Individual Final project_wangxiao19451539

May 8, 2020

1 Final Project-Competition from Kaggle

Description of the Challenges of Expedia Planning your dream vacation, or even a weekend escape, can be an overwhelming affair. With hundreds, even thousands, of hotels to choose from at every destination, it's difficult to know which will suit your personal preferences. Should you go with an old standby with those pillow mints you like, or risk a new hotel with a trendy pool bar?

Expedia wants to take the proverbial rabbit hole out of hotel search by providing personalized hotel recommendations to their users. This is no small task for a site with hundreds of millions of visitors every month! Currently, Expedia uses search parameters to adjust their hotel recommendations, but there aren't enough customer specific data to personalize them for each user. In this competition, Expedia is challenging Kagglers to contextualize customer data and predict the likelihood a user will stay at 100 different hotel groups. The data in this competition is a random selection from Expedia and is not representative of the overall statistics.

https://www.kaggle.com/c/expedia-hotel-recommendations

Name: Wang Xiao 19451539

[1]:
```
#numpy,matplotlib,pandas and seaborn
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import datetime
import statistics
import scipy
```

1.1 Read Data

[2]:
```
time_col = ['date_time', 'srch_ci', 'srch_co']
train_set = pd.read_csv("train_hotel.csv",parse_dates=time_col)
test_set = pd.read_csv("test_hotel.csv",parse_dates=time_col)
destinations_df = pd.read_csv('destinations.csv')
```

1.2 Exploratory Data Analysis

[3]:
```
train_set.shape, test_set.shape, destinations_df.shape
```

[3]:
```
((753406, 24), (75341, 24), (62106, 150))
```

[4]:
```
train_set.head()
```

[4]:
```
             date_time  site_name  posa_continent  user_location_country  \
0  2013-10-29 09:25:27         34               3                    205
1  2014-07-06 00:17:02          2               3                     66
2  2014-07-12 19:02:33         11               3                    205
3  2014-03-12 08:32:59          2               3                     66
4  2013-11-03 22:15:17          2               3                    231

   user_location_region  user_location_city  orig_destination_distance  \
0                   354                1666                        NaN
1                   174                8124                  5538.8566
2                   343               37594                   346.1719
3                   435               40631                     4.8720
4                    70               11644                        NaN

   user_id  is_mobile  is_package  …  srch_children_cnt srch_rm_cnt  \
0   313095          0           0  …                  0           1
1   628718          0           1  …                  1           1
2  1064708          0           0  …                  0           1
3   285636          0           0  …                  0           1
4   183708          0           0  …                  0           2

  srch_destination_id  srch_destination_type_id  is_booking  cnt  \
0               14875                         1           0    1
1                8747                         1           1    1
2               25544                         6           0    1
3               12364                         5           0    1
4                1833                         6           1    1

   hotel_continent  hotel_country  hotel_market  hotel_cluster
0                2            198           750             98
1                3            106           107             25
2                2             50          1094             35
3                2             50           647             39
4                3             99          1225             82

[5 rows x 24 columns]
```

[5]:
```
train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 753406 entries, 0 to 753405
Data columns (total 24 columns):
date_time                    753406 non-null datetime64[ns]
site_name                    753406 non-null int64
posa_continent               753406 non-null int64
user_location_country        753406 non-null int64
user_location_region         753406 non-null int64
user_location_city           753406 non-null int64
orig_destination_distance    482736 non-null float64
user_id                      753406 non-null int64
is_mobile                    753406 non-null int64
is_package                   753406 non-null int64
channel                      753406 non-null int64
srch_ci                      752449 non-null datetime64[ns]
srch_co                      752450 non-null datetime64[ns]
srch_adults_cnt              753406 non-null int64
srch_children_cnt            753406 non-null int64
srch_rm_cnt                  753406 non-null int64
srch_destination_id          753406 non-null int64
srch_destination_type_id     753406 non-null int64
is_booking                   753406 non-null int64
cnt                          753406 non-null int64
hotel_continent              753406 non-null int64
hotel_country                753406 non-null int64
hotel_market                 753406 non-null int64
hotel_cluster                753406 non-null int64
dtypes: datetime64[ns](3), float64(1), int64(20)
memory usage: 138.0 MB
```

[6]:
```
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75341 entries, 0 to 75340
Data columns (total 24 columns):
date_time                    75341 non-null datetime64[ns]
site_name                    75341 non-null int64
posa_continent               75341 non-null int64
user_location_country        75341 non-null int64
user_location_region         75341 non-null int64
user_location_city           75341 non-null int64
orig_destination_distance    48252 non-null float64
user_id                      75341 non-null int64
is_mobile                    75341 non-null int64
is_package                   75341 non-null int64
channel                      75341 non-null int64
srch_ci                      75243 non-null datetime64[ns]
srch_co                      75243 non-null datetime64[ns]
srch_adults_cnt              75341 non-null int64
```

```
srch_children_cnt             75341 non-null int64
srch_rm_cnt                   75341 non-null int64
srch_destination_id           75341 non-null int64
srch_destination_type_id      75341 non-null int64
is_booking                    75341 non-null int64
cnt                           75341 non-null int64
hotel_continent               75341 non-null int64
hotel_country                 75341 non-null int64
hotel_market                  75341 non-null int64
hotel_cluster                 75341 non-null int64
dtypes: datetime64[ns](3), float64(1), int64(20)
memory usage: 13.8 MB
```

[7]:
```
destinations_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62106 entries, 0 to 62105
Columns: 150 entries, srch_destination_id to d149
dtypes: float64(149), int64(1)
memory usage: 71.1 MB
```

