In [259]:
```
import pandas as pd
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
df=pd.read_csv("data.csv")
df.shape
```

Out[259]:
```
(1000, 27)
```

In [260]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 27 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   index            1000 non-null   int64  
 1   TID              1000 non-null   int64  
 2   breadcrumb       1000 non-null   object 
 3   category_name    1000 non-null   object 
 4   property_type    1000 non-null   object 
 5   building_size    280 non-null    object 
 6   land_size        533 non-null    object 
 7   preferred_size   609 non-null    object 
 8   open_date        302 non-null    object 
 9   listing_agency   1000 non-null   object 
 10  price            1000 non-null   object 
 11  location_number  1000 non-null   int64  
 12  location_type    1000 non-null   object 
 13  location_name    1000 non-null   object 
 14  address          988 non-null    object 
 15  address_1        988 non-null    object 
 16  city             1000 non-null   object 
 17  state            1000 non-null   object 
 18  zip_code         1000 non-null   int64  
 19  phone            1000 non-null   object 
 20  latitude         0 non-null      float64
 21  longitude        0 non-null      float64
 22  product_depth    1000 non-null   object 
 23  bedroom_count    967 non-null    float64
 24  bathroom_count   967 non-null    float64
 25  parking_count    967 non-null    float64
 26  RunDate          1000 non-null   object 
dtypes: float64(5), int64(4), object(18)
memory usage: 211.1+ KB
```

In [261]:
```
df
```

Out[261]:

	index	TID	breadcrumb	category_name	property_type	building_size	land_size	preferred_size	open_date	listing_agency	...	state
0	0	1350988	Buy>NT>DARWIN CITY	Real Estate & Property for sale in DARWIN CITY...	House	NaN	NaN	NaN	Added 2 hours ago	Professionals - DARWIN CITY	...	NT
1	1	1350989	Buy>NT>DARWIN CITY	Real Estate & Property for sale in DARWIN CITY...	Apartment	171m²	NaN	171m²	Added 7 hours ago	Nick Mousellis Real Estate - Eview Group Member	...	NT
2	2	1350990	Buy>NT>DARWIN CITY	Real Estate & Property for sale in DARWIN CITY...	Unit	NaN	NaN	NaN	Added 22 hours ago	Habitat Real Estate - THE GARDENS	...	NT
3	3	1350991	Buy>NT>DARWIN CITY	Real Estate & Property for sale in DARWIN CITY...	House	NaN	NaN	NaN	Added yesterday	Ray White - NIGHTCLIFF	...	NT
4	4	1350992	Buy>NT>DARWIN CITY	Real Estate & Property for sale in DARWIN CITY...	Unit	201m²	NaN	201m²	Added yesterday	Carol Need Real Estate - Fannie Bay	...	NT
...	...	...	...	...	...	...	...	...	...	...	...	...
995	995	1351983	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	House	NaN	9.17ha	9.17ha	Under offer	United Realty NT - Parap	...	NT
996	996	1351984	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	House	203m²	600m²	600m²	NaN	Kassiou Constructions - HOWARD SPRINGS	...	NT
997	997	1351985	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	House	209.6m²	800m²	800m²	NaN	Kassiou Constructions - HOWARD SPRINGS	...	NT

	index	TID	breadcrumb	category_name	property_type	building_size	land_size	preferred_size	open_date	listing_agency	...	state
998	998	1351986	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	House	180m²	450m²	450m²	NaN	Kassiou Constructions - HOWARD SPRINGS	...	NT
999	999	1351987	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT	Unit	120m²	NaN	120m²	NaN	Home Zone NT - DARWIN	...	NT

In [262]:
```
df.tail()
```

Out[262]:

	index	TID	breadcrumb	category_name	property_type	building_size	land_size	preferred_size	open_date	listing_agency	...	state
995	995	1351983	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	House	NaN	9.17ha	9.17ha	Under offer	United Realty NT - Parap	...	NT
996	996	1351984	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	House	203m²	600m²	600m²	NaN	Kassiou Constructions - HOWARD SPRINGS	...	NT
997	997	1351985	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	House	209.6m²	800m²	800m²	NaN	Kassiou Constructions - HOWARD SPRINGS	...	NT
998	998	1351986	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	House	180m²	450m²	450m²	NaN	Kassiou Constructions - HOWARD SPRINGS	...	NT
999	999	1351987	Buy>NT>DARWIN	Real Estate & Property for sale in DARWIN, NT ...	Unit	120m²	NaN	120m²	NaN	Home Zone NT - DARWIN	...	NT

5 rows × 27 columns

In [263]:
```
df.describe()
```

Out[263]:

	index	TID	location_number	zip_code	latitude	longitude	bedroom_count	bathroom_count	parking_count
count	1000.000000	1.000000e+03	1.000000e+03	1000.00000	0.0	0.0	967.000000	967.000000	967.000000
mean	499.500000	1.351488e+06	1.474125e+08	816.64600	NaN	NaN	2.866598	1.739400	2.152017
std	288.819436	2.888194e+02	6.121381e+07	13.22057	NaN	NaN	1.151914	0.635663	1.514818
min	0.000000	1.350988e+06	1.085305e+08	800.00000	NaN	NaN	0.000000	1.000000	0.000000
25%	249.750000	1.351238e+06	1.386598e+08	800.00000	NaN	NaN	2.000000	1.000000	1.000000
50%	499.500000	1.351488e+06	1.390458e+08	820.00000	NaN	NaN	3.000000	2.000000	2.000000
75%	749.250000	1.351737e+06	1.393042e+08	830.00000	NaN	NaN	4.000000	2.000000	2.000000
max	999.000000	1.351987e+06	7.001996e+08	839.00000	NaN	NaN	9.000000	5.000000	12.000000

In [264]:
```
df.isna().sum()
```

Out[264]:
```
index                0
TID                  0
breadcrumb           0
category_name        0
property_type        0
building_size      720
land_size          467
preferred_size     391
open_date          698
listing_agency       0
price                0
location_number      0
location_type        0
location_name        0
address             12
address_1           12
city                 0
state                0
zip_code             0
phone                0
latitude          1000
longitude         1000
product_depth        0
bedroom_count       33
bathroom_count      33
parking_count       33
RunDate              0
dtype: int64
```

In [265]:
```
df2=df.drop(['index','TID','latitude','longitude'],axis=1)
```

In [266]:
```
df2.isna().sum()
```

Out[266]:
```
breadcrumb           0
category_name        0
property_type        0
building_size      720
land_size          467
preferred_size     391
open_date          698
listing_agency       0
price                0
location_number      0
location_type        0
location_name        0
address             12
address_1           12
city                 0
state                0
zip_code             0
phone                0
product_depth        0
bedroom_count       33
bathroom_count      33
parking_count       33
RunDate              0
dtype: int64
```

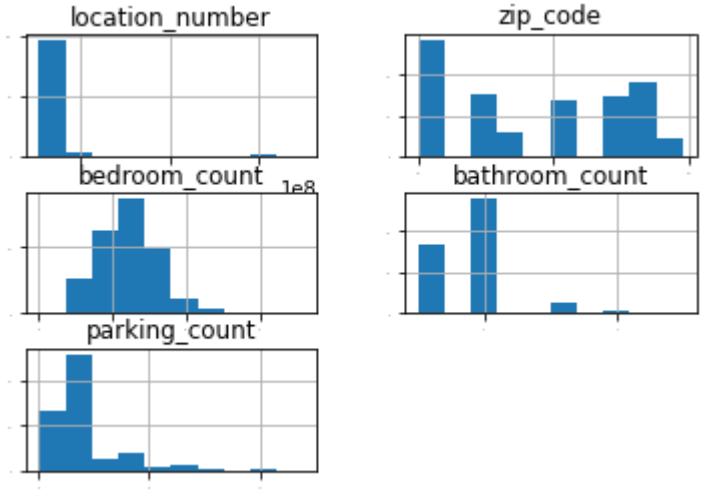
In [267]:
```
df2.corr()
```

Out[267]:

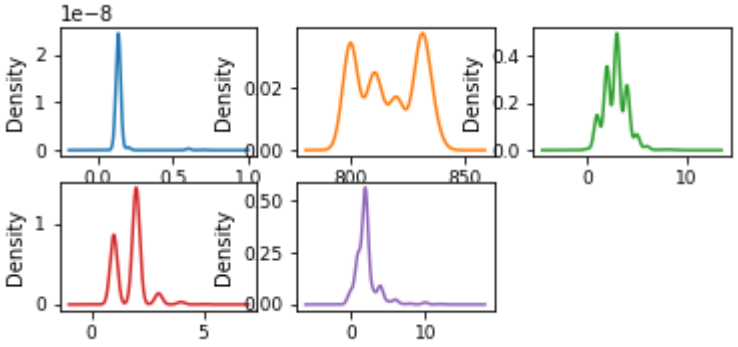
	location_number	zip_code	bedroom_count	bathroom_count	parking_count
location_number	1.000000	0.121483	0.072712	0.033797	0.074206
zip_code	0.121483	1.000000	0.439478	0.118571	0.356923
bedroom_count	0.072712	0.439478	1.000000	0.622596	0.512936
bathroom_count	0.033797	0.118571	0.622596	1.000000	0.389506
parking_count	0.074206	0.356923	0.512936	0.389506	1.000000

In [268]:
```
from matplotlib import pyplot as plt
```

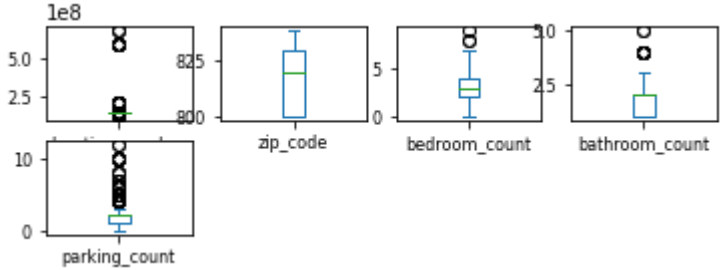
In [269]:
```
df2.hist(sharex=False,sharey=False,xlabelsize=1,ylabelsize=1)
plt.show()
```



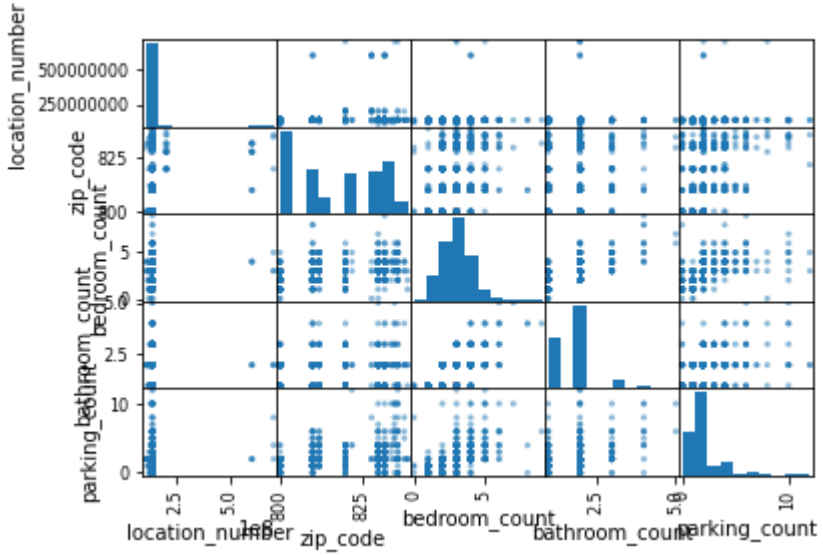
In [270]:
```
df2.plot(kind='density',subplots=True,layout=(3,3),sharex=False,sharey=False,legend=False,fontsize=9)
plt.show()
```



In [271]:
```
df2.plot(kind='box',subplots=True,layout=(4,4),sharex=False,sharey=False,fontsize=8)
plt.show()
```



In [272]:
```
from pandas.plotting import scatter_matrix
scatter_matrix(df2)
plt.show()
```



In [273]:
```
import seaborn as sbn
```

In [274]:
```
sbn.heatmap(df2.corr(),cmap='PuBu',annot=True)
```

Out[274]: <AxesSubplot:>

In [275]:
```
df['land_size'].value_counts()
```

Out[275]:
```
2.02ha      14
800m²       13
2ha         11
450m²       10
817m²        7
            ..
185m²        1
421m²        1
2,510m²      1
612m²        1
600m²        1
Name: land_size, Length: 346, dtype: int64
```

In [276]:
```
# df['land_size']=df['land_size'].str.replace('[m² ha]','')
# df['building_size']=df['building_size'].str.replace('[m²','')
```

In [277]:
```
# dff=df2.loc[:,['property_type','zip_code','location_number','product_depth','bedroom_count','bathroom_cou
# dff
```

In [278]:
```
regex='[\w]'
df2['price_s']=df2['price'].str.replace('[,$]','')
df2['price_s']
```

Out[278]:
```
0                  435000
1      Offers Over 320000
2                  310000
3                  259000
4                  439000
               ...
995           2 Residence
996                601000
997                655000
998                675000
999                399000
Name: price_s, Length: 1000, dtype: object
```

In [279]:
```
import numpy as np
df2['price_ss']=np.where(df2['price_s'].str.isnumeric(),df2['price_s'],0)
df2['price_ss']
```

Out[279]:
```
0      435000
1           0
2      310000
3      259000
4      439000
        ...
995         0
996    601000
997    655000
998    675000
999    399000
Name: price_ss, Length: 1000, dtype: object
```

In [280]:
```
dff2
```

Out[280]:

	breadcrumb	category_name	property_type	building_size	land_size	preferred_size	open_date	listing_agency	location_number	location_typ
0	1	0	4	32	66	82	2	56	855	
1	1	0	1	41	-1	57	11	49	849	
2	1	0	10	-1	-1	-1	3	35	847	
3	1	0	4	-1	-1	-1	13	63	833	
4	1	0	10	69	-1	90	13	12	815	
...	...	...	...	...	...	...	...	...	...	
995	0	1	4	-1	325	350	14	78	107	
996	0	1	4	70	188	207	-1	41	106	
997	0	1	4	74	262	284	-1	41	105	
998	0	1	4	48	160	175	-1	41	104	
999	0	1	10	12	-1	30	-1	37	103	

1000 rows × 23 columns

In [281]:
```
dff2.isna().sum()
```

Out[281]:
```
breadcrumb          0
category_name       0
property_type       0
building_size       0
land_size           0
preferred_size      0
open_date           0
listing_agency      0
location_number     0
location_type       0
location_name       0
address             0
address_1           0
city                0
state               0
zip_code            0
phone               0
product_depth       0
bedroom_count       0
bathroom_count      0
parking_count       0
RunDate             0
price_ss            0
dtype: int64
```

In [282]:
```
dff2.describe()
```

Out[282]:

	breadcrumb	category_name	property_type	building_size	land_size	preferred_size	open_date	listing_agency	location_number	locati
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	
mean	0.184000	0.816000	5.019000	22.255000	93.250000	115.100000	2.834000	42.074000	447.532000	
std	0.387678	0.387678	3.302051	45.706488	114.181009	124.634497	6.305569	22.999641	256.480321	
min	0.000000	0.000000	0.000000	-1.000000	-1.000000	-1.000000	-1.000000	0.000000	0.000000	
25%	0.000000	1.000000	4.000000	-1.000000	-1.000000	-1.000000	-1.000000	23.000000	229.750000	
50%	0.000000	1.000000	4.000000	-1.000000	26.000000	70.500000	-1.000000	43.000000	444.500000	
75%	0.000000	1.000000	9.000000	19.250000	193.250000	230.250000	6.000000	66.000000	670.250000	
max	1.000000	1.000000	12.000000	168.000000	345.000000	375.000000	14.000000	84.000000	888.000000	

8 rows × 23 columns

In [283]:
```
dff2['bathroom_count'].fillna(dff2['bathroom_count'].median(),inplace=True)
```

In [284]:
```
dff2['bedroom_count'].fillna(dff2['bedroom_count'].median(),inplace=True)
dff2['parking_count'].fillna(dff2['parking_count'].median(),inplace=True)
dff2['building_size'].fillna(dff2['building_size'].mode(),inplace=True)

