In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
print( "Type of california housing dataset:", type(housing))
```

```
Type of california housing dataset: <class 'sklearn.utils._bunch.Bunch'>
```

Load data

In [3]:

```
house_df = pd.DataFrame(housing['data'] )
house_df.columns = housing['feature_names']
house_df['PRICE']= housing['target']
house_df.head()
```

Out[3]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

In [4]:

```
from sklearn.model_selection  import train_test_split
from sklearn.utils import shuffle
```

In [5]:

```
y = house_df['PRICE']

# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(house_df.drop('PRICE'
                                                    y,
                                                    test_size=0.3, random
```

Defining Models

In [6]:

```
from sklearn.linear_model import LinearRegression, RidgeCV, LassoCV
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import (RandomForestRegressor, ExtraTreesRegressor,
                              AdaBoostRegressor, GradientBoostingRegresso
                              HistGradientBoostingRegressor)

from sklearn.metrics import r2_score
```

Default Training

In [7]:

```
models = {
    'linear': LinearRegression(),
    'ridge': RidgeCV(alphas = np.linspace(1e-3,10)),
    'lasso': LassoCV(alphas = np.linspace(1e-3,10)),
    'SVR': SVR(C = 0.5),
    'knn': KNeighborsRegressor(n_neighbors = 5),
    'dt' : DecisionTreeRegressor(max_depth = 5),
    'rf' : RandomForestRegressor(),
    'et' : ExtraTreesRegressor(),
    'gbm': GradientBoostingRegressor(),
    'histgbm': HistGradientBoostingRegressor()
}

scores_results = {k: np.array([0,0])for k in models.keys()}
```

In [8]:

```
for k, model in models.items():
    model.fit(X_train, y_train)
    scores_results[k] = np.array([r2_score(y_train,model.predict(X_train)
                                  r2_score(y_test,model.predict(X_test))]
```

In [9]:

```
df_scores = pd.DataFrame.from_dict(scores_results,
                                   orient='index',
                                   columns = ['Train Score', 'Test Score'
```

In [10]:

```
df_scores
```

Out[10]:

	Train Score	Test Score
linear	0.611294	0.592609
ridge	0.611289	0.592612
lasso	0.611257	0.592629
SVR	-0.038183	-0.038798
knn	0.438298	0.140096
dt	0.631082	0.596863
rf	0.972422	0.791590
et	1.000000	0.803486
gbm	0.807847	0.782294
histgbm	0.883238	0.832029

Model Tuning

In [11]:

```
from sklearn.model_selection import GridSearchCV
```

Knn Tuning

In [12]:
```
grid_knn = GridSearchCV(models['knn'],
                    cv = 5,
                    n_jobs = 4,
                    param_grid = {'n_neighbors' : np.linspace(2,50,num =
                   )
```

In [13]:
```
grid_knn.fit(X_train, y_train)
```

Out[13]:
```
▸           GridSearchCV
▸ estimator: KNeighborsRegressor
      ▸ KNeighborsRegressor
```

In [14]:
```
score_cols = ['mean_test_score','rank_test_score']
```

In [15]:
```
pd.DataFrame(grid_knn.cv_results_).sort_values('rank_test_score')[['param
```

Out[15]:

	param_n_neighbors	mean_test_score	rank_test_score
2	8	0.119762	1
3	12	0.114060	2
4	15	0.104821	3
1	5	0.099362	4
5	19	0.097268	5
6	22	0.092623	6
7	26	0.083617	7
8	29	0.079209	8
9	32	0.074951	9
10	36	0.069093	10
11	39	0.065311	11
12	43	0.060987	12
13	46	0.059103	13
14	50	0.055576	14
0	2	-0.012401	15

Decision Tree

In [16]:

```
grid_dt = GridSearchCV(models['dt'],
                    cv = 5,
                    n_jobs = 4,
                    param_grid = {'max_depth' : np.linspace(2,50,num = 15
                   )
```

In [17]:

```
grid_dt.fit(X_train, y_train)
```

Out[17]:

