Questions

February 6, 2022

1 Data Science Challenge

[118]:
```
# To install packages that are not installed by default, uncomment the last two lines
# of this cell and replace <package list> with a list of necessary packages.
# This will ensure the notebook has all the dependencies and works everywhere.

#import sys
#!{sys.executable} -m pip install <package list>
```

[119]:
```
#Libraries
import pandas as pd
pd.set_option("display.max_columns", 101)
```

1.1 Data Description

Column	Description
id	The unique ID assigned to every hotel.
region	The region in which the hotel is located..
latitude	The latitude of the hotel.
longitude	The longitude of the hotel.
accommodation_type	The type of accommodation offered by the hotel. For example: Private room, Entire house/apt, etc.
cost	The cost of booking the hotel for one night. (in $$)
minimum_nights	The minimum number of nights stay required.
number_of_reviews	The number of reviews accumulated by the hotel.
reviews_per_month	The average number of reviews received by the hotel per month.
owner_id	The unique ID assigned to every owner. An owner can own multiple hotels.
owned_hotels	The number of hotels owned by the owner.

Column	Description
yearly_availability	It indicates if the hotel accepts bookings around the year. Values are 0 (not available for 365 days in a year) and 1 (available for 365 days in a year).

1.2 Data Wrangling & Visualization

[120]:
```
# Dataset is already loaded below
data = pd.read_csv("train.csv")
```

[121]:
```
data.head()
```

[121]:
```
      id     region  latitude  longitude accommodation_type  cost  \
0  13232  Manhattan  40.71854  -74.00439    Entire home/apt   170
1    246   Brooklyn  40.64446  -73.95030    Entire home/apt    65
2  19091     Queens  40.78573  -73.81062       Private room    85
3  34305  Manhattan  40.73863  -73.98002       Private room   210
4    444  Manhattan  40.82426  -73.94630        Shared room    75

   minimum_nights  number_of_reviews  reviews_per_month   owner_id  \
0               5                  7               0.56     929983
1               3                238               2.30     281764
2               1                  0                NaN   19923341
3              30                  0                NaN  200380610
4               3                 38               0.42     745069

   owned_hotels  yearly_availability
0             1                    0
1             1                    0
2             1                    1
3            65                    1
4             3                    1
```

[122]:
```
#Explore columns
data.columns
```

[122]:
```
Index(['id', 'region', 'latitude', 'longitude', 'accommodation_type', 'cost',
       'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'owner_id',
       'owned_hotels', 'yearly_availability'],
      dtype='object')
```

[123]:
```
#Description
data.describe()
```

[123]:
```
                 id     latitude    longitude         cost  minimum_nights  \
count   2870.000000  2870.000000  2870.000000  2870.000000     2870.000000
mean   26760.657143    40.731224   -73.950158   195.943206       11.530314
std    14140.930062     0.054942     0.049745   406.184714       37.972339
min        0.000000    40.507080   -74.242850    10.000000        1.000000
25%    15931.750000    40.692462   -73.984003    75.000000        1.000000
50%    28946.500000    40.728250   -73.956720   120.000000        3.000000
75%    38478.500000    40.762658   -73.934202   200.000000        6.000000
max    48893.000000    40.898730   -73.721730  9999.000000      999.000000

       number_of_reviews  reviews_per_month      owner_id  owned_hotels  \
count        2870.000000        2194.000000  2.870000e+03   2870.000000
mean           16.315331           1.157502  7.202195e+07      8.411498
std            32.481722           1.355028  8.076516e+07     27.105522
min             0.000000           0.010000  2.787000e+03      1.000000
25%             1.000000           0.240000  7.388002e+06      1.000000
50%             4.000000           0.650000  3.352708e+07      1.000000
75%            16.000000           1.530000  1.207625e+08      3.000000
max           395.000000          10.370000  2.738123e+08    327.000000

       yearly_availability
count          2870.000000
mean              0.498606
std               0.500085
min               0.000000
25%               0.000000
50%               0.000000
75%               1.000000
max               1.000000
```

[124]:
```
data.drop(['id','owner_id'],axis=1,inplace=True)
```

[125]:
```
#data visualization(univariant)
```

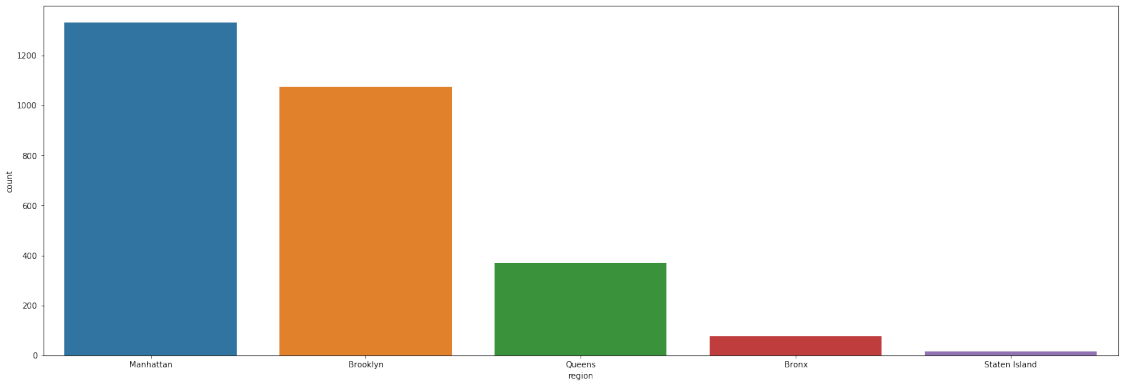
[126]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