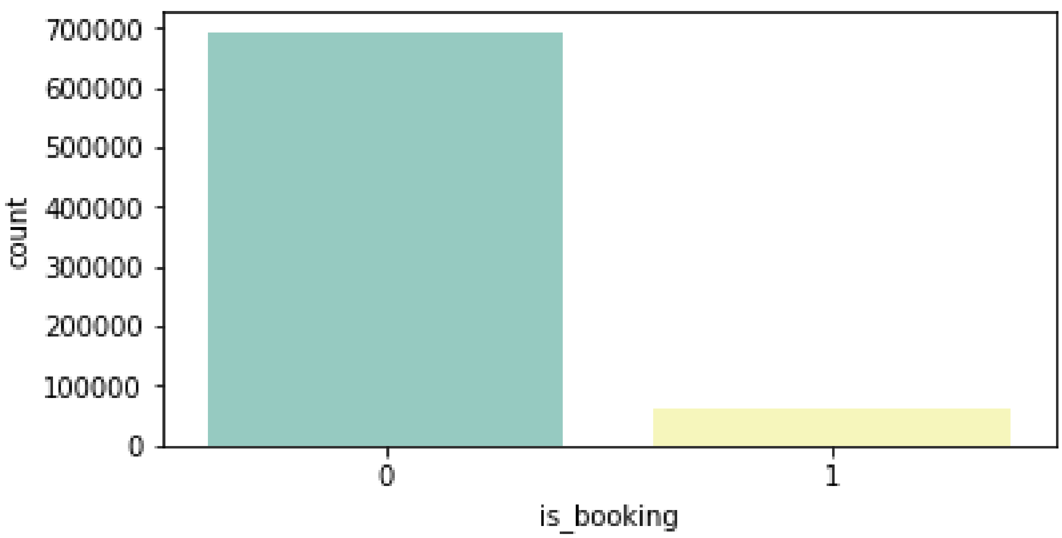
[8]:
```
train_set.corr()['is_booking']
```

[8]:
```
site_name                   -0.012032
posa_continent               0.010616
user_location_country        0.006960
user_location_region         0.007495
user_location_city           0.001340
orig_destination_distance   -0.038670
user_id                      0.002575
is_mobile                   -0.030734
is_package                  -0.077729
channel                      0.025615
srch_adults_cnt             -0.050452
srch_children_cnt           -0.023582
srch_rm_cnt                  0.009334
srch_destination_id          0.024123
srch_destination_type_id     0.041880
is_booking                   1.000000
cnt                         -0.113414
hotel_continent             -0.026935
hotel_country               -0.003840
hotel_market                 0.011368
hotel_cluster               -0.021731
Name: is_booking, dtype: float64
```

[9]:
```
train_set.corr().style.format("{:.4}").background_gradient(cmap=plt.
 ↪get_cmap('coolwarm'), axis=1)
```

[9]: <pandas.io.formats.style.Styler at 0x2bc43a94208>

[10]:
```
fig = plt.figure(figsize=(6,3))
sns.countplot(x='is_booking',data=train_set, order=[0,1],palette="Set3")
plt.show()
```



1.3 Data Processing

[11]:
```
def process_date_info(df):
    df['srch_ci_day'] = df["srch_ci"].apply(lambda x: x.day)
    df['srch_ci_month'] = df["srch_ci"].apply(lambda x: x.month)
    df['srch_ci_year'] = df["srch_ci"].apply(lambda x: x.year)

    df['date_time_day'] = df["date_time"].apply(lambda x: x.day)
    df['date_time_month'] = df["date_time"].apply(lambda x: x.month)
    df['date_time_year'] = df["date_time"].apply(lambda x: x.year)

    df['stay_dur'] = (df['srch_co'] - df['srch_ci']).astype('timedelta64[h]')
```

[12]:
```
process_date_info(train_set)
process_date_info(test_set)
```

[13]:
```
train_set.dropna(axis=0, inplace=True)
test_set.dropna(axis=0, inplace=True)
```

[14]:
```
train_set = pd.merge(train_set, destinations_df, on='srch_destination_id')
test_set = pd.merge(test_set, destinations_df, on='srch_destination_id')

train_set.drop(['orig_destination_distance', 'date_time', 'srch_ci', 'srch_co',␣
↪'user_id', 'srch_destination_id' ], axis=1, inplace=True)
test_set.drop(['orig_destination_distance', 'date_time', 'srch_ci', 'srch_co',␣
↪'user_id', 'srch_destination_id' ], axis=1, inplace=True)
```

[15]:
```
train_set.head()
```

[15]:
```
   site_name  posa_continent  user_location_country  user_location_region  \
0          2               3                     66                   174
1          2               3                     66                   246
2          2               3                     66                   246
3         34               3                    205                   354
4          2               3                     66                   348

   user_location_city  is_mobile  is_package  channel  srch_adults_cnt  \
0                8124          0           1        9                2
1               50661          0           1        9                2
2               50661          0           0        3                4
3               25315          0           0        9                1
4               37377          1           0        9                2

   srch_children_cnt  …      d140      d141     d142      d143      d144  \
0                  1  … -2.279495 -2.274158 -2.25451 -2.278753 -2.279495
1                  0  … -2.279495 -2.274158 -2.25451 -2.278753 -2.279495
2                  0  … -2.279495 -2.274158 -2.25451 -2.278753 -2.279495
3                  0  … -2.279495 -2.274158 -2.25451 -2.278753 -2.279495
4                  0  … -2.279495 -2.274158 -2.25451 -2.278753 -2.279495

       d145      d146      d147      d148      d149
0 -2.279495 -2.279495 -2.279495 -2.279495 -2.262535
1 -2.279495 -2.279495 -2.279495 -2.279495 -2.262535
2 -2.279495 -2.279495 -2.279495 -2.279495 -2.262535
3 -2.279495 -2.279495 -2.279495 -2.279495 -2.262535
4 -2.279495 -2.279495 -2.279495 -2.279495 -2.262535

[5 rows x 174 columns]
```

[16]:
```
feature_cols = train_set.columns.tolist()
feature_cols.remove('is_booking')
label_col = 'is_booking'#as y variable to predict
```

[17]:
```
from sklearn.model_selection import train_test_split

# split train_hotel dataset by 7 to 3
```

```
X_train, X_test, y_train, y_test = train_test_split(train_set[feature_cols],␣
↪train_set[label_col], test_size=0.3, random_state=2020)
```

[18]:
```
X_train.isnull().sum()
```

[18]:
```
site_name                0
posa_continent           0
user_location_country    0
user_location_region     0
user_location_city       0
                        ..
d145                     0
d146                     0
d147                     0
d148                     0
d149                     0
Length: 173, dtype: int64
```

[19]:
```
y_train.isnull().sum()
```

[19]: 0

[20]:
```
X_test.isnull().sum()
```

[20]:
```
site_name                0
posa_continent           0
user_location_country    0
user_location_region     0
user_location_city       0
                        ..
d145                     0
d146                     0
d147                     0
d148                     0
d149                     0
Length: 173, dtype: int64
```

