In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
%matplotlib inline
```

In [3]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
```

In [4]:

```
import datetime
```

In [59]:

```
##Importing data
train = pd.read_csv('train.csv', nrows = 100000).dropna()
train = train.sample(frac=0.01, random_state=99)
train.head()
```

Out[59]:

	date_time	site_name	posa_continent	user_location_country	user_location_region	user_
54146	2013-02-27 11:09:47	2	3	66	288	
73939	2014-04-14 18:36:32	2	3	66	442	
41988	2014-12-25 22:02:39	2	3	66	462	
29834	2013-12-16 19:47:34	2	3	66	442	
67723	2014-04-17 18:53:37	34	3	205	354	

5 rows × 24 columns

In [6]:

```
destinations = pd.read_csv('destinations.csv')
destinations.head()
```

Out[6]:

	srch_destination_id	d1	d2	d3	d4	d5	d6	
0	0	-2.198657	-2.198657	-2.198657	-2.198657	-2.198657	-1.897627	-2.19
1	1	-2.181690	-2.181690	-2.181690	-2.082564	-2.181690	-2.165028	-2.18
2	2	-2.183490	-2.224164	-2.224164	-2.189562	-2.105819	-2.075407	-2.22
3	3	-2.177409	-2.177409	-2.177409	-2.177409	-2.177409	-2.115485	-2.17
4	4	-2.189562	-2.187783	-2.194008	-2.171153	-2.152303	-2.056618	-2.19

5 rows × 150 columns

In [8]:

```
test = pd.read_csv('test.csv', nrows=100000)
test
```

Out[8]:

	id	date_time	site_name	posa_continent	user_location_country	user_location_regic
0	0	2015-09-03 17:09:54	2	3	66	17
1	1	2015-09-24 17:38:35	2	3	66	17
2	2	2015-06-07 15:53:02	2	3	66	14
3	3	2015-09-14 14:49:10	2	3	66	25
4	4	2015-07-17 09:32:04	2	3	66	46
...	...	...	...	...	...	
99995	99995	2015-07-15 05:16:04	2	3	66	44
99996	99996	2015-09-08 00:31:42	2	3	66	44
99997	99997	2015-09-15 02:20:39	2	3	0	27
99998	99998	2015-09-22 21:36:29	2	3	66	44
99999	99999	2015-05-11 09:32:58	2	3	66	18

100000 rows × 22 columns

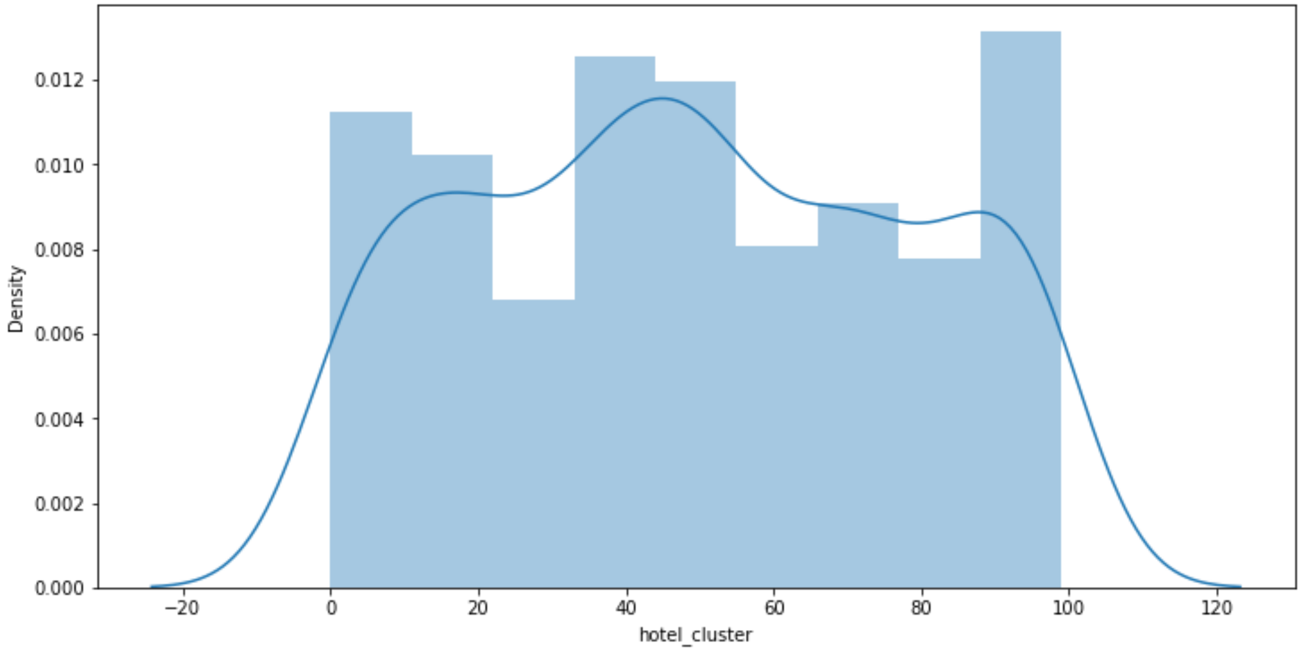
In [10]:

```
plt.figure(figsize=(12, 6))
sns.distplot(train['hotel_cluster'])
##Creating histogram of hotel clusters range
```

```
/Users/aarondrexler/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