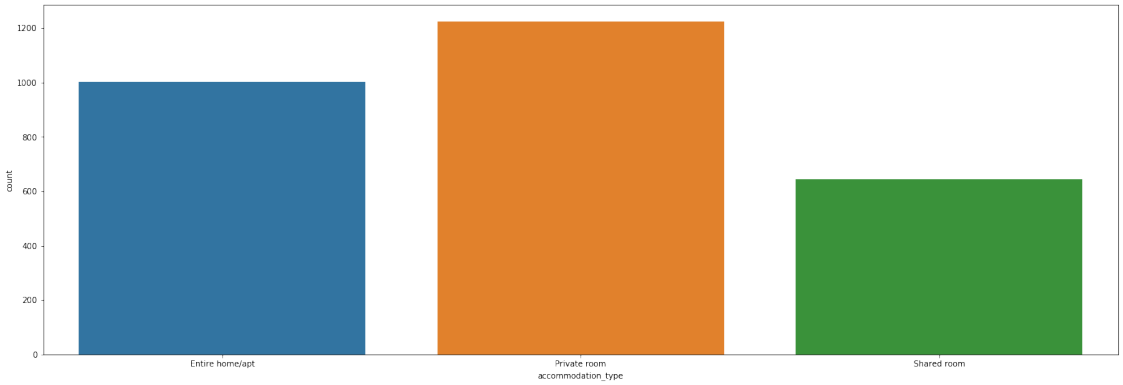
[127]:
```
cat_col = ['region','accommodation_type','yearly_availability']
cont_col =  data.drop(cat_col,axis=1).columns
```

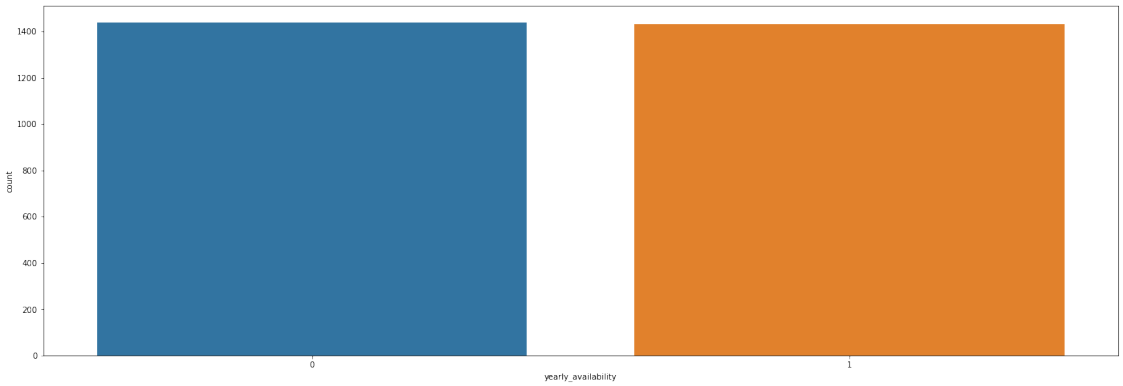
[128]:
```
cant_column
```

[128]:
```
Index(['latitude', 'longitude', 'cost', 'minimum_nights', 'number_of_reviews',
       'reviews_per_month', 'owned_hotels'],
      dtype='object')
```

[129]:
```
#3 plot for cat column
for col in cat_col:
    plt.figure(figsize=(24,8))
    sns.countplot(data[col])
```







[130]:
```
for col in cat_col:
    print(f"freq table for col ==>. {col}")
    print(data[col].value_counts(),end='\n\n')
```

```
freq table for col ==>. region
Manhattan        1333
Brooklyn         1075
Queens            370
Bronx              78
Staten Island      14
Name: region, dtype: int64

freq table for col ==>. accommodation_type
Private room       1225
Entire home/apt    1002
Shared room         643
Name: accommodation_type, dtype: int64

freq table for col ==>. yearly_availability
0    1439
1    1431
Name: yearly_availability, dtype: int64
```

[131]:
```
#3 missing value treatment
data.isna().sum()
```

[131]:
```
region                  0
latitude                0
longitude               0
accommodation_type      0
cost                    0
minimum_nights          0
number_of_reviews       0
reviews_per_month     676
owned_hotels            0
yearly_availability     0
dtype: int64
```

