In [1]:

```
import pandas as pd
```

In [2]:

```
df1 = pd.read_csv('train_c.csv')
df2 = pd.read_csv('test_c.csv')
df3 = pd.read_csv('sample_submission_c.csv')
```

In [3]:

```
df1.head()
```

Out[3]:

	id	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	M
0	0	2.3859	15	3.827160	1.112100	1280.0	2.486989	34.60	-120.12	
1	1	3.7188	17	6.013373	1.054217	1504.0	3.813084	38.69	-121.22	
2	2	4.7750	27	6.535604	1.103175	1061.0	2.464602	34.71	-120.45	
3	3	2.4138	16	3.350203	0.965432	1255.0	2.089286	32.66	-117.09	
4	4	3.7500	52	4.284404	1.069246	1793.0	1.604790	37.80	-122.41	

In [4]:

```
df1.tail()
```

Out[4]:

	id	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Long
37132	37132	3.3438	50	4.936508	1.079365	1775.0	3.022222	34.19	-1
37133	37133	3.7308	26	5.087533	0.966019	1006.0	4.316901	37.32	-1
37134	37134	4.1716	52	4.678862	1.101485	1156.0	1.431734	37.75	-1
37135	37135	2.7143	16	5.710074	1.068376	584.0	2.803659	38.40	-1
37136	37136	2.2419	34	5.424419	1.058685	1340.0	3.799065	36.34	-1

In [5]:

```
df1.shape
```

Out[5]:

```
(37137, 10)
```

In [6]:

```
df1.columns
```

Out[6]:

```
Index(['id', 'MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population',
       'AveOccup', 'Latitude', 'Longitude', 'MedHouseVal'],
      dtype='object')
```

In [7]:

```
df1.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
df1.isnull().sum()
```

Out[8]:

```
id             0
MedInc         0
HouseAge       0
AveRooms       0
AveBedrms      0
Population     0
AveOccup       0
Latitude       0
Longitude      0
MedHouseVal    0
dtype: int64
```

In [9]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37137 entries, 0 to 37136
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   id           37137 non-null  int64  
 1   MedInc       37137 non-null  float64
 2   HouseAge     37137 non-null  int64  
 3   AveRooms     37137 non-null  float64
 4   AveBedrms    37137 non-null  float64
 5   Population   37137 non-null  float64
 6   AveOccup     37137 non-null  float64
 7   Latitude     37137 non-null  float64
 8   Longitude    37137 non-null  float64
 9   MedHouseVal  37137 non-null  float64
dtypes: float64(8), int64(2)
memory usage: 2.8 MB
```

In [10]:

```
df1.describe()
```

Out[10]:

	id	MedInc	HouseAge	AveRooms	AveBedrms	Population	Av
count	37137.00000	37137.000000	37137.000000	37137.000000	37137.000000	37137.000000	37137
mean	18568.00000	3.851029	26.057005	5.163124	1.062204	1660.778919	2
std	10720.67281	1.803167	12.158221	1.206242	0.096490	1302.469608	2
min	0.00000	0.499900	2.000000	0.851064	0.500000	3.000000	0
25%	9284.00000	2.602300	17.000000	4.357522	1.020305	952.000000	2
50%	18568.00000	3.515600	25.000000	5.068611	1.054545	1383.000000	2
75%	27852.00000	4.699700	35.000000	5.858597	1.088825	1856.000000	3
max	37136.00000	15.000100	52.000000	28.837607	5.873181	35682.000000	502

In [11]:

```
df1.nunique()
```

Out[11]:

```
id             37137
MedInc         12310
HouseAge          51
AveRooms       21278
AveBedrms      13303
Population      3694
AveOccup       20253
Latitude         791
Longitude        755
MedHouseVal     3723
dtype: int64
```

In [12]:

```
df2.head()
```

Out[12]:

	id	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	37137	1.7062	35	4.966368	1.096539	1318.0	2.844411	39.75	-121.85
1	37138	1.3882	22	4.187035	1.098229	2296.0	3.180218	33.95	-118.29
2	37139	7.7197	21	7.129436	0.959276	1535.0	2.888889	33.61	-117.81
3	37140	4.6806	49	4.769697	1.048485	707.0	1.743590	34.17	-118.34
4	37141	3.1284	25	3.765306	1.081633	4716.0	2.003827	34.17	-118.29

In [13]:

```
df2.tail()
```

Out[13]:

	id	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Long
24754	61891	2.2875	34	3.914729	1.085271	866.0	2.071429	34.44	-1
24755	61892	3.0781	33	4.771971	1.038674	1628.0	2.326848	34.09	-1
24756	61893	2.6961	14	4.593960	1.170380	3900.0	2.540034	37.51	-1
24757	61894	7.2315	8	7.508403	1.018692	1388.0	2.601202	33.67	-1
24758	61895	5.7260	30	6.000000	1.000000	15.0	2.500000	37.96	-1

In [14]:

```
df2.shape
```

Out[14]:

```
(24759, 9)
```

In [15]:

```
df2.columns
```

Out[15]:

```
Index(['id', 'MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population',
       'AveOccup', 'Latitude', 'Longitude'],
      dtype='object')
```

In [16]:

```
df2.duplicated().sum()
```

Out[16]:

```
0
```

In [17]:

```
df2.isnull().sum()
```

Out[17]:

```
id            0
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
dtype: int64
```

In [18]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24759 entries, 0 to 24758
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   id          24759 non-null  int64  
 1   MedInc      24759 non-null  float64
 2   HouseAge    24759 non-null  int64  
 3   AveRooms    24759 non-null  float64
 4   AveBedrms   24759 non-null  float64
 5   Population  24759 non-null  float64
 6   AveOccup    24759 non-null  float64
 7   Latitude    24759 non-null  float64
 8   Longitude   24759 non-null  float64
dtypes: float64(7), int64(2)
memory usage: 1.7 MB
```

In [19]:

```
df2.describe()
```

Out[19]:

	id	MedInc	HouseAge	AveRooms	AveBedrms	Population	A
count	24759.000000	24759.000000	24759.000000	24759.000000	24759.000000	24759.000000	2475
mean	49516.000000	3.832618	26.041561	5.168789	1.063599	1679.327548	
std	7147.451994	1.797503	12.177907	1.252874	0.123630	1365.598976	
min	37137.000000	0.499900	2.000000	1.000000	0.560000	3.000000	
25%	43326.500000	2.590150	17.000000	4.356443	1.020460	955.000000	
50%	49516.000000	3.504600	25.000000	5.077143	1.054094	1398.000000	
75%	55705.500000	4.687500	35.000000	5.858646	1.088295	1874.000000	
max	61895.000000	15.000100	52.000000	56.269231	10.500000	35682.000000	23

In [20]:

```
df2.nunique()
```

Out[20]:

```
id            24759
MedInc        10239
HouseAge         51
AveRooms      16283
AveBedrms     10688
Population     3454
AveOccup      15496
Latitude        745
Longitude       736
dtype: int64
```

In [21]:

```
df3.head()
```

Out[21]:

	id	MedHouseVal
0	37137	2.079751
1	37138	2.079751
2	37139	2.079751
3	37140	2.079751
4	37141	2.079751

In [22]:

```
df3.tail()
```

Out[22]:

	id	MedHouseVal
24754	61891	2.079751
24755	61892	2.079751
24756	61893	2.079751
24757	61894	2.079751
24758	61895	2.079751

In [23]:

```
df3.shape
```

Out[23]:

```
(24759, 2)
```

In [24]:

```
df3.columns
```

Out[24]:

```
Index(['id', 'MedHouseVal'], dtype='object')
```

In [25]:

```
df3.duplicated().sum()
```

Out[25]:

```
0
```

In [26]:

```
df3.isnull().sum()
```

Out[26]:

```
id             0
MedHouseVal    0
dtype: int64
```

In [27]:

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24759 entries, 0 to 24758
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   id           24759 non-null  int64  
 1   MedHouseVal  24759 non-null  float64
dtypes: float64(1), int64(1)
memory usage: 387.0 KB
```

In [28]:

```
df3.describe()
```

Out[28]:

	id	MedHouseVal
count	24759.000000	24759.000000
mean	49516.000000	2.079751
std	7147.451994	0.000000
min	37137.000000	2.079751
25%	43326.500000	2.079751
50%	49516.000000	2.079751
75%	55705.500000	2.079751
max	61895.000000	2.079751

In [29]:

```
df1 = df1.drop('id', axis = 1)
```

In [30]:

```
df2 = df2.drop('id', axis = 1)
```

In [31]:

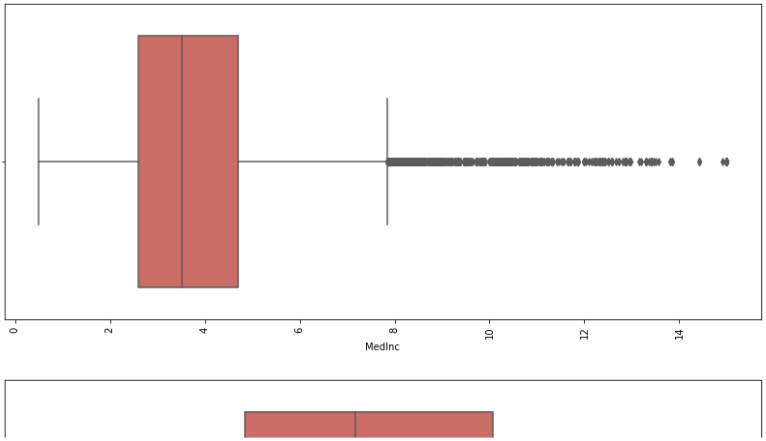
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [32]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [33]:

```
for i in df1.columns:
    plt.figure(figsize = (14,6))
    sns.boxplot(df1[i],data=df1, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.yticks(rotation = 90)
    plt.show()
```



In [34]:

```
Q1 = df1.quantile(0.25)
Q3 = df1.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
MedInc           2.097400
HouseAge        18.000000
AveRooms         1.501075
AveBedrms        0.068521
Population     904.000000
AveOccup         0.730818
Latitude         3.770000
Longitude        3.780000
MedHouseVal      1.452000
dtype: float64
```

In [35]:

```
df1_new = df1[~((df1 < (Q1 - 1.5 * IQR)) |(df1 > (Q3 + 1.5 * IQR))).any(axis=1)]
```

In [36]:

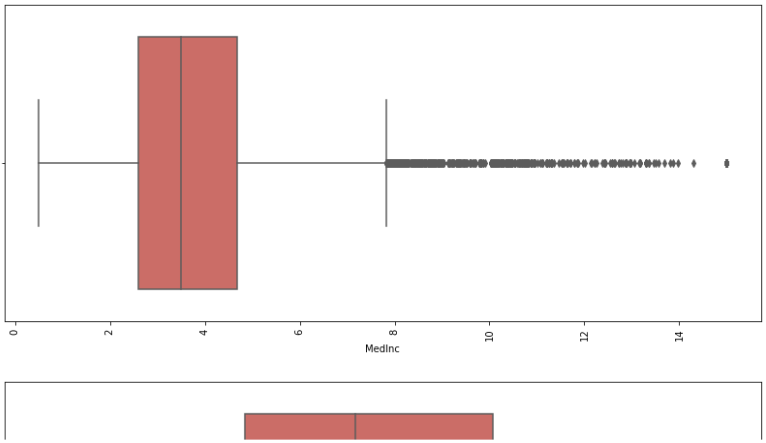
```
df1_new.shape
```

Out[36]:

```
(29631, 9)
```

In [37]:

```
for i in df2.columns:
    plt.figure(figsize = (14,6))
    sns.boxplot(df2[i],data=df2, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.yticks(rotation = 90)
    plt.show()
```



In [38]:

```
Q1 = df2.quantile(0.25)
Q3 = df2.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
MedInc          2.097350
HouseAge       18.000000
AveRooms        1.502204
AveBedrms       0.067835
Population    919.000000
AveOccup        0.729167
Latitude        3.790000
Longitude       3.780000
dtype: float64
```

In [39]:

```
df2_new = df2[~((df2 < (Q1 - 1.5 * IQR)) |(df2 > (Q3 + 1.5 * IQR))).any(axis=1)]
```

In [40]:

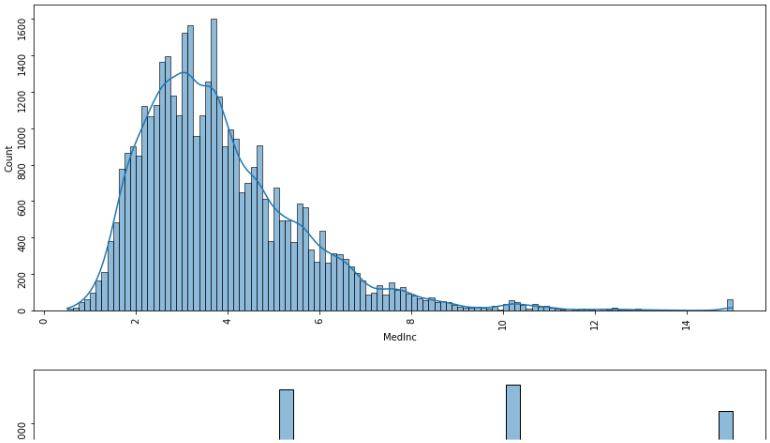
```
df2_new.shape
```

Out[40]:

```
(20327, 8)
```

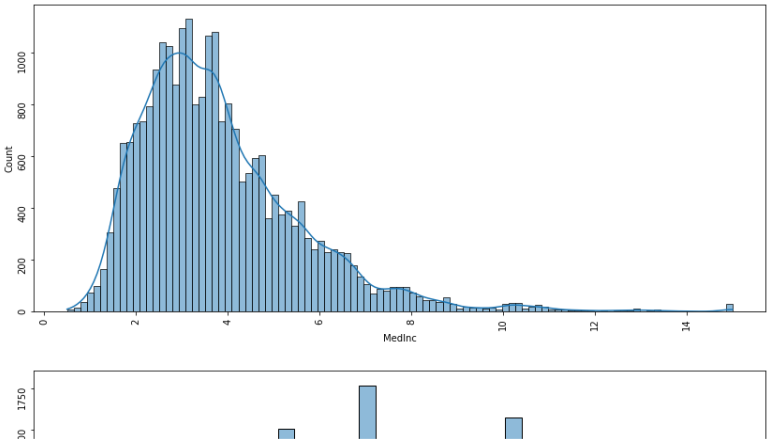
In [41]:

```
for i in df1.columns:
    plt.figure(figsize = (14,6))
    sns.histplot(df1[i], kde = True, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.yticks(rotation = 90)
    plt.show()
```



In [42]:

```
for i in df2.columns:
    plt.figure(figsize = (14,6))
    sns.histplot(df2[i], kde = True, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.yticks(rotation = 90)
    plt.show()
```



In [43]:

```
corr1 = df1_new.corr()
```

In [44]:

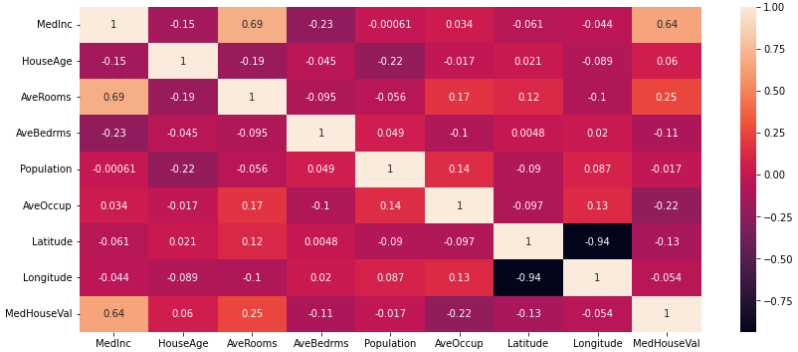
```
corr1
```

Out[44]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Lo
MedInc	1.000000	-0.154998	0.691408	-0.231312	-0.000613	0.033867	-0.060739	-(
HouseAge	-0.154998	1.000000	-0.193637	-0.045089	-0.216033	-0.017077	0.021367	-(
AveRooms	0.691408	-0.193637	1.000000	-0.095022	-0.055967	0.171741	0.118046	-(
AveBedrms	-0.231312	-0.045089	-0.095022	1.000000	0.048547	-0.103755	0.004771	(
Population	-0.000613	-0.216033	-0.055967	0.048547	1.000000	0.142742	-0.089915	(
AveOccup	0.033867	-0.017077	0.171741	-0.103755	0.142742	1.000000	-0.096950	(
Latitude	-0.060739	0.021367	0.118046	0.004771	-0.089915	-0.096950	1.000000	-
Longitude	-0.044226	-0.088901	-0.101521	0.020426	0.086667	0.128508	-0.938111	
MedHouseVal	0.644598	0.059935	0.253826	-0.110152	-0.016966	-0.224396	-0.128430	-(

In [45]:

```
plt.figure(figsize = (14,6))
sns.heatmap(corr1, annot = True)
plt.show()
```



In [46]:

```
corr2 = df2_new.corr()
```

In [47]:

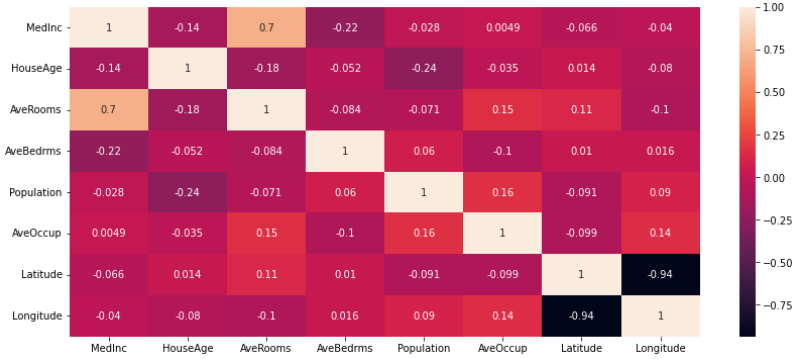
```
corr2
```

Out[47]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Lon
MedInc	1.000000	-0.135119	0.699506	-0.222035	-0.027705	0.004920	-0.066105	-0.0
HouseAge	-0.135119	1.000000	-0.177665	-0.052257	-0.241461	-0.034786	0.013681	-0.0
AveRooms	0.699506	-0.177665	1.000000	-0.083990	-0.071239	0.149369	0.113657	-0.1
AveBedrms	-0.222035	-0.052257	-0.083990	1.000000	0.059616	-0.101494	0.010358	0.0
Population	-0.027705	-0.241461	-0.071239	0.059616	1.000000	0.158499	-0.090892	0.0
AveOccup	0.004920	-0.034786	0.149369	-0.101494	0.158499	1.000000	-0.099062	0.1
Latitude	-0.066105	0.013681	0.113657	0.010358	-0.090892	-0.099062	1.000000	-0.9
Longitude	-0.039657	-0.080053	-0.100590	0.015702	0.089963	0.138032	-0.936620	1.0

In [48]:

```
plt.figure(figsize = (14,6))
sns.heatmap(corr2, annot = True)
plt.show()
```



In [49]:

```
X = df1_new.iloc[:, :-1].values
y = df1_new.iloc[:, -1].values
```

In [50]:

```
from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
print(model.feature_importances_)
```

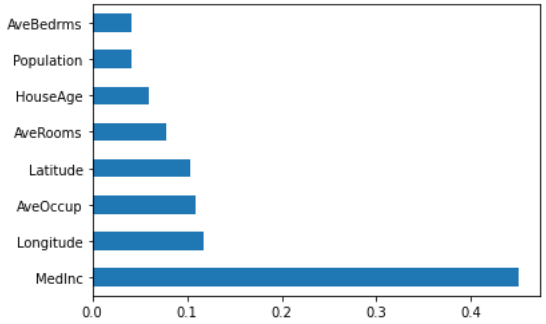
```
[0.44956365 0.06024292 0.07753374 0.04130218 0.04166166 0.10917792
 0.10365402 0.11686392]
```

In [51]:

```
X = df1_new.iloc[:,:-1]
```

In [52]:

```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



In [53]:

```
X1 = df2_new.iloc[:, :-1].values
y1 = df2_new.iloc[:, -1].values
```

In [54]:

```
model1 = ExtraTreesRegressor()
model1.fit(X1,y1)
print(model1.feature_importances_)
```

