Predict which 'hotel_cluster' a user will book given the information in his (or her) search.

In [72]:
```
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn import svm
```

In [96]:
```
#randomly sample 1% of the records in order to decrease loading time
df = pd.read_csv('data/train.csv', sep=',').dropna()
dest = pd.read_csv('data/destinations.csv')
df = df.sample(frac=0.01, random_state=99)
df.shape
```

Out[96]:
```
(241179, 24)
```

In [97]:
```
# Output for use in R
df.to_csv('data/train_sample.csv')
```

EDA

I chose to export my sample dataset to R for EDA purposes.

Results of the ETA:

- The data consists of the string variables 'date_time', 'srch_ci', and 'srch_co' which will not be usable by some machine learning algorithms so I'll need to break them apart if I want to use them.
- From the histogram of 'hotel_cluster', it appears the data is fairly evenly distributed across all 100 clusters, but there is some skew to the data on the right-hand side.

Feature Engineering

I want to break up the date_time column into 'year' and 'month' as I believe they could be good predictors of 'hotel_cluster'. This will result in

In [75]:
```
from datetime import datetime
def get_year(x):
    if x is not None and type(x) is not float:
        try:
            return datetime.strptime(x, '%Y-%m-%d').year
        except ValueError:
            return datetime.strptime(x, '%Y-%m-%d %H:%M:%S').year
    else:
        return 2013
    pass
def get_month(x):
    if x is not None and type(x) is not float:
        try:
            return datetime.strptime(x, '%Y-%m-%d').month
        except:
            return datetime.strptime(x, '%Y-%m-%d %H:%M:%S').month
    else:
        return 1
    pass

def left_merge_dataset(left_dframe, right_dframe, merge_column):
    return pd.merge(left_dframe, right_dframe, on=merge_column, how='left')
```

Dealing with date_time column
This meant removing

In [76]:
```
df['date_time_year'] = pd.Series(df.date_time, index = df.index)
df['date_time_month'] = pd.Series(df.date_time, index = df.index)

from datetime import datetime
df.date_time_year = df.date_time_year.apply(lambda x: get_year(x))
df.date_time_month = df.date_time_month.apply(lambda x: get_month(x))

# remove the 'date_time' column
del df['date_time']
```

Dealing with srch_ci (Checkin date) column:

In [77]:
```
df['srch_ci_year'] = pd.Series(df.srch_ci, index=df.index)
df['srch_ci_month'] = pd.Series(df.srch_ci, index=df.index)

# convert year & months to int
df.srch_ci_year = df.srch_ci_year.apply(lambda x: get_year(x))
df.srch_ci_month = df.srch_ci_month.apply(lambda x: get_month(x))

# remove the srch_ci column
del df['srch_ci']
```

Dealing with srch_co (Checkout date) column:

In [78]:
```
df['srch_co_year'] = pd.Series(df.srch_co, index=df.index)
df['srch_co_month'] = pd.Series(df.srch_co, index=df.index)

# convert year & months to int
df.srch_co_year = df.srch_co_year.apply(lambda x: get_year(x))
df.srch_co_month = df.srch_co_month.apply(lambda x: get_month(x))

# remove the srch_co column
del df['srch_co']
```

Preliminary Analysis

After creating the new features, I want to know if anything correlates well with 'hotel_cluster'. This is how I will identify if any variables in particular happen to more easily predict a 'hotel_cluster'. Using 'corr()' I am able to find the pairwise correlation of all columns in the datagrame. I don't have to worry about 'NA' values as those are excluded automatically.

In [79]:
```
df.corr()["hotel_cluster"].sort_values()
```

Out[79]:
```
srch_destination_type_id    -0.036120
site_name                   -0.027497
hotel_country               -0.023837
is_booking                  -0.022898
user_location_country       -0.020239
srch_destination_id         -0.016736
srch_co_month               -0.005874
srch_rm_cnt                 -0.005570
srch_ci_month               -0.005015
date_time_month             -0.002142
channel                     -0.001386
date_time_year              -0.000435
cnt                          0.000378
hotel_continent              0.000422
user_location_city           0.001241
user_id                      0.003891
orig_destination_distance    0.006084
user_location_region         0.006927
srch_ci_year                 0.008562
is_mobile                    0.008788
srch_co_year                 0.009287
posa_continent               0.012180
srch_adults_cnt              0.012407
srch_children_cnt            0.014901
hotel_market                 0.022149
is_package                   0.047598
hotel_cluster                1.000000
Name: hotel_cluster, dtype: float64
```

The values are all very low (except for hotel_cluster which is itself) thus I conclude that there is no strong, positive, linear correlation of any particular variable with 'hotel_cluster'.

Strategy

I anticipate that 'hotel_country' and 'hotel_market' will be useful in determining 'hotel_cluster'. After all, the user will likely be shown a hotel in their destination country and market.

In [80]:
```
pieces = [df.groupby(['srch_destination_id','hotel_country','hotel_market','hotel_cluster'])['is_booking'].agg(['sum','count'])]
agg = pd.concat(pieces).groupby(level=[0,1,2,3]).sum()
agg.dropna(inplace=True)
agg.head()
```

Out[80]:

				sum	count
srch_destination_id	**hotel_country**	**hotel_market**	**hotel_cluster**		
4	**7**	**246**	**22**	0	1
			29	0	1
			30	0	1
			32	1	2
			43	0	1

In [81]:
```
agg['sum_and_cnt'] = 0.85*agg['sum'] + 0.15*agg['count']
agg = agg.groupby(level=[0,1,2]).apply(lambda x: x.astype(float)/x.sum())
agg.reset_index(inplace=True)
agg.head()
```

Out[81]:

	srch_destination_id	hotel_country	hotel_market	hotel_cluster	sum	count	sum_and_cnt
0	4	7	246	22	0.0	0.125	0.073171
1	4	7	246	29	0.0	0.125	0.073171
2	4	7	246	30	0.0	0.125	0.073171
3	4	7	246	32	1.0	0.250	0.560976
4	4	7	246	43	0.0	0.125	0.073171

In [82]:
```
agg_pivot = agg.pivot_table(index=['srch_destination_id','hotel_country','hote
l_market'], columns='hotel_cluster', values='sum_and_cnt').reset_index()
agg_pivot.head()
```

Out[82]:

hotel_cluster	srch_destination_id	hotel_country	hotel_market	0	1	2	3	4	5
0	4	7	246	NaN	NaN	NaN	NaN	NaN	NaN
1	8	50	416	NaN	NaN	NaN	NaN	NaN	NaN
2	11	50	824	NaN	NaN	NaN	NaN	NaN	NaN
3	14	27	1434	NaN	NaN	NaN	NaN	NaN	NaN
4	16	50	419	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 103 columns

Merge with the destination table and our newly created aggregate pivot table.

In [83]:
```
df = pd.merge(df, dest, how='left', on='srch_destination_id')
df = pd.merge(df, agg_pivot, how='left', on=['srch_destination_id','hotel_coun
try','hotel_market'])
df.fillna(0, inplace=True)
df.shape
```

Out[83]: (241179, 276)

In [84]:
```
df.shape
```

Out[84]: (241179, 276)

Implementing Algorithms

In [85]:
```
# Only interested in booking events, not clicks
df = df.loc[df['is_booking'] == 1]
```

In [86]:
```
# Get features and labels
X = df.drop(['user_id', 'hotel_cluster', 'is_booking'], axis=1)
y = df.hotel_cluster
```

Naive Bayes

In [87]:
```
from sklearn.naive_bayes import GaussianNB

clf = make_pipeline(preprocessing.StandardScaler(), GaussianNB(priors=None))
np.mean(cross_val_score(clf, X, y, cv=10))
```

Out[87]: 0.10347912437041926

K-Nearest Neighbors Classifier

I tried using 3, 5, 7, and 9 nearest neighbors to see which one would give me the best accuracy. The best was nearest neighbors.

In [95]:
```
from sklearn.neighbors import KNeighborsClassifier
```

n = 3

In [92]:
```
clf = make_pipeline(preprocessing.StandardScaler(), KNeighborsClassifier(n_nei
ghbors=3))
np.mean(cross_val_score(clf, X, y, cv=10, scoring='accuracy'))
```

Out[92]: 0.2339068655585447

n = 5

In [88]:
```
clf = make_pipeline(preprocessing.StandardScaler(), KNeighborsClassifier(n_nei
ghbors=5))
np.mean(cross_val_score(clf, X, y, cv=10, scoring='accuracy'))
```

Out[88]: 0.25631461834732266

n = 7

In [93]:
```
clf = make_pipeline(preprocessing.StandardScaler(), KNeighborsClassifier(n_nei
ghbors=7))
np.mean(cross_val_score(clf, X, y, cv=10, scoring='accuracy'))
```

Out[93]: 0.27103574280950904

n = 9

In [94]:
```
clf = make_pipeline(preprocessing.StandardScaler(), KNeighborsClassifier(n_nei
ghbors=9))
np.mean(cross_val_score(clf, X, y, cv=10, scoring='accuracy'))
```

Out[94]: 0.275165163629703

Random Forest Classifier

In [90]:
```
clf = make_pipeline(preprocessing.StandardScaler(), RandomForestClassifier(n_e
stimators=273,max_depth=10,random_state=0))
np.mean(cross_val_score(clf, X, y, cv=10))
```

Out[90]: 0.24865023372782996

SVM Classifier

In [91]:
```
from sklearn import svm

clf = make_pipeline(preprocessing.StandardScaler(), svm.SVC(decision_function_
shape='ovo'))
np.mean(cross_val_score(clf, X, y, cv=10))
```

Out[91]:
```
0.3228727137315005
```