[21]:
```
y_test.isnull().sum()
```

[21]: 0

1.4 Q1 : Wrangle with the dataset 1 and select any 4 algorithms, search out the "feature importance" to get the insight to design and conduct your AB test in part 2. The usual step of dividing the train_hotel dataset into a "learning" set and "model testing" set, use the APIs from ski-learn package of python to find the "optimal" model based on your selection criteria of the confusion matrix. (10%)

1.4.1 Light GBM

[22]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import xgboost as xgb
import lightgbm as lgb

from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪precision_recall_fscore_support
from sklearn.model_selection import KFold, cross_val_score, train_test_split
```

[23]:
```
def plot_feature_importances(model,importances, feature_cols,␣
↪features_num=None, figsize=(15,12), is_print=False):
    if features_num is None:
        features_num = len(feature_cols)

    indices = np.argsort(importances)[::-1]
    featurerank=[]
    for f in range(features_num):
        featurerank.append([feature_cols[indices[f]], importances[indices[f]]])
        if is_print:
            print("%d. feature %s (%f)" % (f + 1, feature_cols[indices[f]],␣
↪importances[indices[f]]))

    plt.figure(figsize=figsize)
    feature_imp = pd.Series(importances,index=feature_cols).
↪sort_values(ascending=False)
    feature_imp = pd.DataFrame(featurerank, columns=['name', 'importances'])
    sns.barplot(x=feature_imp['importances'], y= feature_imp['name'])
    plt.xlabel('Feature Importance Score')
    plt.ylabel('Features')
    plt.title("Visualizing Important Features")
    plt.show()
```

[24]:
```
# n_estimators, learning_rate, max_depth,
model = lgb.LGBMClassifier( random_state=2020, importance_type='gain')
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
y_pred_prob = model.predict_proba(X_test)
```

[25]:
```
auc = metrics.roc_auc_score(y_test,y_pred_prob[:, 1])
print('The AUC is :{}'.format(auc))
```

```
The AUC is :0.7747993013030329
```

[26]:
```
cm = confusion_matrix(y_test, y_pred)
cm
```

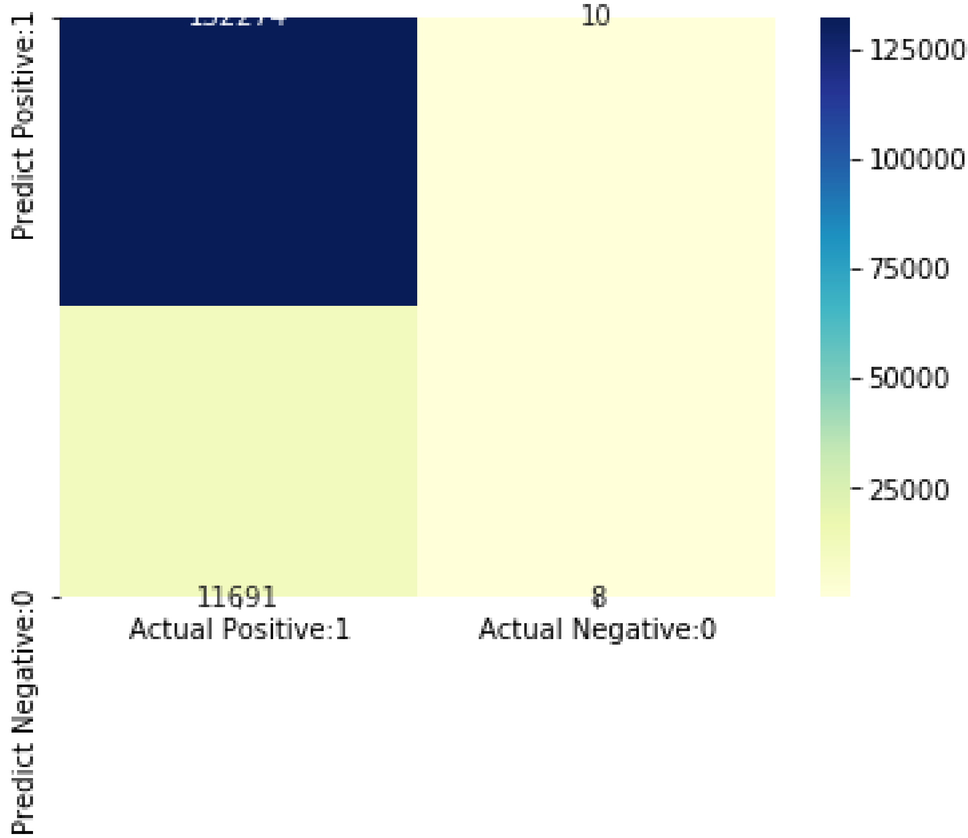
[26]:
```
array([[132274,     10],
       [ 11691,      8]], dtype=int64)
```

[27]:
```
print('Confusion matrix\n\n', cm)
print('\nTrue Positives(TP) = ', cm[0,0])
print('\nTrue Negatives(TN) = ', cm[1,1])
print('\nFalse Positives(FP) = ', cm[0,1])
print('\nFalse Negatives(FN) = ', cm[1,0])
```

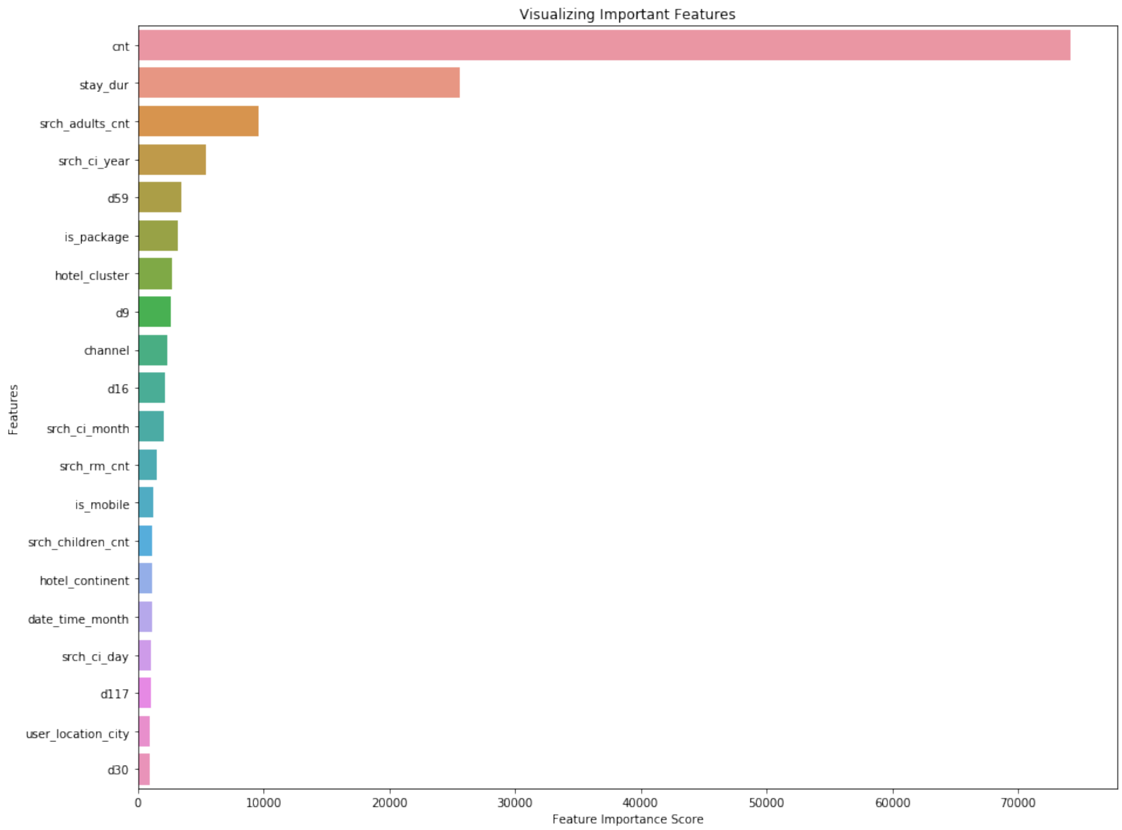
```
Confusion matrix

 [[132274     10]
 [ 11691      8]]

True Positives(TP) =  132274

True Negatives(TN) =  8

False Positives(FP) =  10

False Negatives(FN) =  11691
```

[28]:
```
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual␣
 ↪Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:
 ↪0'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2bc3623e948>
```



[29]:
```
importances = model.feature_importances_
plot_feature_importances(model, importances, feature_cols, features_num=20)
```



For the lightGBM, the auc is 0.7747. Also, the Most important feature is cnt as the figure shows.

1.4.2 Random Forest

[30]:
```
# Random Forest
model_rf = RandomForestClassifier(random_state=2020)
model_rf.fit(X_train, y_train)
y_pred_rf = model_rf.predict(X_test)
y_pred_prob_rf = model_rf.predict_proba(X_test)
```

```
C:\Users\maris\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

[31]:
```
auc = metrics.roc_auc_score(y_test,y_pred_prob_rf[:, 1])
print('The AUC is :{}'.format(auc))
```

```
The AUC is :0.6398687619639043
```

[32]:
```
cm = confusion_matrix(y_test, y_pred_rf)
cm
```

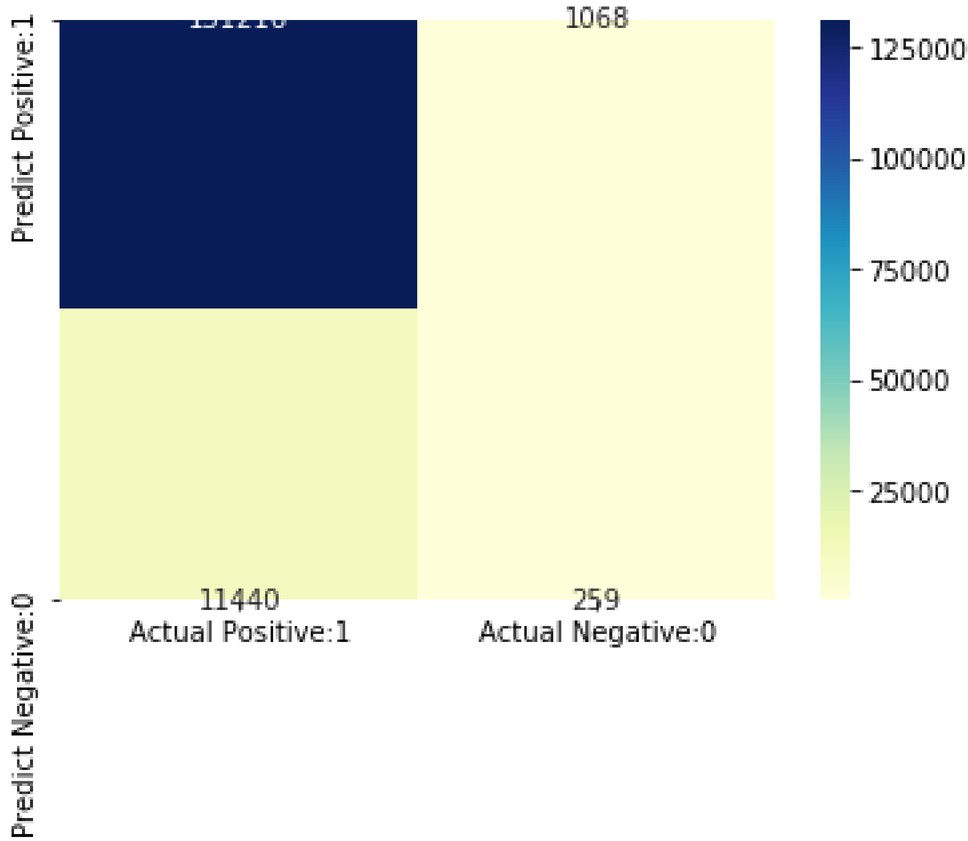
[32]:
```
array([[131216,   1068],
       [ 11440,    259]], dtype=int64)
```

[33]:
```
print('Confusion matrix\n\n', cm)
print('\nTrue Positives(TP) = ', cm[0,0])
print('\nTrue Negatives(TN) = ', cm[1,1])
print('\nFalse Positives(FP) = ', cm[0,1])
print('\nFalse Negatives(FN) = ', cm[1,0])
```