```
<AxesSubplot:xlabel='hotel_cluster', ylabel='Density'>
```



In [11]:

```python
from datetime import datetime
def year(i):
    if i is not None and type(i) is not float:
        try:
            return datetime.strptime(i, '%Y-%m-%d').year
        except ValueError:
            return datetime.strptime(i, '%Y-%m-%d %H:%M:%S').year
    else:
        return 2013
    pass
##Gets year of the date
```

In [12]:

```python
def month(i):
    if i is not None and type(i) is not float:
        try:
            return datetime.strptime(i, '%Y-%m-%d').month
        except:
            return datetime.strptime(i, '%Y-%m-%d %H:%M:%S').month
    else:
        return 1
    pass
##Gets month of the date
```

In [13]:

```
##Gets year and month from date time in train df
train['date_time_year'] = pd.Series(train.date_time, index = train.index)
train['date_time_month'] = pd.Series(train.date_time, index = train.index)
```

In [15]:

```
train.date_time_year = train.date_time_year.apply(lambda i: year(i))
train.date_time_month = train.date_time_month.apply(lambda i: month(i))
del train['date_time']
```

In [16]:

```
##Gets year and month from check in in train df
train['srch_ci_year'] = pd.Series(train.srch_ci, index = train.index)
train['srch_ci_month'] = pd.Series(train.srch_ci, index = train.index)
```

In [18]:

```
train.srch_ci_year = train.srch_ci_year.apply(lambda i: year(i))
train.srch_ci_month = train.srch_ci_month.apply(lambda i: month(i))
del train['srch_ci']
```

In [19]:

```
##Gets year and month from check out in train df
train['srch_co_year'] = pd.Series(train.srch_co, index = train.index)
train['srch_co_month'] = pd.Series(train.srch_co, index = train.index)
```

In [20]:

```
train.srch_co_year = train.srch_co_year.apply(lambda i: year(i))
train.srch_co_month = train.srch_co_month.apply(lambda i: month(i))
del train['srch_co']
```

In [21]:

```
train
```

Out[21]:

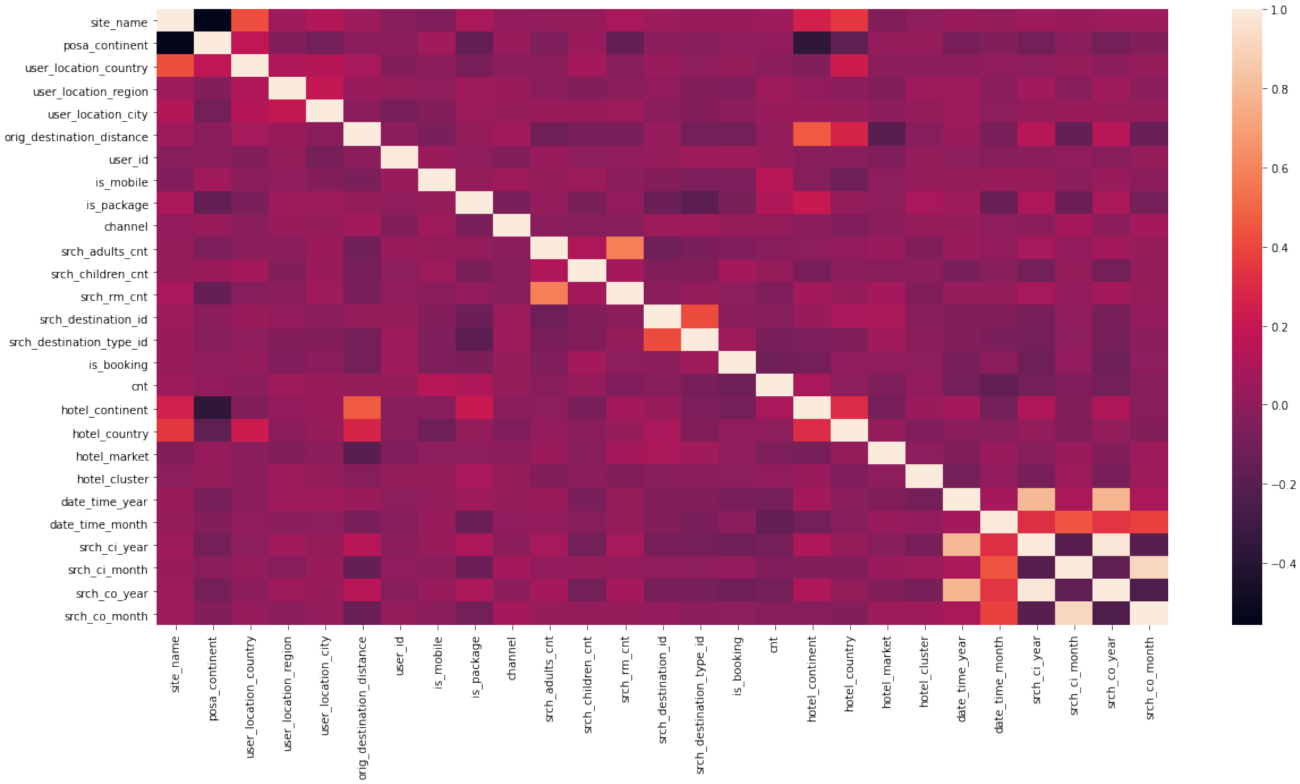
	site_name	posa_continent	user_location_country	user_location_region	user_location_cit
54146	2	3	66	288	1920
73939	2	3	66	442	2862
41988	2	3	66	462	1148
29834	2	3	66	442	4916
67723	34	3	205	354	4149
...	...	...	...	...	
27897	13	1	46	172	5472
92954	2	3	66	333	2319
20135	2	3	66	220	2088
10848	2	3	66	314	4866
40849	2	3	66	258	4688

630 rows × 27 columns

In [23]:

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 10)
sns.heatmap(train.corr())
##Correlation heat map
```

```
Out[23]: <AxesSubplot:>
```



```
In [24]: train.corr()["hotel_cluster"].sort_values()
         ##Sorted correlation values in comparison to hotel cluster
```