dff2['land_size'].fillna(dff2['land_size'].mode(),inplace=True)
dff2['preferred_size'].fillna(dff2['preferred_size'].mode(),inplace=True)
```

In [285]:
```
dff2.isna().sum()
```

Out[285]:
```
breadcrumb         0
category_name      0
property_type      0
building_size      0
land_size          0
preferred_size     0
open_date          0
listing_agency     0
location_number    0
location_type      0
location_name      0
address            0
address_1          0
city               0
state              0
zip_code           0
phone              0
product_depth      0
bedroom_count      0
bathroom_count     0
parking_count      0
RunDate            0
price_ss           0
dtype: int64
```

In [286]:
```
# dff3=pd.get_dummies(dff2,columns=['property_type','product_depth'])
```

In [287]:
```
# dff3
```

In [288]:
```
from sklearn.preprocessing import StandardScaler,MinMaxScaler
```

In [289]:
```
dff2['price_ss']=dff2['price_ss'].astype(int)
```

In [290]:
```
dff2.isna().sum()
```

Out[290]:
```
breadcrumb          0
category_name       0
property_type       0
building_size       0
land_size           0
preferred_size      0
open_date           0
listing_agency      0
location_number     0
location_type       0
location_name       0
address             0
address_1           0
city                0
state               0
zip_code            0
phone               0
product_depth       0
bedroom_count       0
bathroom_count      0
parking_count       0
RunDate             0
price_ss            0
dtype: int64
```

In [291]:
```
# dff3=dff2.loc[:,['property_type','zip_code','location_number','product_depth','bedroom_count','bathroom_c
# dff
dff3=dff2
```

In [292]:
```
dff3.shape
```

Out[292]:
```
(1000, 23)
```

In [293]:
```
#Handle categorical data
for i in dff3.columns.tolist():
    dff3[i]=dff3[i].astype('category').cat.codes
```

In [294]:
```
dff3.head()
```

Out[294]:

	breadcrumb	category_name	property_type	building_size	land_size	preferred_size	open_date	listing_agency	location_number	location_type
0	1	0	4	33	67	83	3	56	855	0
1	1	0	1	42	0	58	12	49	849	0
2	1	0	10	0	0	0	4	35	847	0
3	1	0	4	0	0	0	14	63	833	0
4	1	0	10	70	0	91	14	12	815	0

5 rows × 23 columns

In [295]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.metrics import r2_score
X=dff3.drop('price_ss', axis=1)
y=dff3['price_ss']
X_train, X_test, y_train, y_test= train_test_split(X,y,test_size=0.30, random_state=0)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(700, 22)
(700,)
(300, 22)
(300,)
```

In [296]:
```
#Linear Regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,accuracy_score,r2_score
regressor=LinearRegression()
model=regressor.fit(X_train, y_train) # Generates the model
```

In [297]:
```
lr=LinearRegression()
lr.fit(X_train, y_train)
lr.score(X_test, y_test)
y_pred=lr.predict(X_test)
mse=mean_squared_error(y_pred,y_test)
r2=r2_score(y_pred,y_test)
```

In [298]:
```
print("MAE:", metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 30.291234329017183
MSE: 1510.0888897073871
RMSE: 38.859862193623215
```

In [299]:
```
#Predict the model
y_pred=regressor.predict(X_test)
```

In [300]:
```
#Gradient boosting
import statsmodels.api as sm
from sklearn import ensemble
clf=ensemble.GradientBoostingRegressor(n_estimators=400, max_depth=5, min_samples_split=2, learning_rate=0.1
clf.fit(X_train, y_train)

clf.score(X_test, y_test)
```

Out[300]:
```
0.786975237802058
```

In [301]:
```
# Logistic regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_lr = logreg.predict(X_test)

log_train = round(logreg.score(X_train, y_train) * 100, 2)
log_accuracy_MinMax = round(accuracy_score(y_pred_lr, y_test) * 100, 2)

print("Training Accuracy    :",log_train ,"%")
print("Model Accuracy Score :",log_accuracy_MinMax ,"%")
```

```
Training Accuracy    : 66.0 %
Model Accuracy Score : 49.67 %

/home/vas/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [302]:
```
# SVC
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
y_pred_svc = svc.predict(X_test)

svc_train = round(svc.score(X_train, y_train) * 100, 2)
svc_accuracy_without_scaling = round(accuracy_score(y_pred_svc, y_test) * 100, 2)

print("Training Accuracy    :",svc_train ,"%")
print("Model Accuracy Score :",svc_accuracy_without_scaling ,"%")
```

```
Training Accuracy    : 56.0 %
Model Accuracy Score : 49.0 %
```