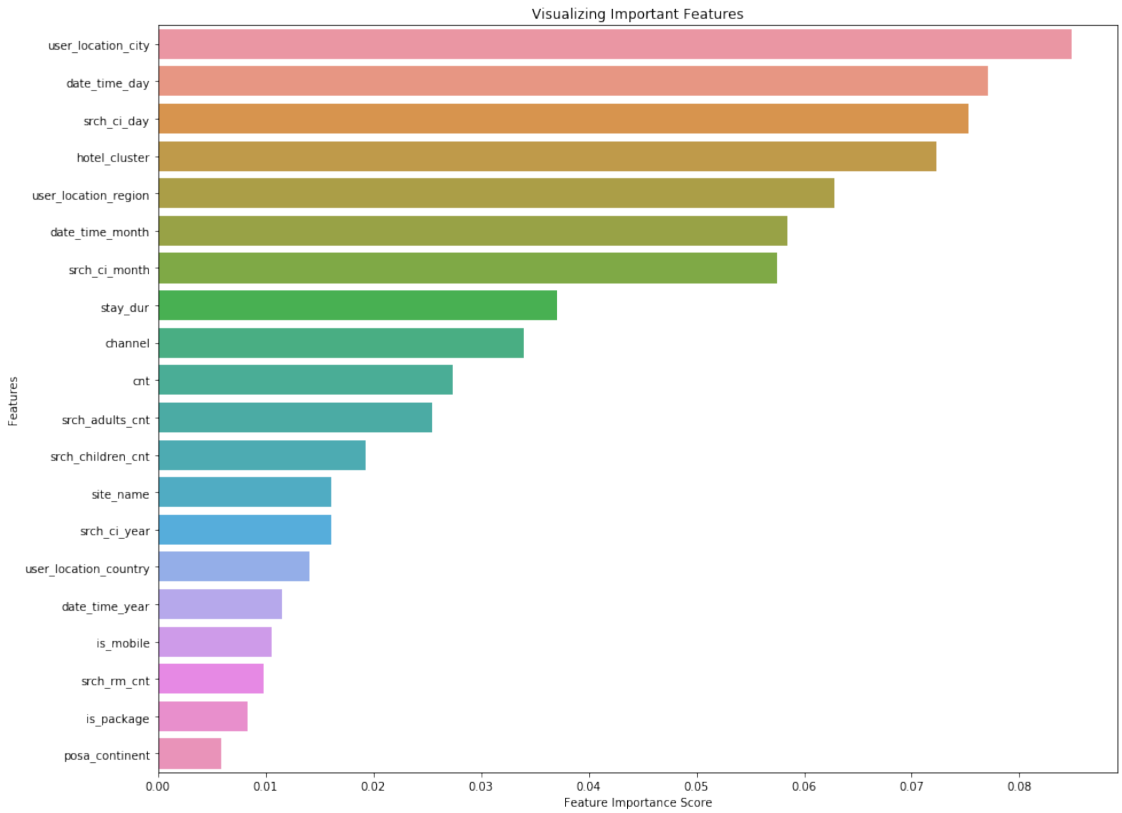
```
Confusion matrix

 [[131216   1068]
 [ 11440    259]]

True Positives(TP) =  131216

True Negatives(TN) =  259

False Positives(FP) =  1068

False Negatives(FN) =  11440
```

[34]:
```
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual␣
 ↪Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:
 ↪0'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2bc38cc8108>
```



[35]:
```
importances = model_rf.feature_importances_
plot_feature_importances(model_rf, importances, feature_cols, features_num=20)
```



The AUC of the Random Forest classifier is 0.6399. The most important feature for the Random Forest is user_location city

1.4.3 Gradient Boosting

[36]:
```
model_GB = GradientBoostingClassifier(random_state=2020)
model_GB.fit(X_train, y_train)
y_pred_GB = model_rf.predict(X_test)
y_pred_prob_GB = model_GB.predict_proba(X_test)
```

[37]:
```
auc = metrics.roc_auc_score(y_test,y_pred_prob_GB[:, 1])
print('The AUC is :{}'.format(auc))
```

```
The AUC is :0.7705682676850936
```

[40]:
```
cm2 = confusion_matrix(y_test, y_pred_GB)
cm2
```

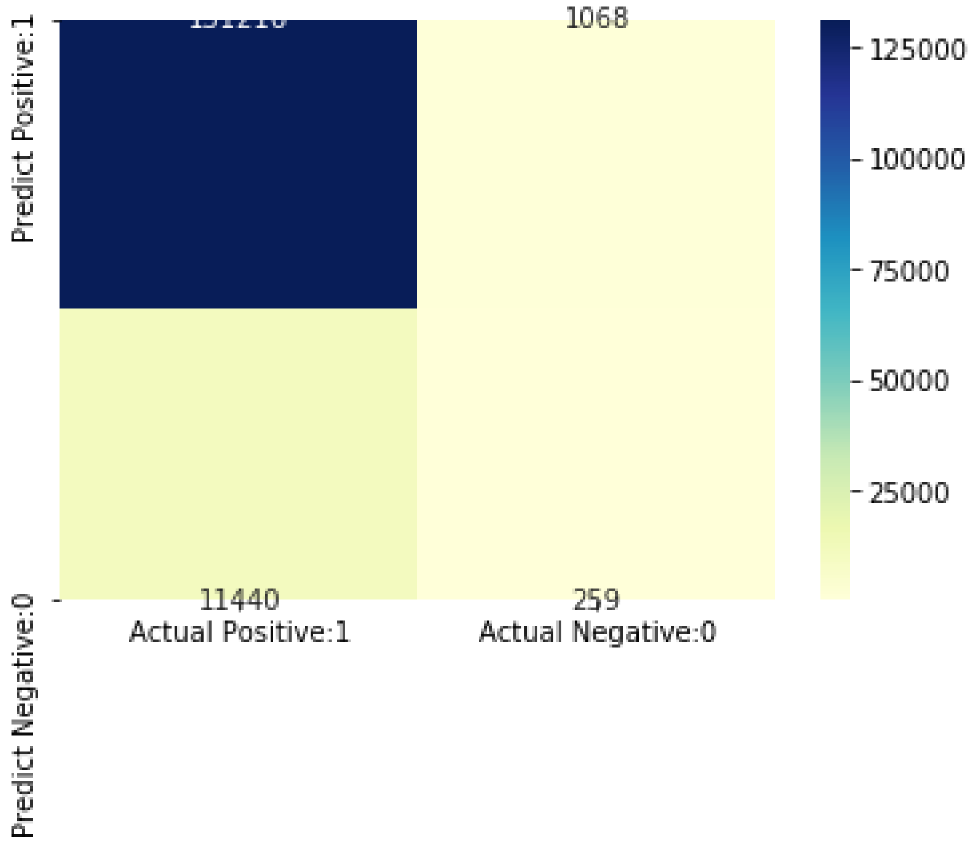
[40]:
```
array([[131216,   1068],
       [ 11440,    259]], dtype=int64)
```