```
▸            GridSearchCV
▸ estimator: DecisionTreeRegressor
      ▸ DecisionTreeRegressor
```

In [18]:

```
pd.DataFrame(grid_dt.cv_results_).sort_values('rank_test_score')[['param_
```

Out[18]:

	param_max_depth	mean_test_score	rank_test_score
2	8	0.665432	1
3	12	0.641301	2
4	15	0.608608	3
1	5	0.607672	4
12	43	0.590179	5
13	46	0.589862	6
6	22	0.586469	7
5	19	0.586396	8
8	29	0.586105	9
10	36	0.584260	10
9	32	0.583259	11
14	50	0.582565	12
7	26	0.582141	13
11	39	0.581091	14
0	2	0.446610	15

Tradition Random Forest

```
In [19]: grid_rf = GridSearchCV(models['rf'],
                                cv = 5,
                                n_jobs = 4,
                                param_grid = {'min_samples_leaf' : np.linspace(2,15,n
                                              'max_features': np.array([0.5, 0.66, 0.
                                             }
                               )
```

```
In [20]: grid_rf.fit(X_train, y_train)
```

```
Out[20]: ▸           GridSearchCV
         ▸ estimator: RandomForestRegressor
                 ▸ RandomForestRegressor
```

```
In [21]: pd.DataFrame(grid_rf.cv_results_).sort_values('rank_test_score')[['param_
                                                                            'param_
                                                                            *score_
```

Out[21]:

	param_min_samples_leaf	param_max_features	mean_test_score	rank_test_score
1	2	0.5	0.810069	1
0	2	0.5	0.809996	2
2	3	0.5	0.808271	3
15	2	0.66	0.806959	4
16	2	0.66	0.806630	5
...	...	...	...	...
58	14	0.8	0.780199	71
44	15	0.75	0.779342	72
73	14	0.9	0.779097	73
59	15	0.8	0.778672	74
74	15	0.9	0.777823	75

75 rows × 4 columns

Extra Tree

In [22]:
```
grid_et = GridSearchCV(models['et'],
                       cv = 5,
                       n_jobs = 4,
                       param_grid = {'min_samples_leaf' : np.linspace(2,15,n
                                     'max_features': [0.5, 0.66,0.75,0.8,0.9
                                    }
                      )
```

In [23]:
```
grid_et.fit(X_train, y_train)
```

Out[23]:
```
▸           GridSearchCV
▸ estimator: ExtraTreesRegressor
      ▸ ExtraTreesRegressor
```

In [24]:
```
score_cols = ['mean_test_score','rank_test_score']
pd.DataFrame(grid_rf.cv_results_).sort_values('rank_test_score')[['param_
                                                                   'param_
                                                                   *score_
```

Out[24]:

	param_min_samples_leaf	param_max_features	mean_test_score	rank_test_score
1	2	0.5	0.810069	1
0	2	0.5	0.809996	2
2	3	0.5	0.808271	3
15	2	0.66	0.806959	4
16	2	0.66	0.806630	5
...	...	...	...	...
58	14	0.8	0.780199	71
44	15	0.75	0.779342	72
73	14	0.9	0.779097	73
59	15	0.8	0.778672	74
74	15	0.9	0.777823	75

75 rows × 4 columns

SVR

In [25]:
```
grid_svr = GridSearchCV(models['SVR'],
                        cv = 5,
                        n_jobs = 4,
                        param_grid = {
                                      'C': np.linspace(1, 100, num=10)
                                     }
                       )
```

In [26]:
```
grid_svr.fit(X_train, y_train)
```

Out[26]:
```
▸ GridSearchCV
▸ estimator: SVR
      ▸ SVR
```

In [27]:
```
pd.DataFrame(grid_svr.cv_results_).sort_values('rank_test_score')[['param
```

Out[27]:

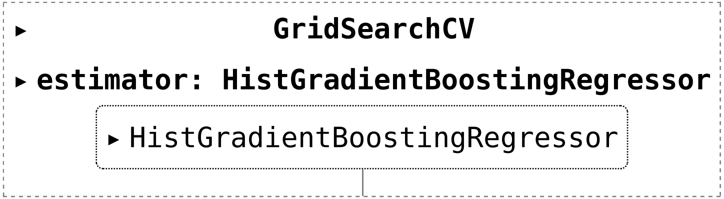
	param_C	mean_test_score	rank_test_score
9	100.0	0.473768	1
8	89.0	0.460383	2
7	78.0	0.445348	3
6	67.0	0.424560	4
5	56.0	0.399404	5
4	45.0	0.359311	6
3	34.0	0.303316	7
2	23.0	0.227075	8
1	12.0	0.121449	9
0	1.0	-0.031922	10

Hist GBM