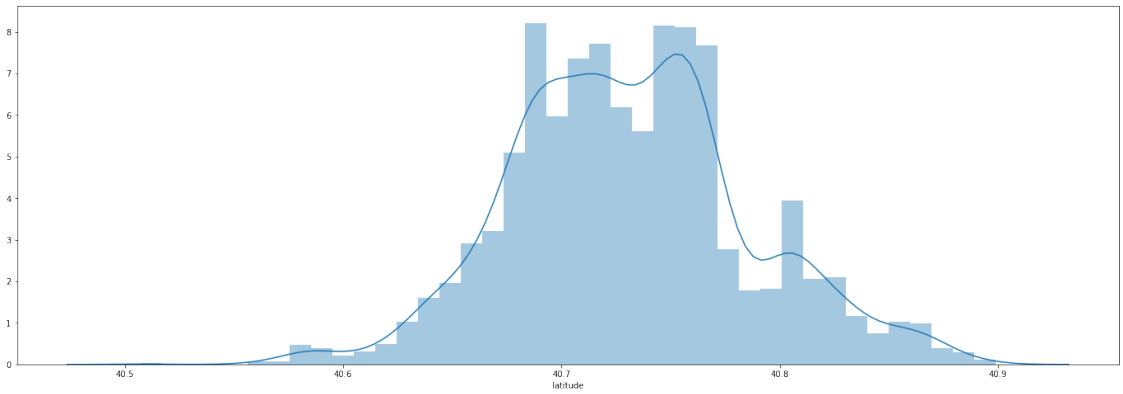
[132]:
```
data["reviews_per_month"] = data["reviews_per_month"].
↪fillna(data["reviews_per_month"].median())
```

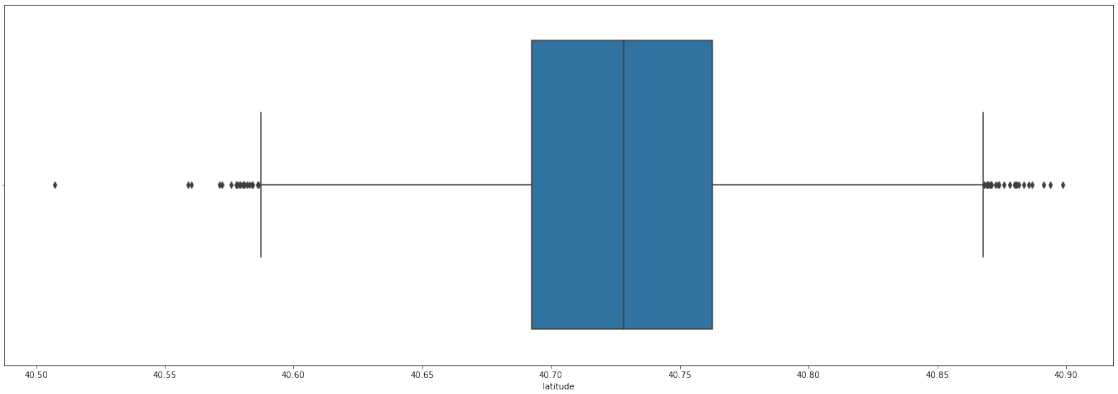
[133]:
```
data.isna().sum()
```

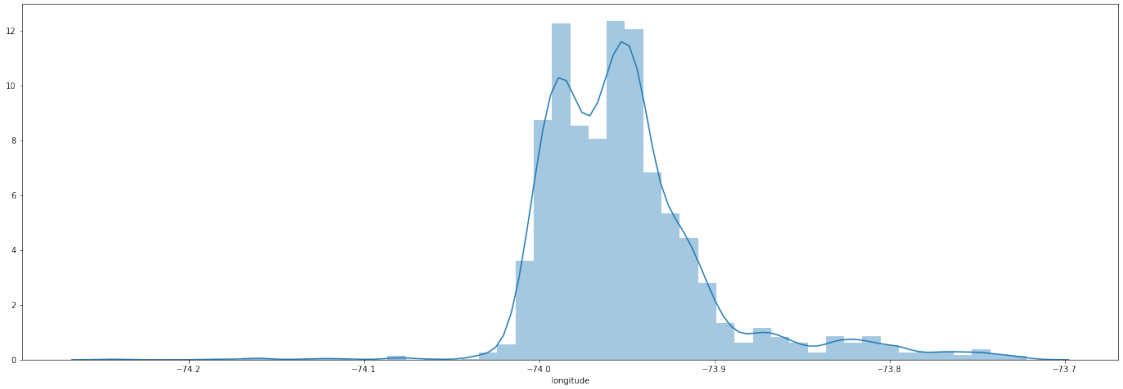
[133]:
```
region                0
latitude              0
longitude             0
```

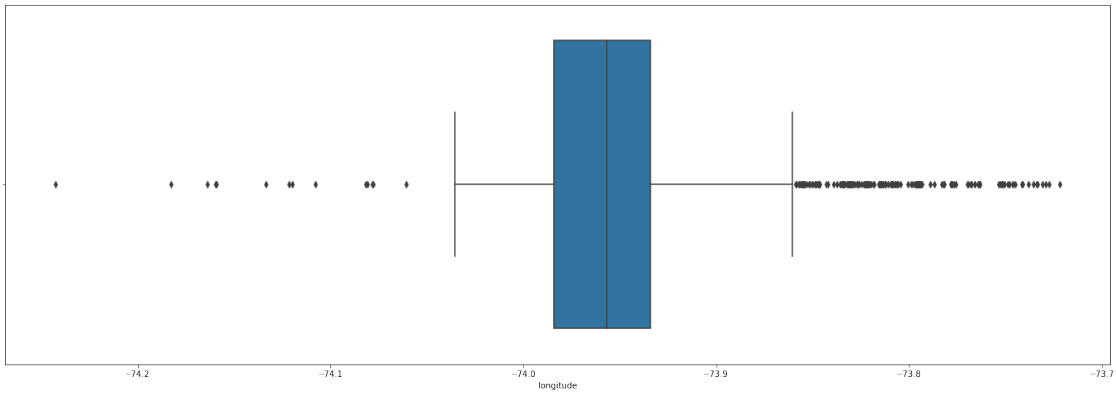
```
accommodation_type     0
cost                   0
minimum_nights         0
number_of_reviews      0
reviews_per_month      0
owned_hotels           0
yearly_availability    0
dtype: int64
```