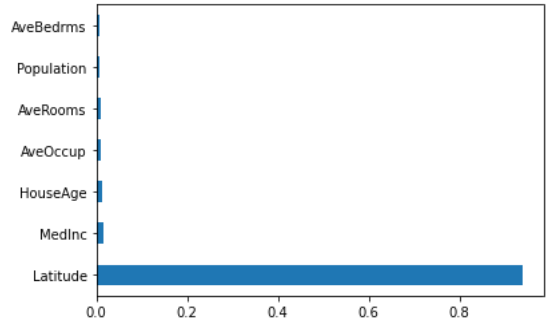
```
[0.01533467 0.01319221 0.0099897  0.00672347 0.006853   0.01019466
 0.93771229]
```

In [55]:

```
X1 = df2_new.iloc[:,:-1]
```

In [56]:

```
feat_importances1 = pd.Series(model1.feature_importances_, index=X1.columns)
feat_importances1.nlargest(10).plot(kind='barh')
plt.show()
```



In [57]:

```
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 = 0)
```

In [58]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [59]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[59]:

```
▾ LinearRegression
LinearRegression()
```

In [60]:

```
y_pred= lr.predict(X_test)
```

In [61]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [62]:

```
import numpy as np
```

In [63]:

```
print("MAE",mean_absolute_error(y_test,y_pred))
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
r2 = r2_score(y_test,y_pred)
print('R2 ',r2)
```

```
MAE 0.4559020229348219
MSE 0.36220681897846946
RMSE 0.6018362061046755
R2  0.5951473610461915
```

In [64]:

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(X_train, y_train)
```

Out[64]:

▾ DecisionTreeRegressor

DecisionTreeRegressor()

In [65]:

```
y_pred = dt.predict(X_test)
```

In [66]:

```
print("MAE",mean_absolute_error(y_test,y_pred))
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
r2 = r2_score(y_test,y_pred)
print('R2 ',r2)
```

```
MAE 0.5446529458410663
MSE 0.5653147126236038
RMSE 0.7518741334981566
R2  0.3681257744109878
```

In [67]:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 100, random_state = 0)
rf.fit(X_train, y_train)
```

Out[67]:

```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

In [68]:

```
y_pred = rf.predict(X_test)
```

In [69]:

```
print("MAE",mean_absolute_error(y_test,y_pred))
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
r2 = r2_score(y_test,y_pred)
print('R2 ',r2)
```

```
MAE 0.3866588000506158
MSE 0.28044284579795753
RMSE 0.5295685468359668
R2  0.6865381316750745
```

In [70]:

```
from sklearn.svm import SVR
sv = SVR(kernel = 'rbf')
sv.fit(X_train, y_train)
```

Out[70]:

```
SVR
SVR()
```

In [71]:

```
y_pred = sv.predict(X_test)
```

In [72]:

```
print("MAE",mean_absolute_error(y_test,y_pred))
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
r2 = r2_score(y_test,y_pred)
print('R2 ',r2)
```

```
MAE 0.4105013446301161
MSE 0.3215388050041222
RMSE 0.5670439180558435
R2  0.6406035808516599
```

In [81]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [87]:

```
model = Sequential()
model.add(Dense(13, input_shape=(8,), kernel_initializer='normal',
                activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))
model.compile(loss='mean_squared_error', optimizer='adam')
```

In [88]:

```
history = model.fit(X_train, y_train, epochs = 100)
```

```
Epoch 1/100
741/741 [==============================] - 1s 1ms/step - loss: 1.2269
Epoch 2/100
741/741 [==============================] - 1s 1ms/step - loss: 0.3572
Epoch 3/100
741/741 [==============================] - 1s 1ms/step - loss: 0.3367
Epoch 4/100
741/741 [==============================] - 1s 2ms/step - loss: 0.3265
Epoch 5/100
741/741 [==============================] - 2s 3ms/step - loss: 0.3218
Epoch 6/100
741/741 [==============================] - 2s 3ms/step - loss: 0.3188
Epoch 7/100
741/741 [==============================] - 2s 2ms/step - loss: 0.3171
Epoch 8/100
741/741 [==============================] - 2s 3ms/step - loss: 0.3158
Epoch 9/100
741/741 [==============================] - 2s 3ms/step - loss: 0.3149
Epoch 10/100
```

In [89]:

```
print(model.evaluate(X_test, y_test))
```

```
186/186 [==============================] - 0s 1ms/step - loss: 0.3137
0.3137081265449524
```