```
In [28]: grid_histgbm = GridSearchCV(models['histgbm'],
                                    cv = 5,
                                    n_jobs = 4,
                                    param_grid = {
                                        'learning_rate': np.linspace(1e-3, 0.5,
                                    }
                                   )
```

```
In [29]: grid_histgbm.fit(X_train, y_train)
```

Out[29]:



▸ **GridSearchCV**

▸ **estimator: HistGradientBoostingRegressor**

▸ HistGradientBoostingRegressor

```
In [30]: pd.DataFrame(grid_histgbm.cv_results_).sort_values('rank_test_score')[['p
```

Out[30]:

	param_learning_rate	mean_test_score	rank_test_score
17	0.174122	0.834391	1
18	0.184306	0.833940	2
15	0.153755	0.833706	3
14	0.143571	0.833111	4
12	0.123204	0.832650	5
19	0.19449	0.831996	6
16	0.163939	0.831808	7
20	0.204673	0.831417	8
13	0.133388	0.831129	9
21	0.214857	0.831103	10
11	0.11302	0.830831	11
10	0.102837	0.830544	12
9	0.092653	0.830393	13
23	0.235224	0.829551	14
22	0.225041	0.828905	15
24	0.245408	0.828815	16

	param_learning_rate	mean_test_score	rank_test_score
8	0.082469	0.828551	17
29	0.296327	0.828505	18
27	0.275959	0.828269	19
25	0.255592	0.826976	20
28	0.286143	0.826832	21
26	0.265776	0.826182	22
31	0.316694	0.826143	23
30	0.30651	0.825690	24
32	0.326878	0.825264	25
7	0.072286	0.824936	26
33	0.337061	0.824926	27
6	0.062102	0.823383	28
35	0.357429	0.823006	29
34	0.347245	0.822805	30
37	0.377796	0.822548	31
36	0.367612	0.822263	32

	param_learning_rate	mean_test_score	rank_test_score
40	0.408347	0.819058	33
42	0.428714	0.818583	34
38	0.38798	0.818149	35
5	0.051918	0.818016	36
39	0.398163	0.817643	37
43	0.438898	0.817031	38
44	0.449082	0.815944	39
41	0.418531	0.815506	40
46	0.469449	0.813895	41
45	0.459265	0.813579	42
4	0.041735	0.810461	43
48	0.489816	0.809050	44
49	0.5	0.808209	45
47	0.479633	0.807835	46
3	0.031551	0.797225	47
2	0.021367	0.754759	48

	param_learning_rate	mean_test_score	rank_test_score
1	0.011184	0.643038	49
0	0.001	0.117243	50

GBM

In [31]:
```
grid_gbm = GridSearchCV(models['gbm'],
                        cv = 5,
                        n_jobs = 4,
                        param_grid = {'learning_rate': np.linspace(1e-3, 0.5,
                                      }
                       )
```

In [32]:
```
grid_gbm.fit(X_train, y_train)
```

Out[32]:
```
▸              GridSearchCV
▸ estimator: GradientBoostingRegressor
       ▸ GradientBoostingRegressor
```

In [33]:
```
pd.DataFrame(grid_gbm.cv_results_).sort_values('rank_test_score')[['param
```

Out[33]:

	param_learning_rate	mean_test_score	rank_test_score
33	0.337061	0.813311	1
42	0.428714	0.813066	2
38	0.38798	0.811609	3
48	0.489816	0.811556	4
45	0.459265	0.811501	5
37	0.377796	0.811358	6
30	0.30651	0.811212	7
39	0.398163	0.810741	8
34	0.347245	0.810692	9
43	0.438898	0.810616	10
35	0.357429	0.810516	11
44	0.449082	0.810285	12
29	0.296327	0.810056	13
27	0.275959	0.810002	14
32	0.326878	0.809929	15
47	0.479633	0.809924	16

	param_learning_rate	mean_test_score	rank_test_score
36	0.367612	0.809810	17
26	0.265776	0.809678	18
41	0.418531	0.809588	19
31	0.316694	0.809480	20
46	0.469449	0.809445	21
49	0.5	0.809432	22
25	0.255592	0.809053	23
28	0.286143	0.808706	24
40	0.408347	0.808494	25
23	0.235224	0.808393	26
24	0.245408	0.808073	27
22	0.225041	0.807909	28
21	0.214857	0.806149	29
19	0.19449	0.804491	30
20	0.204673	0.804043	31
18	0.184306	0.803326	32

	param_learning_rate	mean_test_score	rank_test_score
17	0.174122	0.801299	33
16	0.163939	0.800008	34
15	0.153755	0.796878	35
14	0.143571	0.794834	36
13	0.133388	0.794122	37
12	0.123204	0.790928	38
11	0.11302	0.790297	39
10	0.102837	0.787303	40
9	0.092653	0.783350	41
8	0.082469	0.779510	42
7	0.072286	0.771788	43
6	0.062102	0.766482	44
5	0.051918	0.756829	45
4	0.041735	0.742691	46
3	0.031551	0.711230	47
2	0.021367	0.647559	48

	param_learning_rate	mean_test_score	rank_test_score
1	0.011184	0.530347	49
0	0.001	0.097162	50

Training New Model

In [34]:

```
new_models = {
    'SVR': SVR(C = 0.5),
    'knn': KNeighborsRegressor(n_neighbors = 8),
    'dt' : DecisionTreeRegressor(max_depth = 8),
    'rf' : RandomForestRegressor(min_samples_leaf = 2, max_features = 0.5
    'et' : ExtraTreesRegressor(min_samples_leaf = 2, max_features = 0.5),
    'gbm': GradientBoostingRegressor(learning_rate = 0.337061),
    'histgbm': HistGradientBoostingRegressor(learning_rate = 0.184306)
}

new_scores_results = {k: np.array([0,0])for k in new_models.keys()}
```

In [35]:

```
for k, model in new_models.items():
    model.fit(X_train, y_train)
    new_scores_results[k] = np.array([r2_score(y_train,model.predict(X_tr
                                  r2_score(y_test,model.predict(X_test))]
```

In [36]:

```
df_new_scores = pd.DataFrame.from_dict(new_scores_results,
                                       orient='index',
                                       columns = ['Train Score', 'Test Score'

df_new_scores.index = 'tuned_' + df_new_scores.index
```

In [37]:

```
pd.concat([df_scores,df_new_scores]).sort_values('Test Score',ascending=
```

Out[37]:

	Train Score	Test Score
tuned_histgbm	0.906846	0.835539
histgbm	0.883238	0.832029
tuned_gbm	0.855144	0.811253
tuned_et	0.961004	0.808369
tuned_rf	0.954766	0.807331
et	1.000000	0.803486
rf	0.972422	0.791590
gbm	0.807847	0.782294
tuned_dt	0.755842	0.667557
dt	0.631082	0.596863
lasso	0.611257	0.592629
ridge	0.611289	0.592612
linear	0.611294	0.592609
tuned_knn	0.344870	0.156911
knn	0.438298	0.140096
tuned_SVR	-0.038183	-0.038798

	Train Score	Test Score
SVR	-0.038183	-0.038798

In [ ]: