```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
df1 =pd.read_csv("bengaluru_house_prices.csv")
df1.head()
df1['area_type'].unique()
df1['area_type'].value_counts()
df2 = df1.drop(['area_type','society','balcony','availability'],axis='columns')
df2.shape
df2.isnull().sum()
df3 = df2.dropna()
df3.isnull().sum()
df3['bhk'] = df3['size'].apply(lambda x:int(x.split('')[0]))
df3.bhk.unique()
def is_float(x):
 try:
  float(x)
 except:
  return False
 return True
df3[~df3['total_sqft'].apply(is_float)].head(10)
def convert_sqft_to_num(x):
 tokens = x.split('-')
 if len(tokens) == 2:
  return (float(tokens[0])+float(tokens[1]))/2
 try:
  return float(x)
 except:
  return None
df4 = df3.copy()
df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
df4 = df4[df4.total_sqft.notnull()]
df4.head(2)
df5 = df4.copy()
df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
df5.head()
df5_stats = df5['price_per_sqft'].describe()
df5_stats
df5.to_csv("bhp.csv",index=False)
df5.location = df5.location.apply(lambda x: x.strip())
location_stats=df5['location'].value_counts(ascending=False)
location_stats
location_stats.values.sum()
len(location_stats[location_stats>10])
len(location_stats)
location_stats_less_than_10=location_stats[location_stats<=10]
location_stats_less_than_10
len(df5.location.unique())
df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 elsex)
len(df5.location.unique())
df5[df5.total_sqft/df5.bhk<300].head()
df6 = df5[~(df5.total_sqft/df5.bhk<300)]
```

```
df6.shape
df6.price_per_sqft.describe()
def remove_pps_outliers(df):
 df_out = pd.DataFrame()
 for key, subdf in df.groupby('location'):
  m = np.mean(subdf.price_per_sqft)
  st = np.std(subdf.price_per_sqft)
  reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st))]
  df_out = pd.concat([df_out,reduced_df],ignore_index=True)
 return df_out
df7 = remove_pps_outliers(df6)
df7.shape
def plot_scatter_chart(df,location):
 bhk2 = df[(df.location==location) & (df.bhk==2)]
 bhk3 = df[(df.location==location) & (df.bhk==3)]
 matplotlib.rcParams['figure.figsize'] = (15,10)
 plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
 plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', color='green',label='3 BHK', s=50)
 plt.xlabel("Total Square Feet Area")
 plt.ylabel("Price (Lakh Indian Rupees)")
 plt.title(location)
 plt.legend()
plot_scatter_chart(df7,"Rajaji Nagar")
plot_scatter_chart(df7,"Hebbal")
def remove_bhk_outliers(df):
 exclude_indices = np.array([])
 for location, location_df in df.groupby('location'):
  bhk_stats = {}
  for bhk, bhk_df in location_df.groupby('bhk'):
   bhk_stats[bhk] = {
    'mean': np.mean(bhk_df.price_per_sqft),
    'std': np.std(bhk_df.price_per_sqft),
    'count': bhk_df.shape[0]}
  for bhk, bhk_df in location_df.groupby('bhk'):
                    stats = bhk_stats.get(bhk-1)
       if stats and stats['count']>5:


         exclude_indices = np.append(exclude_indices, bhk_df[bhk_df.price_per_sqft<(stats['mean'])].index.
values)
 return df.drop(exclude_indices,axis='index') df8 = remove_bhk_outliers(df7)
# df8 = df7.copy()
df8.shape
plot_scatter_chart(df8,"Rajaji Nagar")
plot_scatter_chart(df8,"Hebbal")
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
plt.hist(df8.price_per_sqft,rwidth=0.8)
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
df8.bath.unique()
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
df8[df8.bath>10]
```

```
df8[df8.bath>df8.bhk+2]
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
df10=df9.drop(['size','price_per_sqft'],axis='columns')
df10.head(3)
dummies = pd.get_dummies(df10.location) dummies.head(3)
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
df12 = df11.drop('location',axis='columns')
df12.head(2)
X = df12.drop(['price'],axis='columns')
X.head(3)
from sklearn.model_selection import train_test_split
X_train, X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
cross_val_score(LinearRegression(), X, y, cv=cv)

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor

def find_best_model_using_gridsearchcv(X,y): algos = {
 'linear_regression' : {
 'model': LinearRegression(),
 'params': {
  'normalize': [True, False]
 }
},
'lasso': {
 'model': Lasso(), 'params': {
 'alpha': [1,2],
 'selection': ['random', 'cyclic']
 }
},
'decision_tree': {
 'model': DecisionTreeRegressor(), 'params': {
 'criterion' : ['mse','friedman_mse'],
 'splitter': ['best','random']
  }
 }
}
scores = []
cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
for algo_name, config in algos.items():
 gs = GridSearchCV(config['model'], config['params'], cv=cv, return_train_score=False)
 gs.fit(X,y)
 scores.append({
     'model ': algo_name,
```

```
  'best_score': gs.best_score_,
  'best_params': gs.best_params_
})

 return pd.DataFrame(scores,columns=['model','best_score','best_params']) find_best_model_using_grid
searchcv(X,y)
def predict_price(location,sqft,bath,bhk):
 loc_index = np.where(X.columns==location)[0][0]

 x = np.zeros(len(X.columns)) x[0] = sqft
 x[1] = bath x[2] = bhk
 if loc_index >= 0:
  x[loc_index] = 1
 return lr_clf.predict([x])[0]
predict_price('1st Phase JP Nagar',1000, 2, 2)
predict_price('1st Phase JP Nagar',1000, 3, 3)
```