Random forest usimg grid search

Methodology

- **Data Cleaning:** Checking for null values and based on their number either dropping them or replacing with mean, median, mode based on the type and description of data. Also converting the data types of values in correct format, like for price the given type is object so need to convert this to float.
- **Data Visualization:** This step helps understand the understand the data in a visually. We can understand normality of the data as well. This helps us to decide whether to normalize the data.
- **Feature Selection:** Based on the Pearson correlation between the labeled column and rest of the features. In general, a very great correlation should have an absolute value greater than 0.75. When the labeled column is depended on multiple columns, the correlation with one column may be less. But combined features may have higher effect.
- **Train Test Split:** We split the data into 80:20 ratio for tarining testing respectively.
- **Model Selection:** Based on the data visualization and data correlation, we need to select a model that would best suit. Here we need to use Rnadom forest with grid search.
- **Evalution:** In this case we are using RMSE, R2 Score to determine the accuracy of the predicting model.

importing libraries

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

Reading data

```
df=pd.read_json("amsterdam.json")
```

```
df.dtypes
```

```
host_listings_count               float64
accommodates                        int64
bathrooms                         float64
bedrooms                          float64
guests_included                     int64
minimum_nights                      int64
number_of_reviews                   int64
calculated_host_listings_count      int64
price                              object
latitude                          float64
longitude                         float64
room_type                          object
instant_bookable                   object
dtype: object
```

converting price to Float dtype

```
l=[]
for i in df.price:
  i = i. replace(",", "")
  i= i. replace("$", "")
  l.append(i)
```

```
df.price = l
df.price=pd.to_numeric(df.price)
```

Handling Object dtypes

```
df.dtypes
```

```
df.dtypes
```

```
host_listings_count               float64
accommodates                        int64
bathrooms                         float64
bedrooms                          float64
guests_included                     int64
minimum_nights                      int64
number_of_reviews                   int64
calculated_host_listings_count      int64
price                             float64
latitude                          float64
longitude                         float64
room_type                          object
instant_bookable                   object
dtype: object
```

```
c=df.room_type.unique()
for i in range(len(c)):
  df.room_type=df.room_type.replace(c[i],i+1)
df.room_type.unique()
```

```
array([1, 2, 3])
```

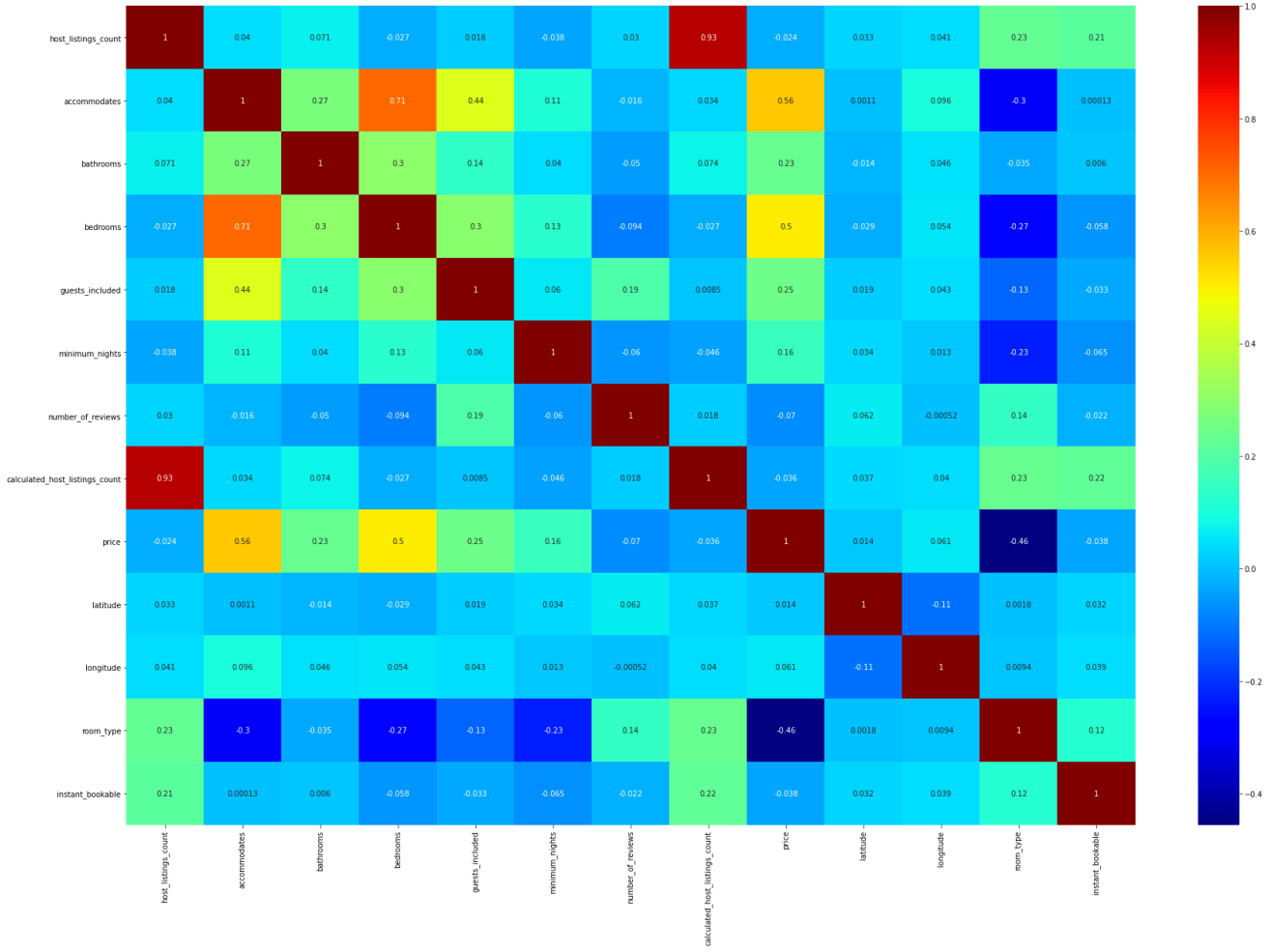
```
c=df.instant_bookable.unique()
for i in range(len(c)):
  df.instant_bookable=df.instant_bookable.replace(c[i],i+1)
df.instant_bookable.unique()
```

```
array([1, 2])
```

Correlation

```
import seaborn as sns
plt.figure(figsize=(30,20))
sns.heatmap(df.corr(method="spearman"),annot = True,cmap="jet")
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is dep
  import pandas.util.testing as tm
<matplotlib.axes._subplots.AxesSubplot at 0x7f12a52ec518>
```



```
df.isna().sum()
```

```
host_listings_count                3
accommodates                       0
bathrooms                         18
bedrooms                          12
guests_included                    0
minimum_nights                     0
number_of_reviews                  0
calculated_host_listings_count     0
price                              0
latitude                           0
longitude                          0
room_type                          0
instant_bookable                   0
dtype: int64
```

```
df=df.dropna()
```

Model

```
cols = [col for col in df.columns if col not in ["price"]]
X = df[cols]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, df['price'], test_size=0.25)
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
para = {"max_depth":(list(range(13,30,1))),"bootstrap":[True, False],"max_features":["auto","log2","sqrt"]}
rfc = RandomForestRegressor()
clf = GridSearchCV(rfc, para)
clf.fit(X_train, y_train)
```

```
GridSearchCV(cv=None, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=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=100, n_jobs=None,
                                             oob_score=False, random_state=None,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'bootstrap': [True, False],
                         'max_depth': [13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
                                       23, 24, 25, 26, 27, 28, 29],
                         'max_features': ['auto', 'log2', 'sqrt']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

Evaluation

```
clf.score(X_train,y_train)
```

```
0.8716104597747697
```

```
clf.score(X_test,y_test)
```

```
0.4886065278971329
```

```
clf.best_params_
```

```
{'bootstrap': True, 'max_depth': 19, 'max_features': 'sqrt'}
```

```
clf.best_score_
```

```
0.4624055454706914
```