AirBnb listings file modeling

In [1]:

```
# Put these at the top of every notebook, to get automatic reloading and inl
from IPython.core.display import display, HTML
import pandas as pd
import warnings
import ast
warnings.filterwarnings('ignore')

%reload_ext autoreload
%autoreload 1
%matplotlib inline

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)

display(HTML("<style>.container { width:100% !important; }</style>"))
```

In [1]:

```
import os
import seaborn as sns
import pandas as pd
import math

import sklearn.model_selection as cv

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, Gradi
from sklearn.model_selection import GridSearchCV

from sklearn.metrics import mean_squared_error as MSE

from imblearn.over_sampling import SMOTE

from Utils.UtilsGeoViz import *
from Utils.UtilsViz import *
from Utils.DataUtils import *

RANDOM_SEED = 42
```

In [3]:

```
data_path = os.path.join(os.getcwd(), "../data/cleaned_listings.csv")
listings = pd.read_csv(data_path, index_col="id")
display(listings.shape)
```

```
(48855, 65)
```

Plot the dstribution

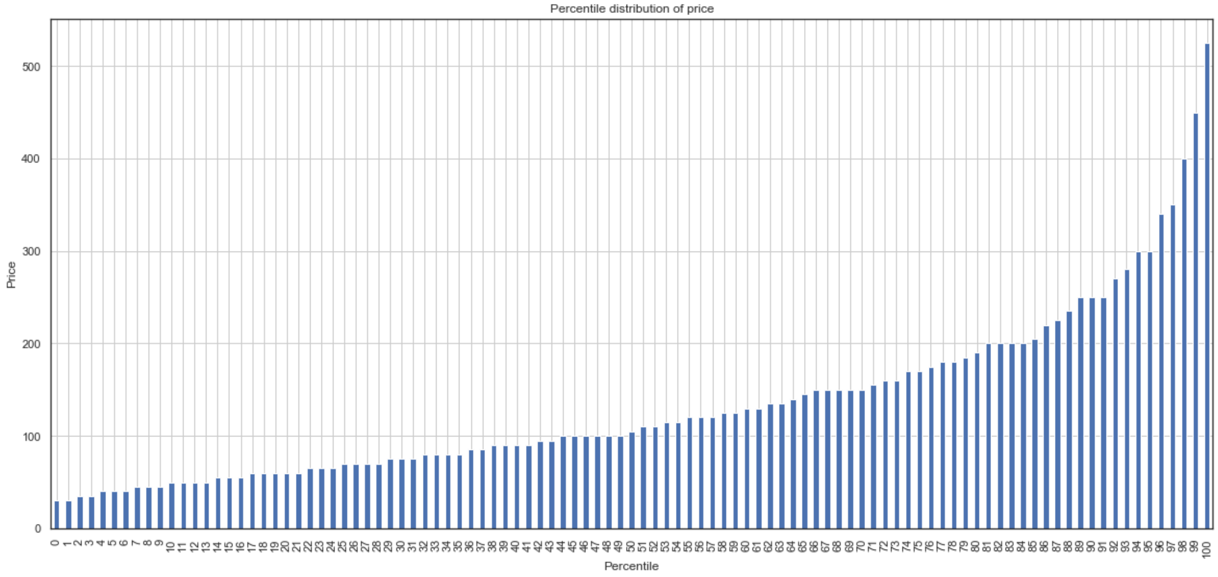
Let's plot the percentile for price

In [4]:

```python
percentiles = list(range(0,101, 1))
price_percentile = {}
for p in percentiles:
    price_percentile[p] = np.percentile(listings['price'].values, p)

sns.set(style="white")
price_percentile = pd.DataFrame.from_dict(price_percentile, orient='index')
price_percentile.plot(kind='bar', figsize=(20,9), grid=True, legend=False)
plt.title("Percentile distribution of price")
plt.xlabel("Percentile")
plt.ylabel("Price")
```

Out[4]:

```
Text(0, 0.5, 'Price')
```

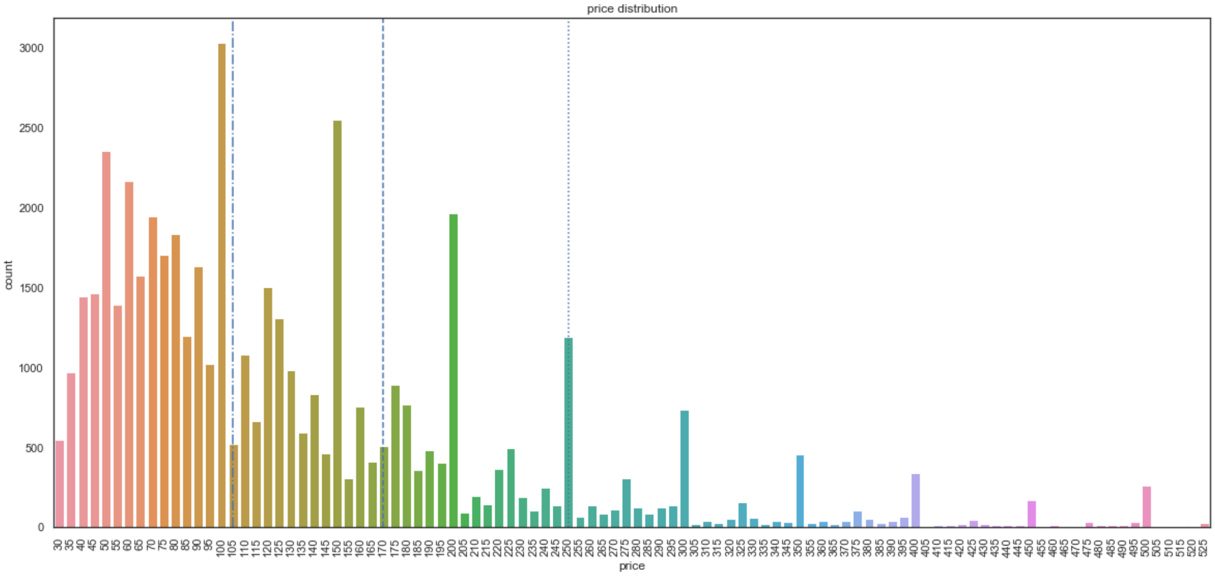


In [5]:

```
f, ax = plt.subplots(1,1,figsize=(20,9))
g = sns.countplot(x="price", data=listings, ax=ax)
t = g.set_xticklabels(g.get_xticklabels(), rotation=90)
t = g.set_title("price distribution")
median_idx = np.where(np.sort(listings["price"].unique())==listings["price"]
plt.axvline(x=median_idx, linestyle="-.")
percentile_75_idx = np.where(np.sort(listings["price"].unique())==price_perc
plt.axvline(x=percentile_75_idx, linestyle="--")
percentile_90_idx = np.where(np.sort(listings["price"].unique())==price_perc
plt.axvline(x=percentile_90_idx, linestyle=":")
```

Out[5]:

```
<matplotlib.lines.Line2D at 0x12b4d870>
```



Quick helper functions

In [6]:

```
def roundto(row, base=5):
    return int(base * round(float(row) / base))

# Get the index of the price columns
def get_index(vallist, val):
    return vallist.index(val)
```

1. Oversampling using SMOTE

In [7]:

```
def check_rep(row):
    if (row <= 200) | (row==250) | (row==350) | (row==450) | (row==550) :
        return 0
    elif (row > 200) & (row < 300) & (row != 250):
        return 1
    elif (row > 300) & (row < 400) & (row != 350):
        return 2
    else:
        return 3

listings["flag_ur"] = listings["price"].apply(check_rep)
```

In [8]:

```
vcs = listings["flag_ur"].value_counts()
vcs
```

Out[8]:

```
0    43321
1     3093
3     1624
2      817
Name: flag_ur, dtype: int64
```