Out[24]:

```
date_time_year               -0.091839
srch_ci_year                 -0.082198
srch_co_year                 -0.080500
hotel_country                -0.056714
srch_rm_cnt                  -0.049394
srch_adults_cnt              -0.046789
orig_destination_distance    -0.029632
srch_destination_id          -0.025882
user_location_country        -0.008373
srch_children_cnt            -0.006115
srch_destination_type_id     -0.004883
hotel_market                 -0.003073
is_booking                   -0.001563
site_name                     0.000731
cnt                           0.010512
date_time_month               0.014342
user_id                       0.015190
posa_continent                0.021725
is_mobile                     0.022681
channel                       0.023892
user_location_city            0.024813
hotel_continent               0.040043
srch_ci_month                 0.050803
srch_co_month                 0.054467
user_location_region          0.056181
is_package                    0.091786
hotel_cluster                 1.000000
Name: hotel_cluster, dtype: float64
```

In [40]:

```
group = [train.groupby(['srch_destination_id','hotel_country','hotel_market',
a = pd.concat(group).groupby(level=[0,1,2,3]).sum()
a.dropna(inplace=True)
```

In [41]:

```
a['sum_and_cnt'] = 0.80*a['sum'] + 0.20*a['count']
a = a.groupby(level=[0,1,2]).apply(lambda i: i.astype(float)/i.sum())
a.reset_index(inplace=True)
```

In [42]:

```
pivot = a.pivot_table(index=['srch_destination_id','hotel_country','hotel_mar
pivot.head(10)
## Creates group by in order to organize, sort, prioritize data to creat pivo
```

Out[42]:

hotel_cluster	srch_destination_id	hotel_country	hotel_market	2	7	10	15	16	18
0	486	50	726	NaN	NaN	NaN	NaN	1.0	NaN
1	3628	50	689	NaN	NaN	NaN	NaN	NaN	NaN
2	3637	50	366	NaN	NaN	NaN	NaN	NaN	NaN
3	3744	50	1630	NaN	NaN	NaN	NaN	NaN	NaN
4	3754	50	350	NaN	NaN	NaN	NaN	NaN	NaN
5	3935	50	661	NaN	NaN	NaN	NaN	NaN	NaN
6	4348	50	1101	NaN	1.0	NaN	NaN	NaN	NaN
7	5405	8	126	NaN	NaN	NaN	NaN	NaN	NaN
8	5736	50	365	NaN	NaN	NaN	NaN	NaN	NaN
9	8239	50	407	NaN	NaN	NaN	NaN	NaN	NaN

10 rows × 37 columns

In [44]:

```
train = pd.merge(train, destinations, on='srch_destination_id')
train = pd.merge(train, pivot, on=['srch_destination_id','hotel_country','hot
train.fillna(0, inplace=True)
##Merge destinations and pivot tables
```

In [45]:

```
train = train.loc[train['is_booking'] == 1]
##Only want to include events that are for booking
```

In [50]:

```
X = train.drop(['user_id', 'hotel_cluster', 'is_booking'], axis=1)
y = train.hotel_cluster
X.shape, y.shape
##Pulls and creates x and y to be used in models
```

Out[50]:

```
((54, 390), (54,))
```

In [56]:

```
from sklearn.linear_model import LogisticRegression
classifier = make_pipeline(preprocessing.StandardScaler(), LogisticRegression
np.mean(cross_val_score(classifier, X, y))
##Logistic Regression of data
```

```
/Users/aarondrexler/opt/anaconda3/lib/python3.8/site-packages/sklearn/model_se
lection/_split.py:676: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=5.
  warnings.warn(
```

Out[56]:

```
0.3327272727272727
```

In [57]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = make_pipeline(preprocessing.StandardScaler(), KNeighborsClassifi
np.mean(cross_val_score(classifier, X, y, scoring='accuracy'))
##K-nearest neighbors of data
```

```
/Users/aarondrexler/opt/anaconda3/lib/python3.8/site-packages/sklearn/model_se
lection/_split.py:676: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=5.
  warnings.warn(
```

Out[57]:

```
0.14909090909090908
```

In [ ]: