_count',
'reviews_per_month']
```

(d) Part 4

Figure: Features in Listings Data

Training set description

```
The shape of the Training set  is: (975499, 69)
```

	price_calendar	month	host_listings_count	accommodates	bathrooms	bedrooms	beds	guests_included	minimum_nights
7203765	110.0	6	1.0	2	1.0	0.0	1.0	1	3
6400000	160.0	5	3.0	4	1.0	1.0	1.0	0	1
6856295	85.0	8	1.0	2	1.0	1.0	1.0	2	1
9149612	69.0	12	1.0	2	1.0	0.0	1.0	2	2
9217337	52.0	7	1.0	2	1.5	1.0	1.0	1	1

(a) X Train

```
The shape of the Training set  is: (975499,)
7203765    0
6400000    1
6856295    1
9149612    0
9217337    1
Name: availability, dtype: int64
```

(b) Y Train

Figure: Features in the Training Set

Dealing with missing data

```
print(dfLnumeric.shape)
```

```
(3818, 33)
```

```
# Count nulls
null_count = dfLnumeric.isnull().sum()
null_count[null_count>0]
```

```
host_listings_count              2
bathrooms                       16
bedrooms                         6
beds                             1
square_feet                   3721
review_scores_rating           647
review_scores_accuracy         658
review_scores_cleanliness      653
review_scores_checkin          658
review_scores_communication    651
review_scores_location         655
review_scores_value            656
license                       3818
reviews_per_month              627
price                            1
dtype: int64
```

(a) Listings Data

```
print(df.shape)
```

```
(1393570, 80)
```

```
# Count nulls
null_count = df.isnull().sum()
null_count[null_count>0]
```

```
price_calendar                 459725
host_listings_count               730
bathrooms                        5840
bedrooms                         2190
beds                              365
review_scores_rating           236155
review_scores_accuracy         240170
review_scores_cleanliness      238345
review_scores_checkin          240170
review_scores_communication    237615
review_scores_location         239075
review_scores_value            239440
reviews_per_month              228855
price_listing                     365
dtype: int64
```

(b) Join of Listings and Calendar data

Figure: Missing Data

Features kept from Analysis of correlation matrix

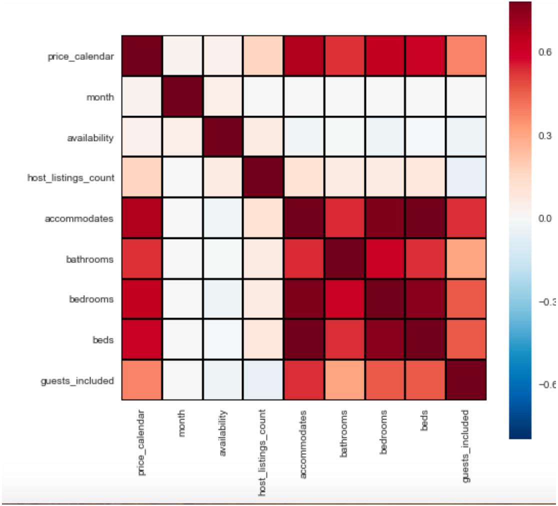
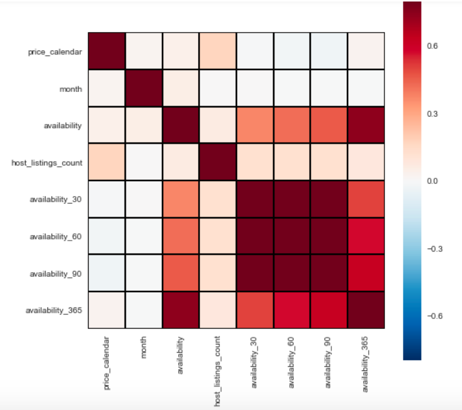
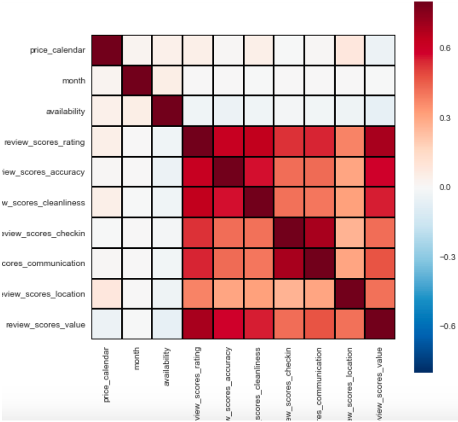


Figure: Features that were kept despite being highly correlated

Features removed from Analysis of correlation matrix



(a) Features correlated to availability



(b) Features correlated to review scores

Figure: Features removed for being highly correlated

Relationship between month and availability

```
### Date at which the Listings data was last scraped.
dfL['last_scraped'][0]
```

```
'2016-01-04'
```

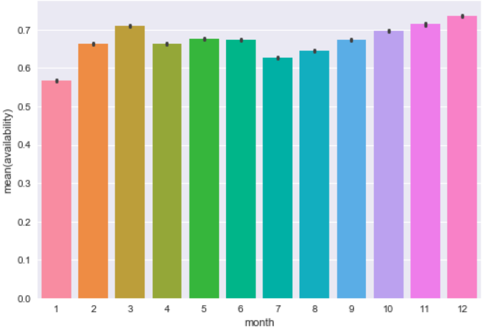
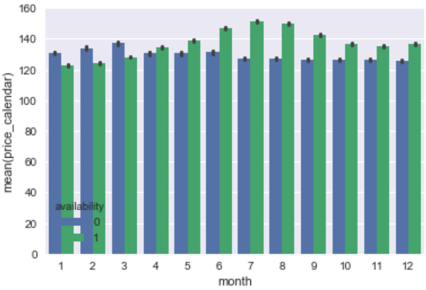
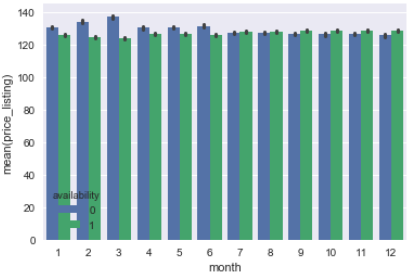


Figure: Distribution of availability per month

Relationship between price calendar and month vs price listing and month



(a) Distribution of calendar price per month



(b) Distribution of listing price per month

Figure: Available and Non Available Prices per month

Relationship between reviews per month and number of reviews

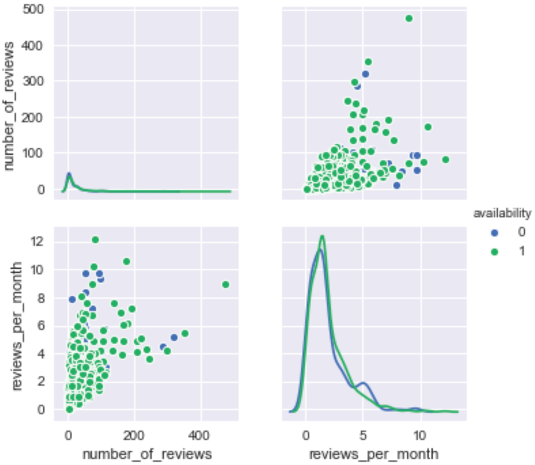


Figure: Number of reviews vs Reviews per month

Occurrence of some features in Data

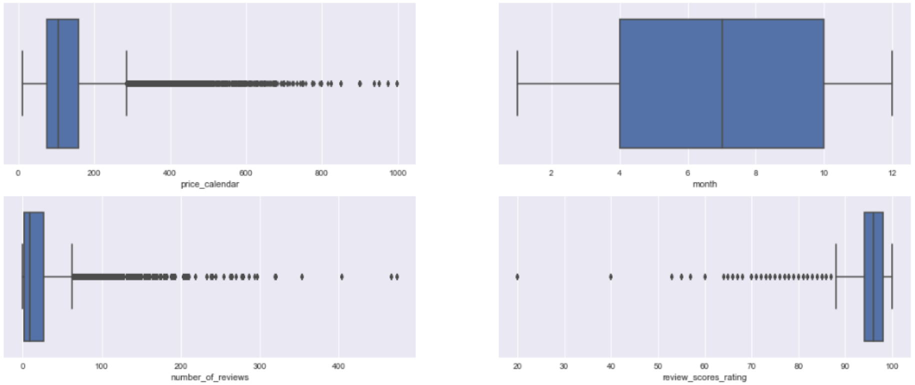


Figure: Histograms of some features in data

Occurrence of keywords selected from Summary in Data

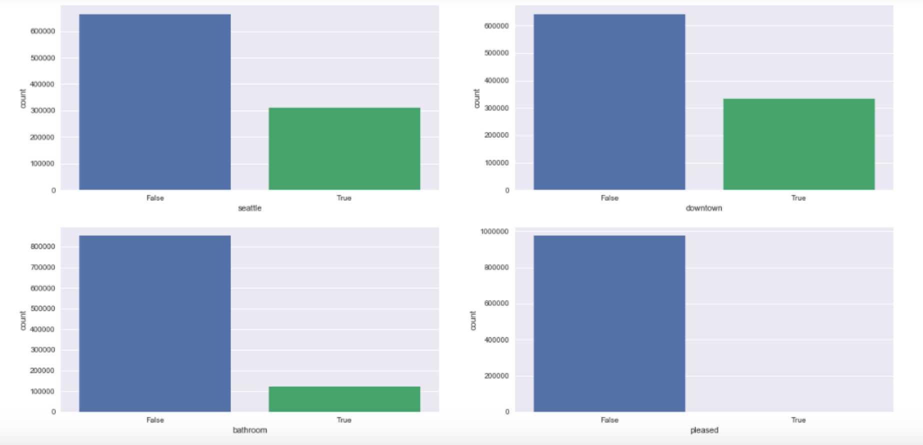


Figure: Count of keywords from Summary

Random Forest Classifier

```
### Random Forest Classifier
rfc = ensemble.RandomForestClassifier(max_depth=10,max_features='auto', n_estimators=40)
start_time = time.clock()
rfc.fit(X_train, Y_train)
print('Runtime for Random Forest: '+'%s seconds'% (time.clock() - start_time)) # End time for execution speed.

df_features = pd.DataFrame(rfc.feature_importances_)
df_features['features']= X_train.columns
df_features.sort_values(0,ascending = False)
best_Rfeatures = list(df_features.sort_values(0,ascending = False)['features'][:30])

print('\nScore on  Training Set: ' + str(rfc.score(X_train, Y_train)))
print('\nScore on  Test Set: ' + str(rfc.score(X_test, Y_test)))
print('\nThe numbers of important features we will use are: ', len(best_Rfeatures))
print('\nUsing Random Forest Classifier the 30 best features are: ')
print(best_Rfeatures)
```

Figure: Random Forest Classifier to select important features

1. **Runtime for Random Forest:** 145.60643800000003 seconds
2. **Score on Training Set:** 0.732044830389
3. **Score on Test Set:** 0.731796752226
4. **The numbers of important features we will use are:** 30

SelectKBest

```
## Select Kbest Features
selector = SelectKBest(f_classif, k=30)
start_time = time.clock()
selector.fit_transform(X_train, Y_train)
print('\nRuntime for SelectKbest: '+'%s seconds'% (time.clock() - start_time)) # End time for execution speed.
names = X_train.columns.values[selector.get_support()]
scores = selector.scores_[selector.get_support()]
names_scores = list(zip(names, scores))
df_features = pd.DataFrame(data = names_scores, columns=['Feat_names', 'F_Scores'])
best_Kfeatures = list(df_features.sort_values(['F_Scores', 'Feat_names'], ascending = [False, True])['Feat_names'])
print('\nThe numbers of important features we will use are: ', len(best_Kfeatures))
print('\nUsing SelectKbest the 30 best features are: ')

print(best_Kfeatures)
```

Figure: SelectKBest to select important features

1. **Runtime for SelectKBest:** 18.22121599999997 seconds
2. **The numbers of important features we will use are:** 30

30 features selected with Random Forest and SelectKBest

```
Using Random Forest Classifier the 30 best features are:
['price_calendar', 'month', 'host_listings_count', 'number_of_reviews', 'maximum_nights', 'calculated_hos
t_listings_count', 'reviews_per_month', 'review_scores_rating', 'bedrooms', 'accommodates', 'minimum_nigh
ts', 'guests_included', 'review_scores_value', 'beds', 'bathrooms', 'review_scores_location', 'host_is_su
perhost', 'require_guest_profile_picture', 'instant_bookable', 'require_guest_phone_verification', 'coz
y', 'bathroom', 'seattle', 'bedroom', 'downtown', 'parking', 'light', 'view', 'bed', 'easy']
```

Figure: Important features using Random Forest Classifier

```
Using SelectKbest the 30 best features are:
['calculated_host_listings_count', 'number_of_reviews', 'host_listings_count', 'review_scores_value', 'mo
nth', 'price_calendar', 'review_scores_location', 'bedrooms', 'guests_included', 'review_scores_rating',
'require_guest_phone_verification', 'accommodates', 'host_has_profile_pic', 'require_guest_profile_pictur
e', 'host_is_superhost', 'spacious', 'bedroom', 'bonus', 'bathroom', 'newly', 'guest', 'groceries', 'furn
ished', 'reviews_per_month', 'beds', 'view', 'restaurant', ' kitchen', 'tea', 'seattle']
```

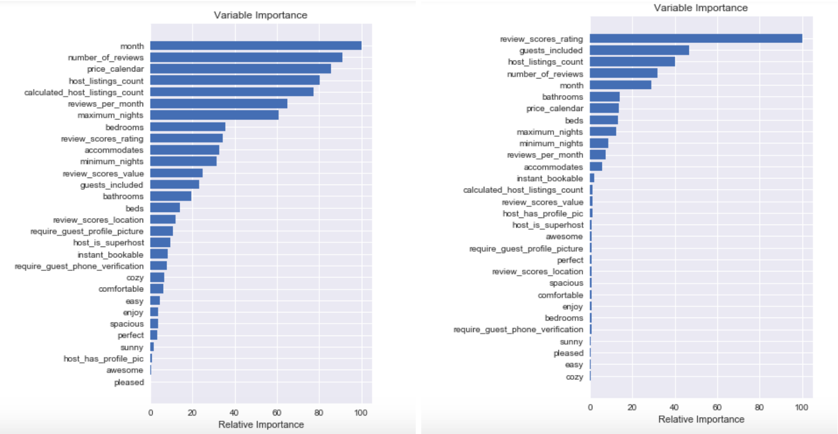
Figure: Important features using SelectKBest

```
The numbers of important features belonging to both are :  20

These features are:
['price_calendar', 'bedrooms', 'review_scores_value', 'number_of_reviews', 'host_is_superhost', 'bathroo
m', 'beds', 'view', 'guests_included', 'review_scores_rating', 'accommodates', 'seattle', 'reviews_per_mo
nth', 'month', 'review_scores_location', 'require_guest_profile_picture', 'host_listings_count', 'bedroo
m', 'calculated_host_listings_count', 'require_guest_phone_verification']
```

Figure: Important features belonging to both

30 features selected with Random Forest and SelectKBest



(a) Using RFC

(b) Using Select KBest

Figure: Variable Importance

R-Squared score using features from the Random Forest Classifier

1. **Random Forest Classifier:**

   hyper parameters: maxdepth=10, maxfeatures='auto', nestimators=40

   - Runtime for Random Forest with RFeatures: 146.56320900000003 seconds
   - Score on Training Set: 0.747988465391
   - Score on Test Set: 0.747870577007
   - Cross validation results: 74.904% $\pm$ 0.105%

2. **Logistic Regression:**

   penalty: 'l2'

   - Runtime For Logistic Regression with RFeatures: 39.91339199999993 seconds
   - Score on Training set: 0.670319498021
   - Score on Test set: 0.670785584267
   - Cross validation results: 67.026% $\pm$ 0.019%

3. **Gradient Boosting:**

   hyper parameters: 50 iterations, 5-deep trees, loss function = deviance

   - Runtime for Gradient Boosting with RFeatures: 386.33381800000006 seconds
   - Score on Training Set: 0.733607107747
   - Score on Test Set: 0.734272408275
   - Cross validation results: 73.465% $\pm$ 0.152%

Results using features from the Random Forest Classifier

	Predict False	Predict True
Actual False	82930	238498
Actual True	7339	646732

```
False Positives (Type I error): 238498 (74.2%)
False Negatives (Type II error): 7339 (1.1%)
```

(a) Random Forest Classifier

	Predict False	Predict True
Actual False	6907	314521
Actual True	7082	646989

```
False Positives (Type I error): 314521 (97.9%)
False Negatives (Type II error): 7082 (1.1%)
```

(b) Logistic Regression

	Predict False	Predict True
Actual False	88253	233175
Actual True	26691	627380

```
False Positives (Type I error): 233175 (72.5%)
False Negatives (Type II error): 26691 (4.1%)
```

(c) Gradient Boosting

Figure: Results for models using RFC Features

R-Squared score using features from the SelectKBest

1. **Random Forest Classifier:**

   hyper parameters: maxdepth=10, maxfeatures='auto', nestimators=40

   - Runtime for Random Forest with KFeatures: 114.39116799999988 seconds
   - Score on Training Set: 0.725604024197
   - Score on Test Set: 0.725977166558
   - Cross validation results: 72.633% ± 0.219%

2. **Logistic Regression:**

   penalty: 'l2'

   - Runtime For Logistic Regression with KFeatures: 33.37867000000006 seconds
   - Score on Training set: 0.671174445079
   - Score on Test set: 0.671718440169
   - Cross validation results: 67.105% ± 0.014%

3. **Gradient Boosting:**

   hyper parameters: 50 iterations, 5-deep trees, loss function = deviance

   - Runtime for Gradient Boosting with KFeatures: 368.2461929999995 seconds
   - Score on Training Set: 0.724877216686
   - Score on Test Set: 0.725450940151
   - Cross validation results: 72.275% ± 0.115%

Results using features from the SelectKBest

	Predict False	Predict True
Actual False	6907	314521
Actual True	7082	646989

```
False Positives (Type I error): 314521 (97.9%)
False Negatives (Type II error): 7082 (1.1%)
```

(a) Random Forest Classifier

	Predict False	Predict True
Actual False	8816	312612
Actual True	8157	645914

```
False Positives (Type I error): 312612 (97.3%)
False Negatives (Type II error): 8157 (1.2%)
```

(b) Logistic Regression

	Predict False	Predict True
Actual False	74748	246680
Actual True	21702	632369

```
False Positives (Type I error): 246680 (76.7%)
False Negatives (Type II error): 21702 (3.3%)
```

(c) Gradient Boosting

Figure: Results for models using KBest Features

R-Squared score updated Gradient Boosting hyper parameters and RFC Features

1. **Gradient Boosting:**

   hyper parameters: 50 iterations, 10-deep trees, loss function = deviance

   - Runtime for Gradient Boosting with RFeatures: 4092.5195510000003 seconds
   - Score on Training Set: 0.904520660708
   - Score on Test Set: 0.903169557324
   - Cross validation results: 90.239% $\pm$ 0.154%

2. **Gradient Boosting:**

   hyper parameters: 100 iterations, 10-deep trees, loss function = deviance

   - Runtime for Gradient Boosting with RFeatures: 7149.300540999997 seconds
   - Score on Training Set: 0.942255194521
   - Score on Test Set: 0.940998060138
   - Cross validation results: 94.277% $\pm$ 0.086%

Results using updated Gradient Boosting hyper parameters and RFC Features

	Predict False	Predict True
Actual False	243227	78201
Actual True	14939	639132

```
False Positives (Type I error): 78201 (24.3%)
False Negatives (Type II error): 14939 (2.3%)
```

(a) 50 iterations, 10-deep trees, loss function = deviance

	Predict False	Predict True
Actual False	277097	44331
Actual True	11999	642072

```
False Positives (Type I error): 44331 (13.8%)
False Negatives (Type II error): 11999 (1.8%)
```

(b) 100 iterations, 10-deep trees loss function = deviance

Figure: Results for models using Gradient Boosting and RFC Features

Prediction Probabilities

- **Random Forest Classifier with KFeatures:**

  hyper parameters: maxdepth=10, maxfeatures='auto', nestimators=40

- **Gradient Boosting with RFeatures:**

  hyper parameters: 100 iterations, 10-deep trees, loss function = deviance

```
ypred_proRFC = rfc.predict_proba(X_testR)
print(ypred_proRFC)
```

```
[[ 0.41552805  0.58447195]
 [ 0.42184976  0.57815024]
 [ 0.31470597  0.68529403]
 ...,
 [ 0.31339018  0.68660982]
 [ 0.44072339  0.55927661]
 [ 0.42121858  0.57878142]]
```

(a) Prediction using RFC

```
ypred_proCLF = clf.predict_proba(X_testR)
print(ypred_proCLF)
```

```
[[ 0.12720408  0.87279592]
 [ 0.86886334  0.13113666]
 [ 0.8554316   0.1445684 ]
 ...,
 [ 0.0601768   0.9398232 ]
 [ 0.27904007  0.72095993]
 [ 0.08502172  0.91497828]]
```

(b) Prediction using GB

Figure: Predictions between using RFC vs GB

Conclusion and Future Work

1. Try to get the Type I and Type II errors to be about the same. Meaning both around 10%.
2. Try to include Reviews Data.
3. Create New Features with some of the existing and deleted features.
4. Using PCA instead of SelectKBest and Random Forest Classifier.
5. Investigate how the Median change our data.
6. Investigate how using price calendar vs price listing may impact the prediction.
7. Rerun the 3 models using features selected by RFC and SelectKBest and see how they impact performance and the confusion matrix.

That's all folks!

Questions?