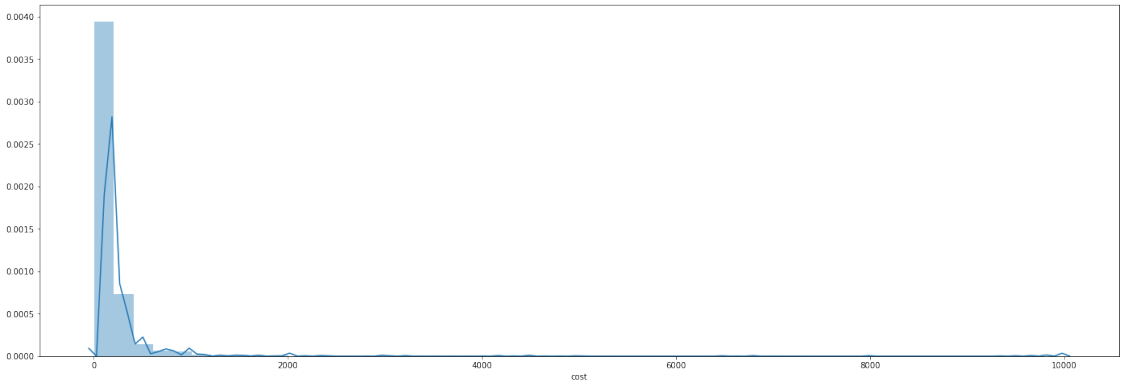
[134]:
```
#3

for col in cont_col:
    plt.figure(figsize=(24,8))
    sns.distplot(data[col])
    plt.show()
    plt.figure(figsize=(24,8))
    sns.boxplot(data[col])
```