In [9]:

```
ycol = ["flag_ur"]
xcol = [i for i in listings.columns if i not in ycol]

x = listings[xcol].values
y = listings[ycol].values

smote_sampling_strategy = {
    1: int(vcs[1]*2)
    ,2: int(vcs[2]*2)
    ,3: int(vcs[3]*2)
}
sm = SMOTE(random_state=RANDOM_SEED, sampling_strategy=smote_sampling_strate
# Fit the smote onto the sample
x_new, y_new = sm.fit_sample(x, y)

# Drop the flag column
listings.drop(labels=["flag_ur"], axis=1, inplace=True)

# ----------------------------------------------------------------------------
# Overwrite X and Y
price_index = get_index(list(listings.columns), "price")

y = x_new[:, price_index]
x = np.delete(x_new, price_index, axis=1)
for i in range(len(y)):
    y[i] = roundto(y[i])
```

```
c:\users\sriharis\appdata\local\programs\python\python37-32\lib\site-packages\s
klearn\utils\validation.py:761: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  y = column_or_1d(y, warn=True)
```

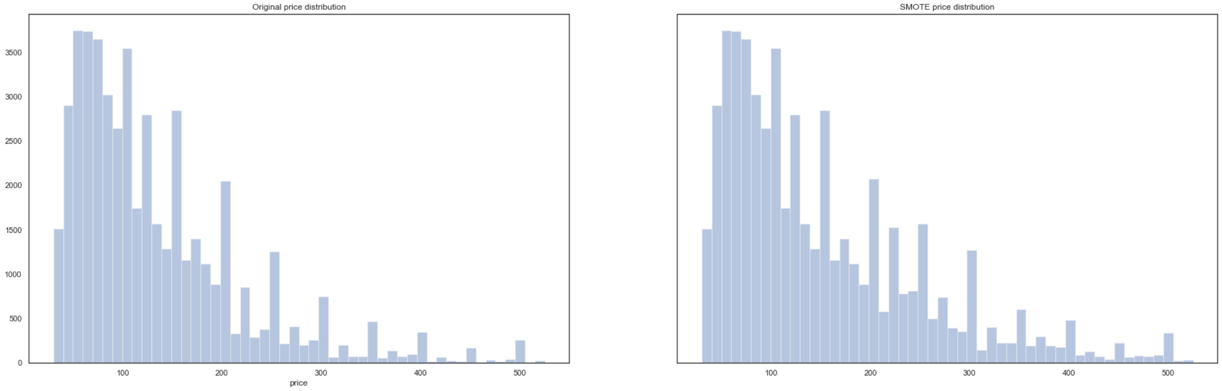
In [10]:

```
print(
    " Old size :", listings.shape, "\n",
    "New size :", x.shape
)
```

```
Old size : (48855, 65)
New size : (54389, 64)
```

In [11]:

```
f, ax = plt.subplots(1, 2, figsize=(30, 9), sharey=True)
g1 = sns.distplot(listings["price"], ax=ax[0], kde=False)
t = g1.set_title("Original price distribution")

g2 = sns.distplot(y, ax=ax[1], kde=False)
t = g2.set_title("SMOTE price distribution")

```



2. Transformation

In [12]:

```
x_cols = listings.drop(['price'], axis=1)
X = pd.DataFrame(data=x, columns=x_cols.columns)
```

In [13]:

```
X_log = X.copy()
X_sqr = X.copy()
X_sqrt = X.copy()
```

In [14]:

```
# Taking log, square and square root transformations of x
for i in range(X_log.shape[1]):
    X_log.iloc[:,i] = np.log(X_log.iloc[:,i] + 1)
    X_sqr.iloc[:,i] = np.square(X_sqr.iloc[:,i])
    X_sqrt.iloc[:,i] = np.sqrt(X_sqrt.iloc[:,i])
X_log.columns = X_log.columns.map(lambda x: x + '_log')
X_sqr.columns = X_sqr.columns.map(lambda x: x + '_sqr')
X_sqrt.columns = X_sqrt.columns.map(lambda x: x + '_sqrt')
```

In [15]:

```
X_sqrt.shape
```

Out[15]:

```
(54389, 64)
```

In [16]:

```
# Appending X, X_log, X_sqr and X_sqrt
X_final = pd.concat([X, X_log, X_sqr, X_sqrt], axis=1)
X_final.shape
```

Out[16]:

```
(54389, 256)
```

Prediction

In [17]:

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ra
```

In [18]:

```
rfr = RandomForestRegressor(random_state=RANDOM_SEED)
rfr.fit(X=x_train, y=y_train)
y_pred_train = rfr.predict(X=x_train)
y_pred_test = rfr.predict(X=x_test)

mse_train = MSE(y_train, y_pred_train)
mse_test = MSE(y_test, y_pred_test)

rmse_train = mse_train**(1/2)
rmse_test = mse_test**(1/2)

print("Train set RMSE with Transformation: {:.2f}".format(rmse_train))
print("Test set RMSE with Transformation: {:.2f}".format(rmse_test))
```

```
c:\users\sriharis\appdata\local\programs\python\python37-32\lib\site-packages\s
klearn\ensemble\forest.py:246: 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)

Train set RMSE with Transformation: 24.25
Test set RMSE with Transformation: 56.60
```

In [19]:

```
# Use cross-validation on Train data

CV_scores = - cv.cross_val_score(rfr, x_train, y_train, scoring='neg_mean_sq

# Compute the 10-folds CV
CV = CV_scores.mean()**(1/2)

# Print Train CV accuracy
print('Cross Validation RMSE with Transformation: {:.2f}'.format(CV))
```

```
Cross Validation RMSE with Transformation: 57.38
```

3. Train test split

In [20]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, ra
```

4. Standardisation

In [21]:

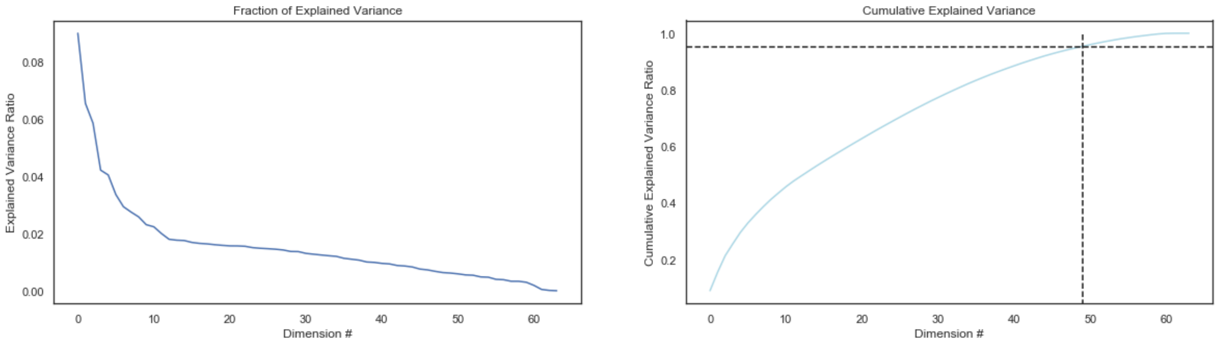
```
standard_scaler = StandardScaler()
x_train = standard_scaler.fit_transform(x_train)
x_test = standard_scaler.transform(x_test)
```

5. PCA

In [22]:

```
pca_naive = PCA(n_components=None)
pca_naive.fit(x_train)
f, ax = plt.subplots(1,2,figsize=(20,5))
plot_naive_variance(pca_naive, ax[0])
num_pc = plot_pca_var_cum(pca_naive, ax=ax[1], cutoff=0.95)
print("Number of components : ", num_pc)
```

```
c:\users\sriharis\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\cbook\__init__.py:1725: UserWarning: Saw kwargs ['c', 'color'] which
are all aliases for 'color'.  Kept value from 'color'
  seen=seen, canon=canonical, used=seen[-1]))

Number of components :  49
```