[41]:
```
print('Confusion matrix\n\n', cm2)
print('\nTrue Positives(TP) = ', cm2[0,0])
print('\nTrue Negatives(TN) = ', cm2[1,1])
print('\nFalse Positives(FP) = ', cm2[0,1])
print('\nFalse Negatives(FN) = ', cm2[1,0])
```

```
Confusion matrix

 [[131216   1068]
 [ 11440    259]]

True Positives(TP) =  131216

True Negatives(TN) =  259

False Positives(FP) =  1068

False Negatives(FN) =  11440
```

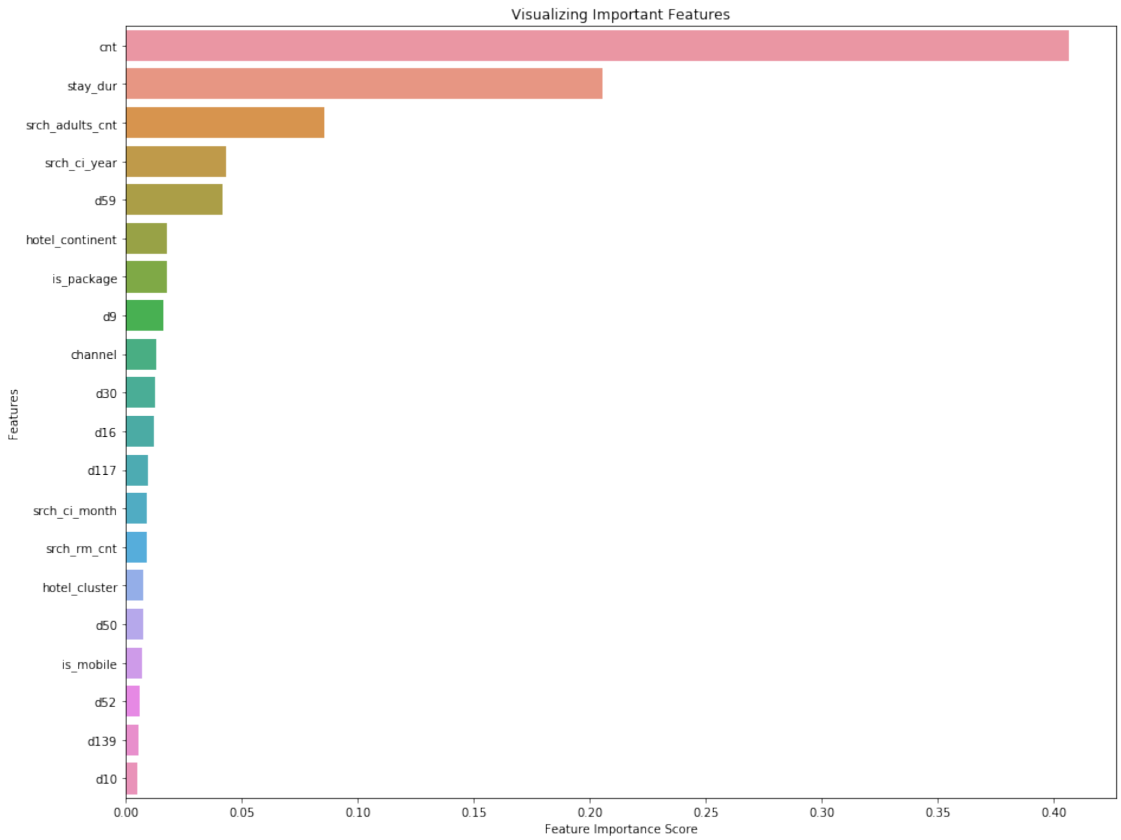
[42]:
```
cm2_matrix = pd.DataFrame(data=cm2, columns=['Actual Positive:1', 'Actual␣
↪Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:
↪0'])

sns.heatmap(cm2_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

[42]: <matplotlib.axes._subplots.AxesSubplot at 0x1ad8a3af708>



[43]:

```
importances = model_GB.feature_importances_
plot_feature_importances(model_GB, importances, feature_cols, features_num=20)
```



The AUC of Gradient Boosting is 0.7705682676850936. The most important feature for GB is 'cnt'.

1.4.4 XGboost

[44]:
```
# n_estimators, learning_rate, max_depth,
model_xg = xgb.XGBClassifier(random_state=2020)
model_xg.fit(X_train, y_train)
y_pred_xg = model_xg.predict(X_test)
y_pred_prob = model_xg.predict_proba(X_test)
```

[45]:
```
auc = metrics.roc_auc_score(y_test,y_pred_prob[:, 1])
print('The AUC is :{}'.format(auc))
```

```
The AUC is :0.7709919873274798
```

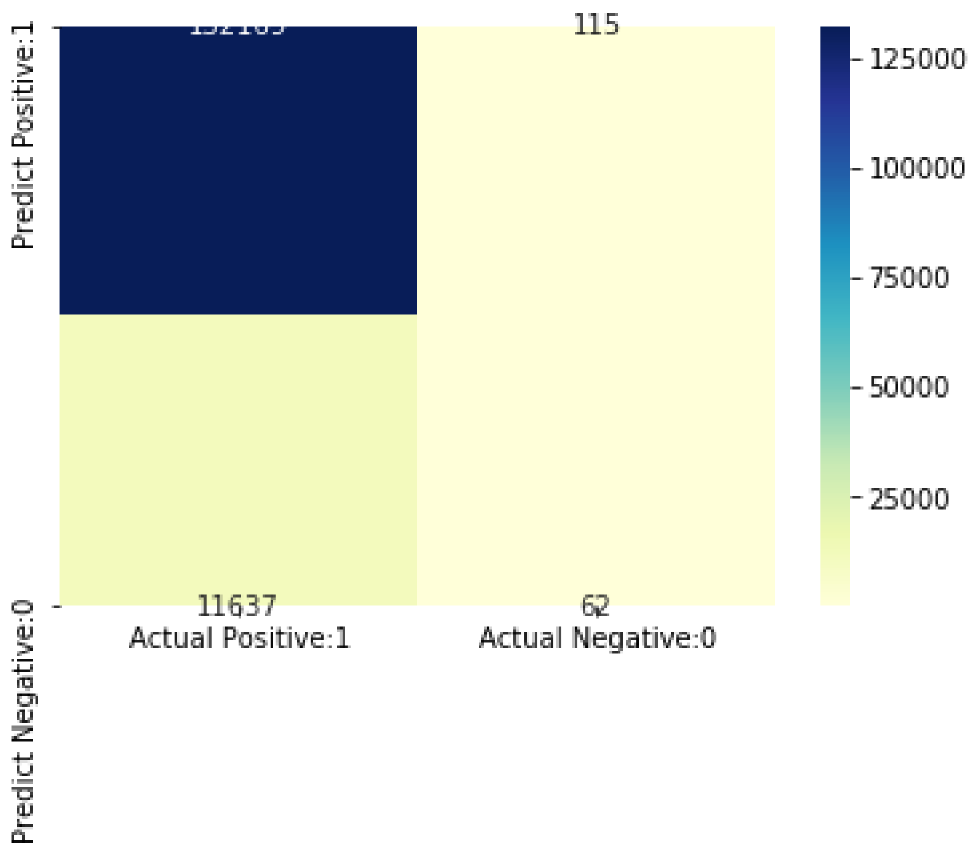
The AUC for the XGboost is 0.771

[46]:
```
cm3 = confusion_matrix(y_test, y_pred_xg)
cm3
```

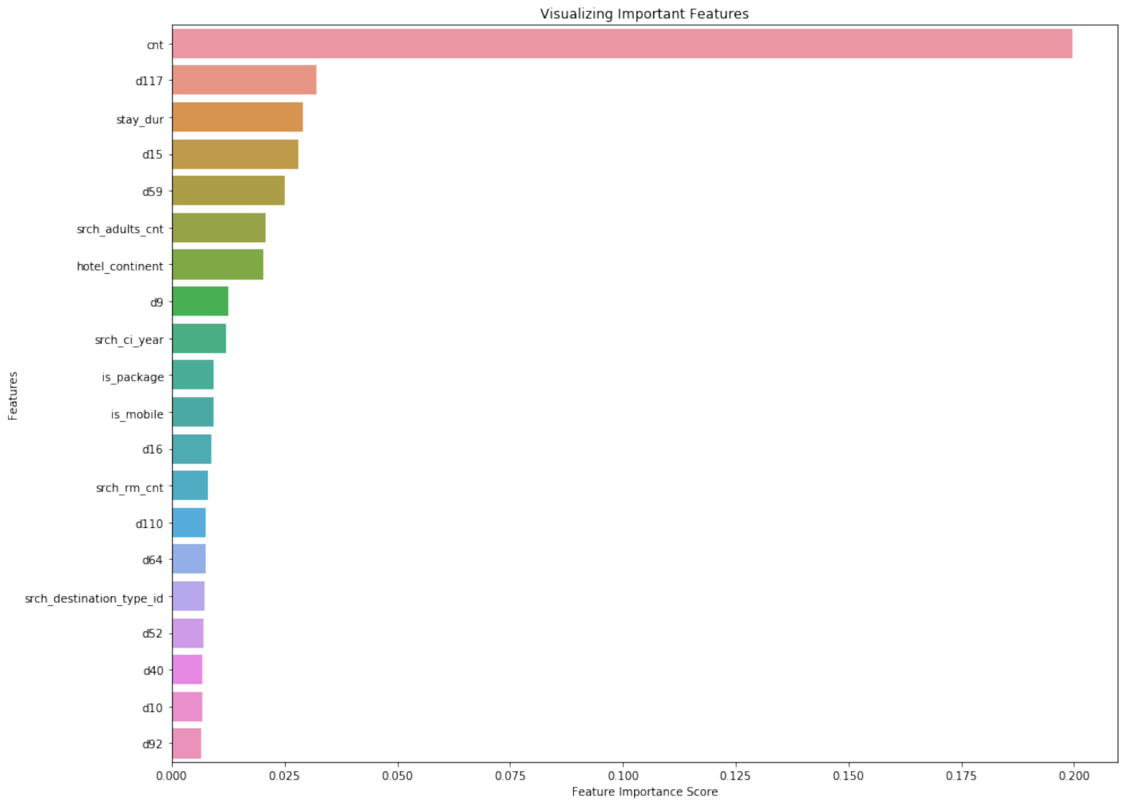
[46]: array([[132169,   115],
       [11637,   62]], dtype=int64)

[47]:
```
cm3_matrix = pd.DataFrame(data=cm3, columns=['Actual Positive:1', 'Actual␣
↪Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:
↪0'])

sns.heatmap(cm3_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1ad8a760d08>



[48]:
```
importances = model_xg.feature_importances_
plot_feature_importances(model_xg, importances, feature_cols, features_num=20)
```



The AUC of XGboost is 0.771. The most important feature for XGboost is 'cnt'

Q1 Conclusion: Based on the above models, we conclude that the most important feature is 'cnt'. For the model accuracy, the Light GBM is the best performance model which the AUC rate is 0.7747.

1.5 Q2: Now suppose you find out the "most" important feature (rank one), select a sample of at least 5000 customers from the test_hotel dataset, (by a random seed number) with the identified important feature to conduct a AB test on the important feature. (10%)

Now, we got the most important feature 'cnt'.Then we want test how important feature affect the y label(is_booking). So, we separate calculate the mean of is_booking==1 and is_booking ==0 respectively.

