In [67]:
```
df12
```

Out[67]:

	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vitt
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	0	0	0	
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	0	0	0	
2	1875.0	2.0	235.0	3	1	0	0	0	0	0	...	0	0	0	
3	1200.0	2.0	130.0	3	1	0	0	0	0	0	...	0	0	0	
4	1235.0	2.0	148.0	2	1	0	0	0	0	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
10233	1200.0	2.0	70.0	2	0	0	0	0	0	0	...	0	0	0	
10234	1800.0	1.0	200.0	1	0	0	0	0	0	0	...	0	0	0	
10237	1353.0	2.0	110.0	2	0	0	0	0	0	0	...	0	0	0	
10238	812.0	1.0	26.0	1	0	0	0	0	0	0	...	0	0	0	
10241	3600.0	5.0	400.0	4	0	0	0	0	0	0	...	0	0	0	

7239 rows × 244 columns

In [68]:
```
df12.shape
```

Out[68]:
```
(7239, 244)
```

In [69]:
```
X=df12.drop(["price"],axis="columns")
X.head(3)
```

Out[69]:

	total_sqft	bath	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasa
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	0	0	0	
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	0	0	0	
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	

3 rows × 243 columns

In [70]:
```
y=df12.price
```

In [71]:
```
y.head()
```

Out[71]:
```
0    428.0
1    194.0
2    235.0
3    130.0
4    148.0
Name: price, dtype: float64
```

In [72]:
```
len(y)
```

Out[72]:
```
7239
```

In [73]:
```
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)
```

In [74]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

Out[74]:
```
0.8629132245229443
```

Use K Fold cross validation to measure accuracy of our LinearRegression model

In [75]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit

cv= ShuffleSplit(n_splits=9,test_size=0.2,random_state=0)
cross_val_score(LinearRegression(),X,y,cv=cv)
```

Out[75]:
```
array([0.82702546, 0.86027005, 0.85322178, 0.8436466 , 0.85481502,
       0.7996843 , 0.8546295 , 0.84180048, 0.79241964])
```

In [76]:
```
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_gridsearchcv(X,y)
```

Out[76]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726738	{'alpha': 2, 'selection': 'cyclic'}
2	decision_tree	0.716838	{'criterion': 'friedman_mse', 'splitter': 'ran...

In [77]:
```
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]
```

In [78]:
```
df12.columns
```

Out[78]:
```
Index(['total_sqft', 'bath', 'price', 'bhk', '1st Block Jayanagar',
       '1st Phase JP Nagar', '2nd Phase Judicial Layout',
       '2nd Stage Nagarbhavi', '5th Block Hbr Layout', '5th Phase JP Nagar',
       ...
       'Vijayanagar', 'Vishveshwarya Layout', 'Vishwapriya Layout',
       'Vittasandra', 'Whitefield', 'Yelachenahalli', 'Yelahanka',
       'Yelahanka New Town', 'Yelenahalli', 'Yeshwanthpur'],
      dtype='object', length=244)
```

In [79]:
```
predict_price('Whitefield',5000,4,5)
```

Out[79]:
```
377.12579897851015
```

In [80]:
```
predict_price('Indira Nagar',2000,3,4)
```

Out[80]:
```
274.14277436167635
```

In [86]:
```
predict_price('Ulsoor',800,1,2)
```

Out[86]:
```
68.49165579594116
```

In [84]:
```
predict_price('Yelahanka',800,1,2)
```

Out[84]:
```
26.519683081756575
```

In [82]:
```
# import pickle
# with open('model.pickle','wb') as f:
#     pickle.dump(lr_clf,f)
```

In [83]:
```
# import json
# columns={'data_columns':[col.lower() for col in X.columns]
#         }
# with open("columns.json","w") as f:
#     f.write(json.dumps(columns))
```

In [ ]:
```
 
```