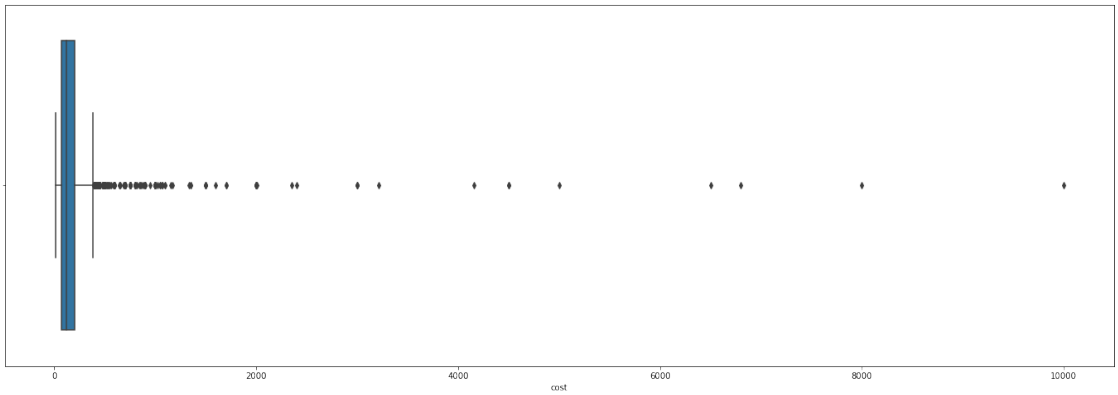


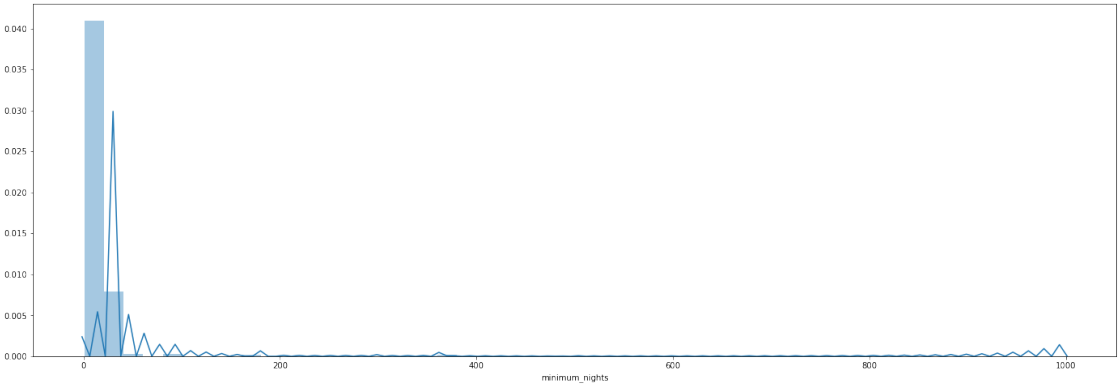


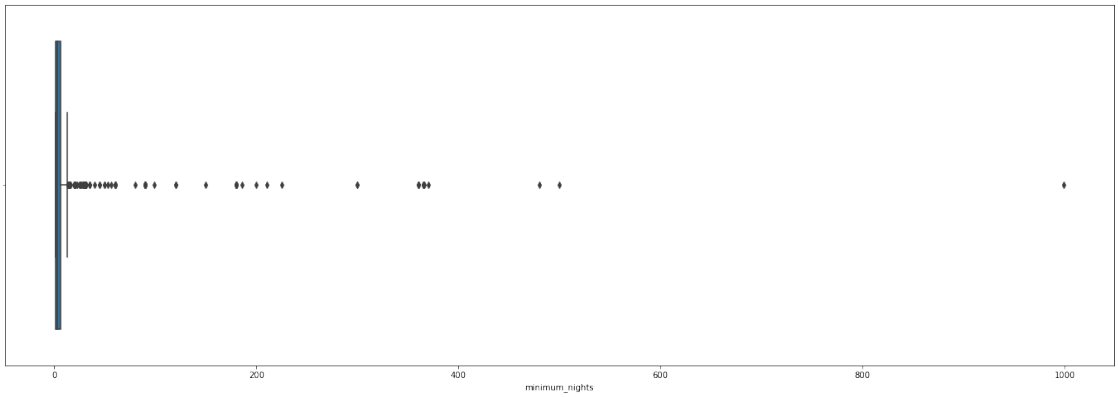

12
10
8
6
4
2
0
−74.2
−74.1
−74.0
−73.9
−73.8
−73.7
longitude

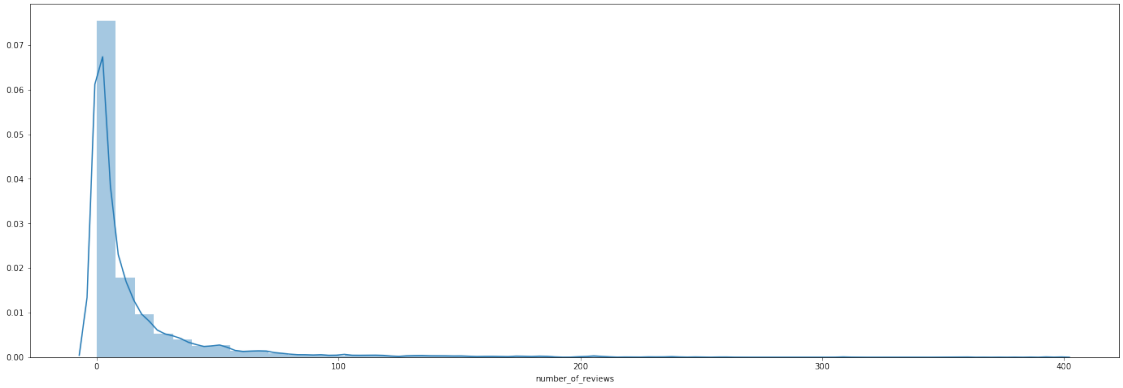

−74.2
−74.1
−74.0
−73.9
−73.8
−73.7
longitude

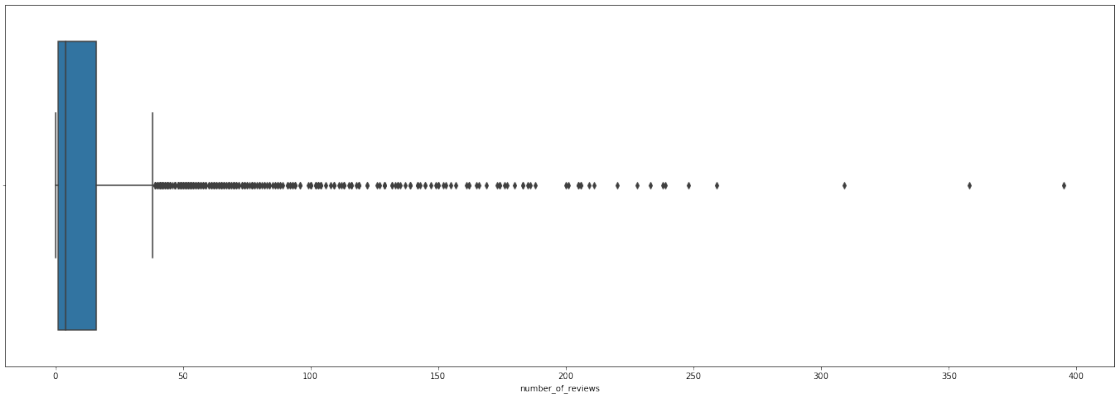

0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
2000
4000
6000
8000
10000
cost

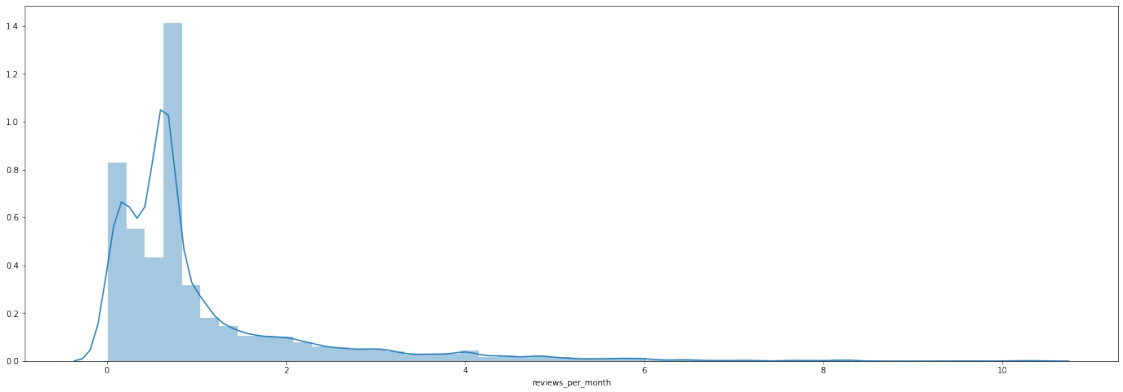

0
2000
4000
6000
8000
10000
cost

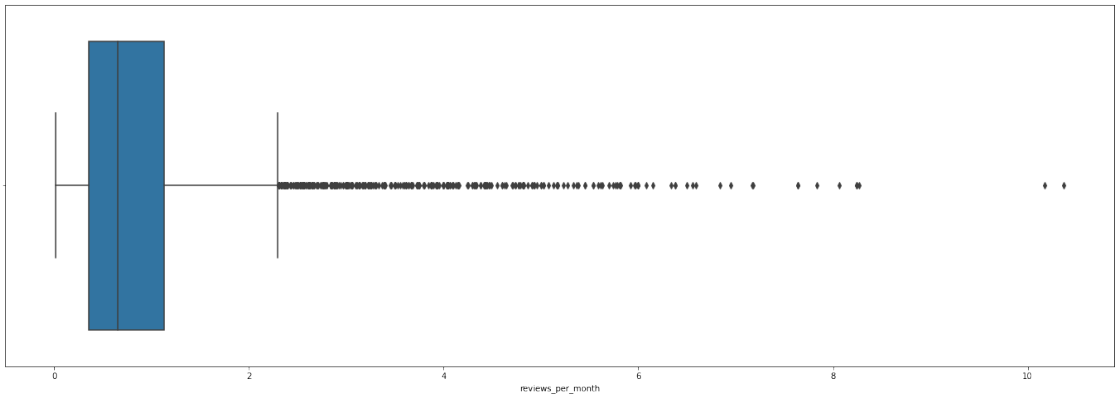

0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000
minimum_nights

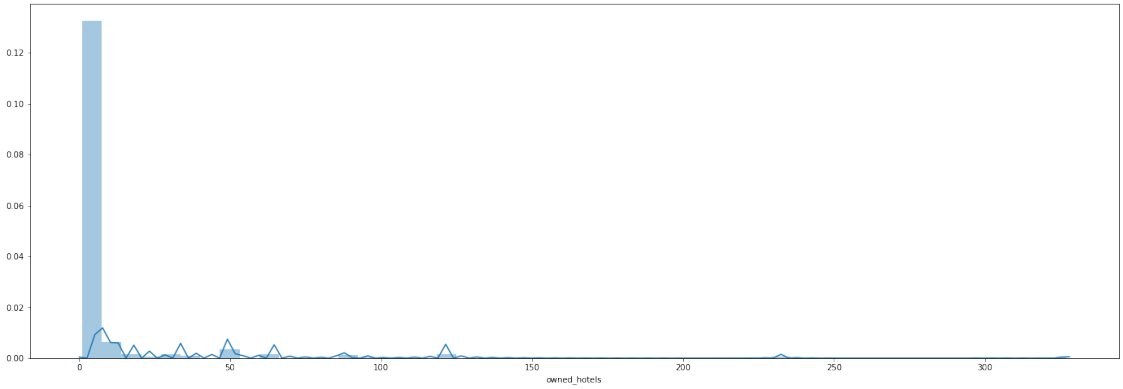

0
200
400
600
800
1000
minimum_nights

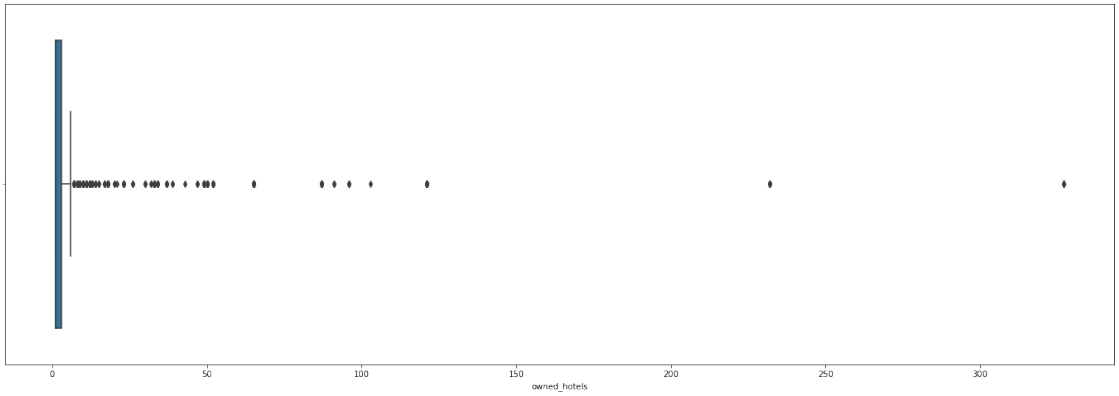

0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
100
200
300
400
number_of_reviews


0
50
100
150
200
250
300
350
400
number_of_reviews


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
reviews_per_month







[135]:
```
#3 bivarinat

#3 t-test
#3 chi sq test
#3 correlation test
```

[136]:
```
from sklearn.preprocessing import LabelBinarizer
import numpy as np
encoder = {}
for col in cat_col:
    if col != 'yearly_availability':
        try:
            lb = LabelBinarizer()
            lb.fit(data[col])
            #3data.reset_index(drop=True,inplace=True)
            encoder_data = lb.transform(data[col])
            encoder_data = pd.DataFrame(encoder_data,columns=lb.classes_)
            data.drop([col],axis=1,inplace = True)
            data = pd.concat([data,encoder_data],axis=1)
            encoder[col] = lb
        except Exception as er:
            print(er)
            print(col)
```

[137]:
```
data
```

[137]:

	latitude	longitude	cost	minimum_nights	number_of_reviews	\
0	40.71854	-74.00439	170	5	7	
1	40.64446	-73.95030	65	3	238	
2	40.78573	-73.81062	85	1	0	
3	40.73863	-73.98002	210	30	0	
4	40.82426	-73.94630	75	3	38	
…	…	…	…	…	…	
2865	40.74316	-73.98038	400	2	0	
2866	40.73523	-73.99465	180	3	2	
2867	40.76619	-73.98987	179	3	17	
2868	40.74637	-73.97207	200	30	0	
2869	40.79208	-73.96482	1000	30	24	

	reviews_per_month	owned_hotels	yearly_availability	Bronx	Brooklyn	\
0	0.56	1	0	0	0	
1	2.30	1	0	0	1	
2	0.65	1	1	0	0	
3	0.65	65	1	0	0	
4	0.42	3	1	0	0	
…	…	…	…	…	…	
2865	0.65	1	1	0	0	
2866	0.07	1	1	0	0	
2867	0.67	1	0	0	0	
2868	0.65	49	1	0	0	
2869	0.33	11	1	0	0	

	Manhattan	Queens	Staten Island	Entire home/apt	Private room	\

```
0              1       0              0              1              0
1              0       0              0              1              0
2              0       1              0              0              1
3              1       0              0              0              1
4              1       0              0              0              0
…          …     …            …              …              …
2865           1       0              0              0              1
2866           1       0              0              0              1
2867           1       0              0              1              0
2868           1       0              0              0              1
2869           1       0              0              0              0

      Shared room
0                0
1                0
2                0
3                0
4                1
…              …
2865             0
2866             0
2867             0
2868             0
2869             1

[2870 rows x 16 columns]
```

[138]:
```
from sklearn.model_selection import train_test_split

x = data.drop(["yearly_availability"],axis=1)
y = data["yearly_availability"]
x_train, x_test , y_train, y_test = train_test_split(x,y,test_size=.3)
```

[139]:
```
from sklearn.ensemble import  RandomForestClassifier
from sklearn.model_selection  import GridSearchCV

rf = RandomForestClassifier()

parameter = {"n_estimators":[5,6,7,8,10,15,20]}
gv = GridSearchCV(rf,parameter)
gv.fit(x_train,y_train)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/model_selection/_split.py:1978:
FutureWarning: The default value of cv will change from 3 to 5 in version 0.22.
Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

[139]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators='warn', n_jobs=None,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'n_estimators': [5, 6, 7, 8, 10, 15, 20]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

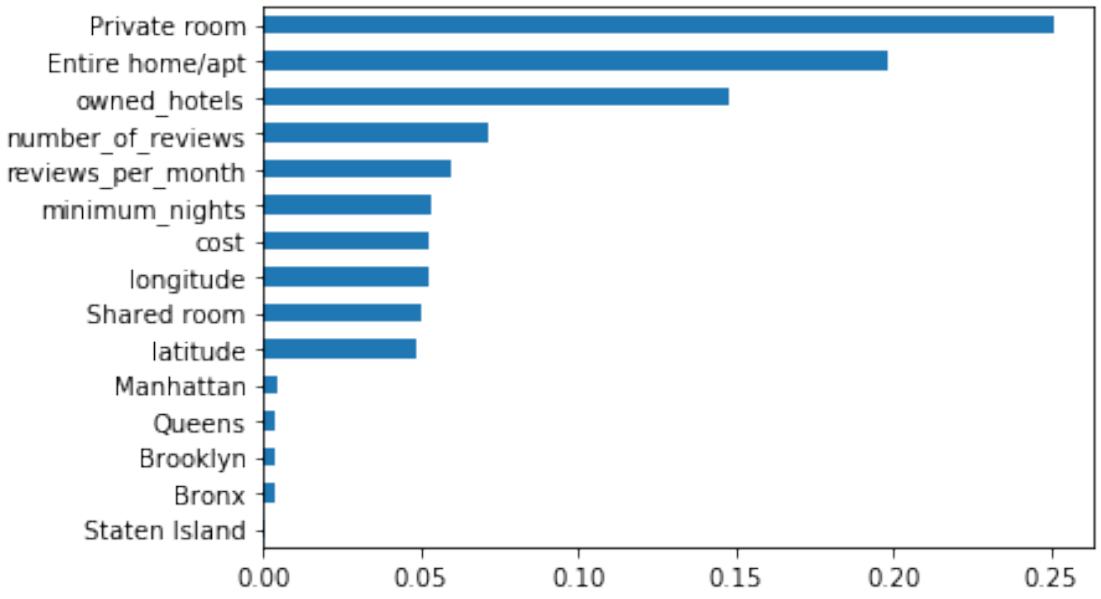
[140]:
```
gv.best_params_
```

[140]:
```
{'n_estimators': 15}
```

[141]:
```
best_mode  = gv.best_estimator_
```

[142]:
```
pd.Series(best_mode.feature_importances_,index=x.columns).sort_values().plot.
↪barh()
```

[142]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e729ca610>
```



[143]:
```
y_pred = best_mode.predict(x_test)
```

[144]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
```

[145]:
```
confusion_matrix(y_test,y_pred)
```

[145]:
```
array([[381,  42],
       [ 25, 413]])
```

[146]:
```
accuracy_score(y_test,y_pred)
```

[146]:
```
0.9221835075493612
```

[ ]:
```

```

1.3 Visualization, Modeling, Machine Learning

Build a model that categorizes hotels on the basis of their yearly availability. Identify how different features influence the decision. Please explain the findings effectively to technical and non-technical audiences using comments and visualizations, if appropriate. - **Build an optimized model that effectively solves the business problem. - The model will be evaluated on the basis of Accuracy. - Read the test.csv file and prepare features for testing.**

[147]:
```
#Loading Test data
test_data=pd.read_csv('test.csv')
test_data.head()
```

[147]:

	id	region	latitude	longitude	accommodation_type	cost	minimum_nights	number_of_reviews	reviews_per_month	owner_id	owned_hotels
0	19215	Brooklyn	40.70912	-73.94513	Shared room	135	2	22	0.66	4360212	1
1	36301	Brooklyn	40.57646	-73.96641	Entire home/apt	69	2	8	0.90	181356989	2
2	40566	Manhattan	40.76616	-73.98228	Private room	225	30	0	NaN	13773574	
3	33694	Manhattan	40.77668	-73.94587	Shared room	125	30	9	0.82	6788748	
4	28873	Manhattan	40.80279	-73.94450	Entire home/apt	43	1	13	0.72	105061915	

```
2            12
3             1
4             2
```

[148]:
```
id_row  = test_data["id"]
```

[149]:
```
test_data.drop(['id',"owner_id"],axis=1,inplace=True)
```

[150]:
```
from sklearn.preprocessing import LabelBinarizer
import numpy as np


for col in cat_col:
    if col != 'yearly_availability':
        try:
            lb = encoder[col]
            encoder_data = lb.transform(test_data[col])
            encoder_data = pd.DataFrame(encoder_data,columns=lb.classes_)
            test_data.drop([col],axis=1,inplace = True)
            test_data = pd.concat([test_data,encoder_data],axis=1)
        except Exception as er:
            print(er)
            print(col)
```

[152]:
```
test_data.isna().sum()
```

[152]:
```
latitude               0
longitude              0
cost                   0
minimum_nights         0
number_of_reviews      0
reviews_per_month    173
owned_hotels           0
Bronx                  0
Brooklyn               0
Manhattan              0
Queens                 0
Staten Island          0
Entire home/apt        0
Private room           0
Shared room            0
dtype: int64
```

[154]:
```
test_data["reviews_per_month"] = test_data["reviews_per_month"].
↪fillna(data["reviews_per_month"].median())
```

[155]:
```
test_pred = best_mode.predict(test_data)
```

[157]:

Highlight the most important features of the model for management.

Task:

- **Visualize the top 20 features and their feature importance.**

In the hotel management,I have obtained the top 20 features which are namely private room,entire home/apt,owned_hotels,number_of_reviews,reviews_per_month,minimum_nights,cost,longitude,shared room,lattitude,manhattan,queens,brooklyn,bronx,staten island. I noticed that the private room is the most critical feature in real time and our model also predicted the same.Usually if there is no private room then people will not book the hotel.

[ ]:

Task:

- **Submit the predictions on the test dataset using your optimized model** For each record in the test set (`test.csv`), predict the value of the `yearly_availability` variable. Submit a CSV file with a header row and one row per test entry.

The file (`submissions.csv`) should have exactly 2 columns: - **id** - **yearly_availability**

[ ]:
```
submission_df = pd.DataFrame()
submission_df["id"]  = id_row
submission_df["yearly_availability"] = test_pred
submission_df.to_csv("submissions.csv",index=False)
```

[ ]:

[ ]:
```
#Submission
#submission_df.to_csv('submissions.csv',index=False)
```