1.6 Statistically

H0: The most important feature:cnt is no effect on is_booking

Ha:The most important feature:cnt has effect on is_booking

[38]:
```
test_set.groupby(['is_booking']).cnt.mean()
```

[38]:
```
is_booking
0    1.516454
1    1.019395
Name: cnt, dtype: float64
```

[39]:
```
test_set.groupby(['is_booking']).cnt.std()
```

[39]:
```
is_booking
0    1.195025
1    0.157208
Name: cnt, dtype: float64
```

[40]:
```
import statsmodels.stats.weightstats as st

the_most_important_feature = 'cnt'

idxs = np.random.randint(0, test_set.shape[1], size=5000)
sample_test_set = test_set.iloc[idxs]

arr = sample_test_set[(sample_test_set.
 ↪is_booking==0)][the_most_important_feature].values
arr2 = sample_test_set[(sample_test_set.
 ↪is_booking==1)][the_most_important_feature].values


stats, pval = st.ztest(arr, arr2, value=0)

print('test statistic is: {} \npvalue of the t-test is: {}'.format(stats, pval))
```

```
test statistic is: 9.218378906197856
pvalue of the t-test is: 3.0162562970014014e-20
```

[41]:
```
if pval<=0.05:
    print("reject the null hypothesis, statistically significant")
else:
    print("Not reject the null hypothesis, statistically insignificant")
```

```
reject the null hypothesis, statistically significant
```

Q2 Conclusion: Since the p-value is too small, so we reject the null hypo which means the most important feature has such large effect on label "is_booking".Cnt could affect the conversion of customer's booking

1.7 Another Way of ABtest without statistically(Optional)

1.7.1 A-test-with the most important feature-"cnt"

[42]:
```
feature_cols = test_set.columns.tolist()
feature_cols.remove('is_booking')

label_col = 'is_booking'

from sklearn.model_selection import train_test_split
#split test set
X_train, X_test, y_train, y_test = train_test_split(test_set[feature_cols],␣
 ↪test_set[label_col], test_size=5000, random_state=0)
```

[43]:
```
model_A = lgb.LGBMClassifier(random_state=2020, importance_type='gain')
model_A.fit(X_train, y_train)
y_pred_A = model.predict(X_test)
y_pred_prob_A = model.predict_proba(X_test)
```

[44]:
```
auc_1 = metrics.roc_auc_score(y_test,y_pred_prob_A[:, 1])
print('The AUC is :{}'.format(auc_1))
```

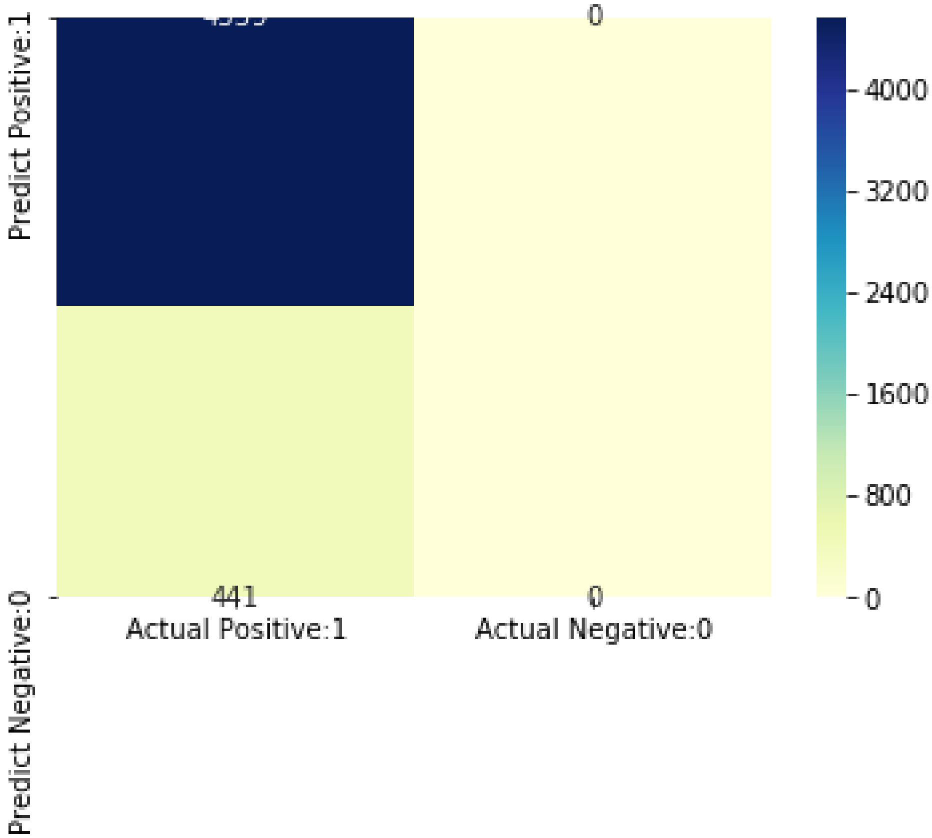
```
The AUC is :0.7833718557248154
```

[45]:
```
cmA = confusion_matrix(y_test, y_pred_A)
cmA
```

[45]:
```
array([[4559,    0],
       [ 441,    0]], dtype=int64)
```

[46]:
```
cmA_matrix = pd.DataFrame(data=cmA, columns=['Actual Positive:1', 'Actual␣
 ↪Negative:0'],
                                  index=['Predict Positive:1', 'Predict Negative:
 ↪0'])

sns.heatmap(cmA_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2bc38fc3748>
```



1.7.2 B-test

[47]:
```
the_most_important_feature = 'cnt'

feature_cols = test_set.columns.tolist()
feature_cols.remove('is_booking')
feature_cols.remove(the_most_important_feature)

label_col = 'is_booking'

from sklearn.model_selection import train_test_split

# split train_hotel dataset
X_train, X_test, y_train, y_test = train_test_split(test_set[feature_cols],␣
↪test_set[label_col], test_size=5000, random_state=2020)
```

[48]:
```
model_B = lgb.LGBMClassifier(random_state=0)
model_B.fit(X_train, y_train)
y_pred_B = model_B.predict(X_test)
y_pred_prob_B = model_B.predict_proba(X_test)
```

[49]:
```
auc_2 = metrics.roc_auc_score(y_test,y_pred_prob_B[:, 1])
print('The AUC is :{}'.format(auc_2))
```

```
The AUC is :0.6634666580295129
```

[50]:
```
auc_2-auc_1
```

[50]:
```
-0.1199051976953025
```

The difference of with/without importance feature is 0.1168 on model accuracy. There is such effect on model accuracy with the most important feature 'cnt'.

[ ]:
```

```