In [23]:

```
# pca = PCA(n_components=num_pc)
pca = PCA(n_components=None)
pca.fit(x_train)
x_train_pca = pca.transform(x_train)
x_test_pca = pca.transform(x_test)
```

Prediction

Random Forest Regressor

With PCA

In [24]:

```
rfr = RandomForestRegressor(random_state=RANDOM_SEED)
rfr.fit(X=x_train_pca, y=y_train)
y_pred_train = rfr.predict(X=x_train_pca)
y_pred_test = rfr.predict(X=x_test_pca)

mse_train = MSE(y_train, y_pred_train)
mse_test = MSE(y_test, y_pred_test)

rmse_train = mse_train**(1/2)
rmse_test = mse_test**(1/2)

print("Train set RMSE with PCA: {:.2f}".format(rmse_train))
print("Test set RMSE with PCA: {:.2f}".format(rmse_test))
```

```
c:\users\sriharis\appdata\local\programs\python\python37-32\lib\site-packages\s
klearn\ensemble\forest.py:246: 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)

Train set RMSE with PCA: 26.72
Test set RMSE with PCA: 62.94
```

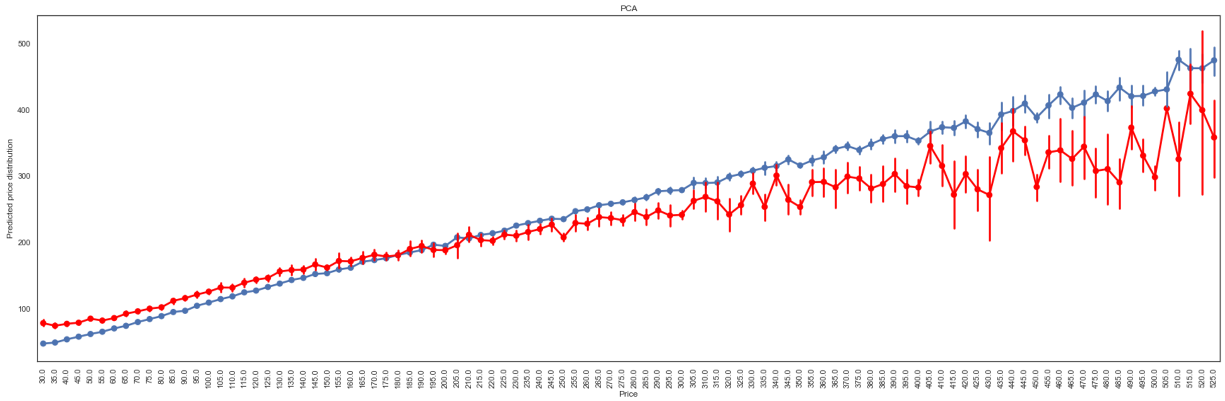
In [25]:

```
# Use cross-validation on Train data

CV_scores = - cv.cross_val_score(rfr, x_train_pca, y_train, scoring='neg_mea

# Compute the 10-folds CV
CV = CV_scores.mean()**(1/2)

# Print Train CV accuracy
print('Cross Validation RMSE with PCA: {:.2f}'.format(CV))
```

```
Cross Validation RMSE with PCA: 63.11
```

In [26]:

```
f, ax = plt.subplots(1,1, figsize=(30, 9))
g = sns.pointplot(x=y_train, y=y_pred_train, ax=ax)
g = sns.pointplot(x=y_test, y=y_pred_test, ax=ax, color="red")

t = g.set_xlabel("Price")
t = g.set_ylabel("Predicted price distribution")
t = g.set_xticklabels(g.get_xticklabels(), rotation=90)
t = g.set_title("PCA")
```


PCA
Predicted price distribution
Price