Small data technique

Practice Session 14

Changho Suh

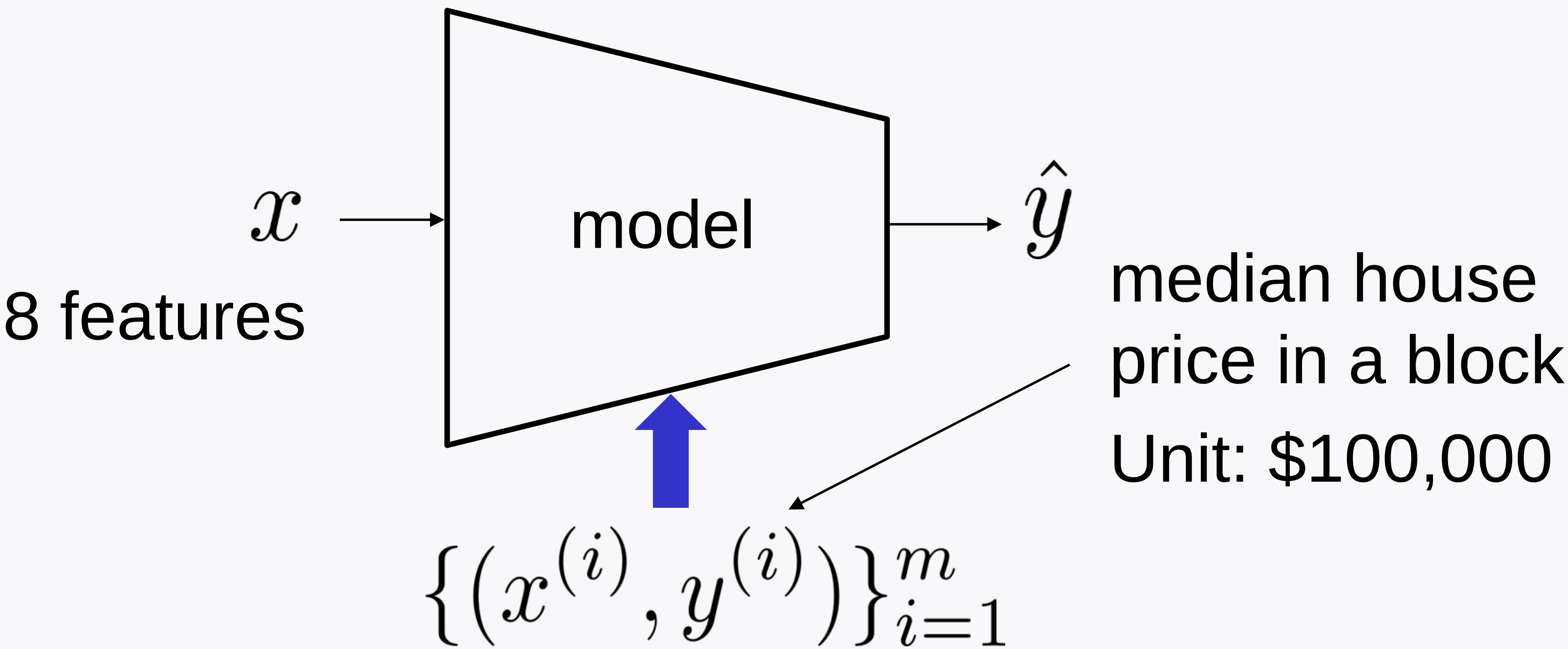
January 26, 2024

Outline

Implementation of **decision tree regressor**

Task: California housing price prediction

California housing price prediction



$$m = 20,640$$

8 features

`MedInc`	median income in block group
`HouseAge`	median house age in block group
`AveRooms`	average # of rooms per household
`AveBedrms`	average # of bedrooms per household
`Population`	block group population
`AveOccup`	average # of household members
`Latitude`	block group latitude
`Longitude`	block group longitude

Load California Housing dataset

```
from sklearn.datasets import fetch_california_housing
```

```
cali_prices = fetch_california_housing()
X_reg = cali_prices.data
y_reg = cali_prices.target
```

```
print(X_reg.shape)
print(y_reg.shape)
```

```
(20640, 8)
(20640,)
```

```
print(cali_prices.feature_names)
```

```
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude',
'Longitude']
```

Data

```
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude',
'Longitude']
```

```
print(X_reg[0])
```

```
[   8.3252       41.            6.98412698    1.02380952  322.
    2.55555556   37.88        -122.23      ]
```

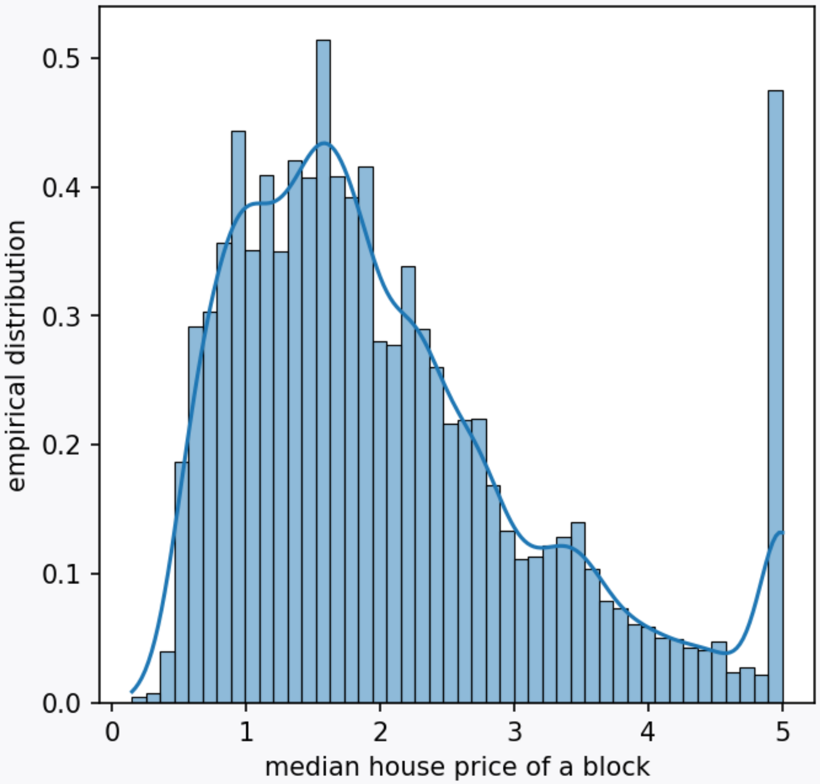
```
print(y_reg)
```

```
[4.526 3.585 3.521 ... 0.923 0.847 0.894]
```

Statistics of median housing prince

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

plt.figure(figsize=(5,5),dpi=150)
sns.histplot(y_reg, kde=True, stat='density')
plt.xlabel('median house price of a block')
plt.ylabel('empirical distribution')
plt.show()
```



Decision tree regressor

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
```

```
# train-test data split
X_train,X_test,y_train,y_test = train_test_split(X_reg,y_reg,test_size=0.01)
```

```
tree_reg = DecisionTreeRegressor(max_depth=4)
```

```
# training
tree_reg.fit(X_train, y_train)
```

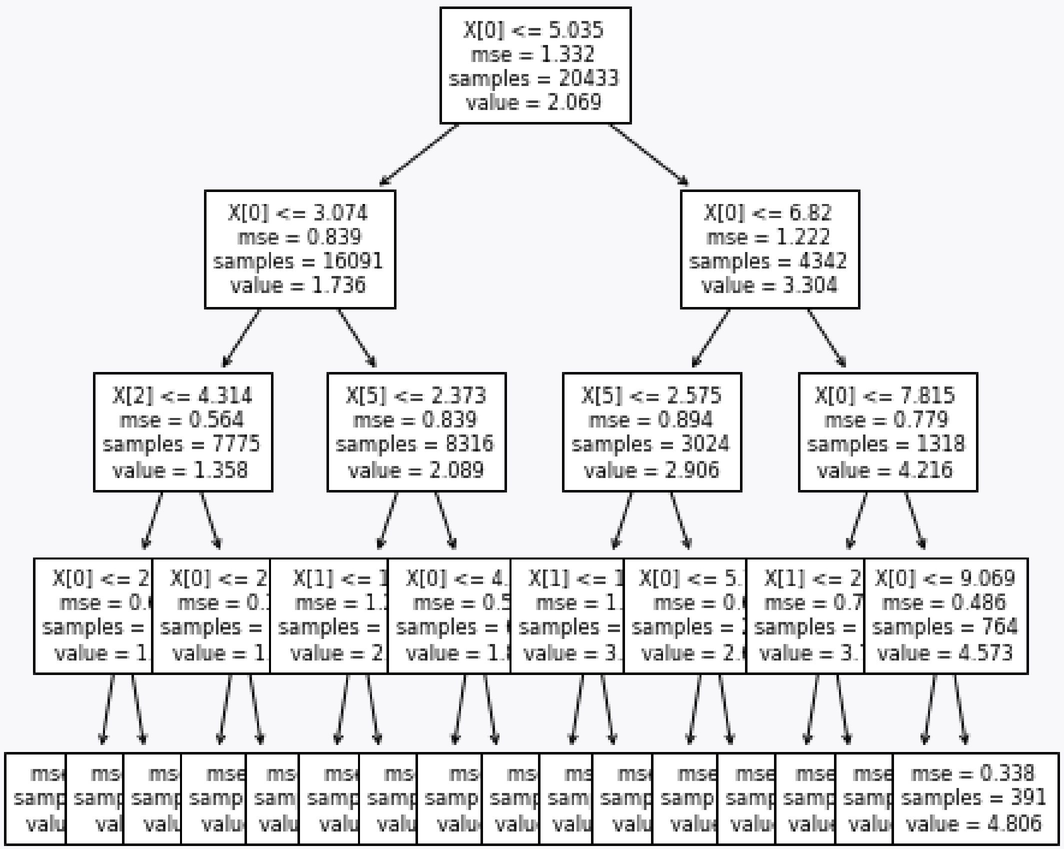
```
# evaluation (R2 score)
test_performance = tree_reg.score(X_test,y_test)
print(test_performance)
```

```
0.698594455059736З
```

Regressor visualization

```
from sklearn.tree import plot_tree
```

```
plot_tree(tree_reg, fontsize=8)
```



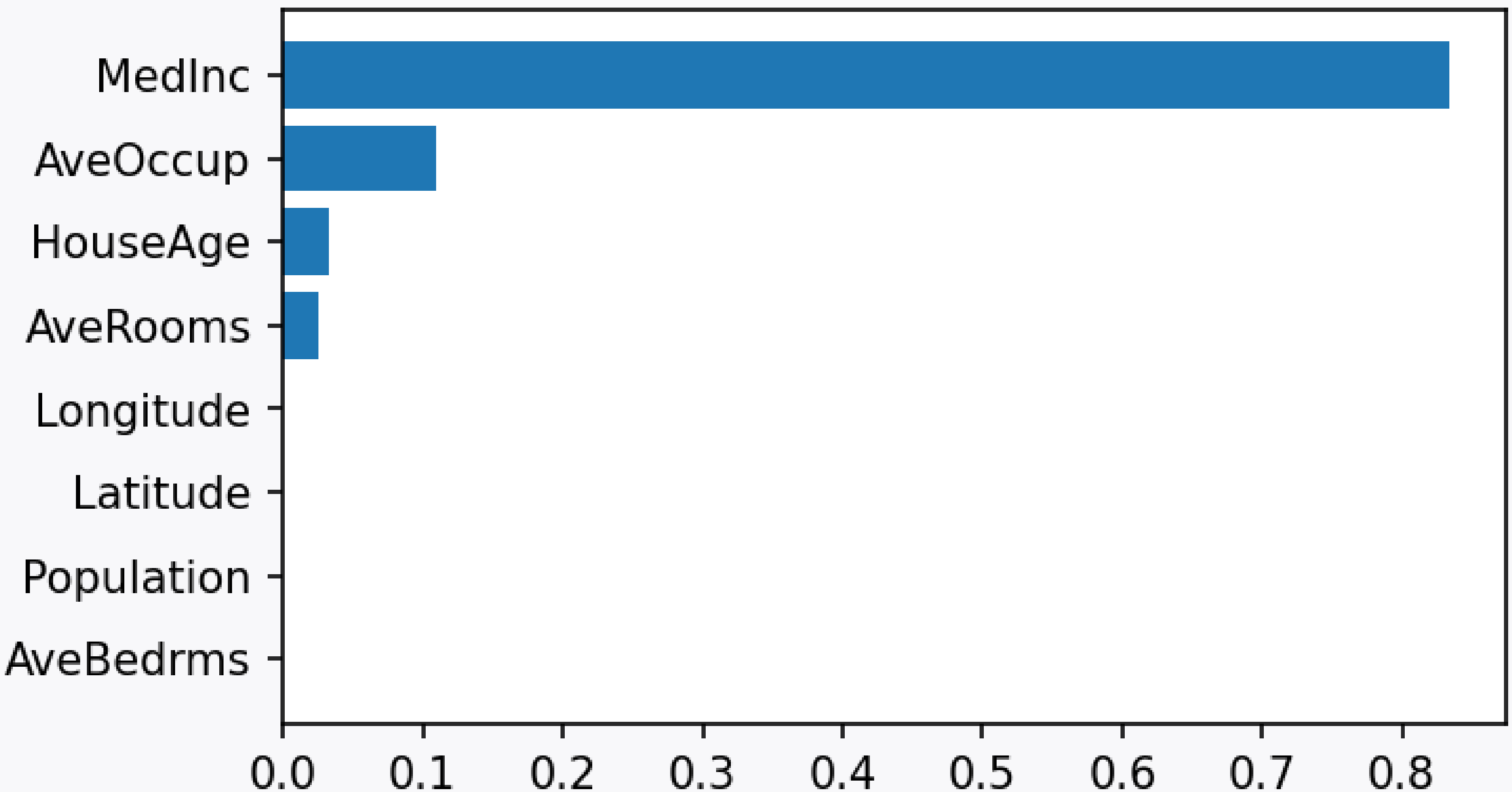
Feature importance

```
feature_importances_cali = tree_reg.feature_importances_
print(feature_importances_cali)
```

```
[0.83298645 0.03137747 0.02509265 0.         0.         0.11054343
 0.         0.        ]
```

Feature importance visualization

```
sorted_idx = tree_reg.feature_importances_.argsort()
plt.barh(np.asarray(cali_prices.feature_names)[sorted_idx],
         feature_importances_cali[sorted_idx])
```



MSE performance

```
from sklearn.metrics import mean_squared_error
```

```
y_pred_train = tree_reg.predict(X_train)
y_pred_test = tree_reg.predict(X_test)
```

```
mse_train = mean_squared_error(y_pred_train, y_train)
mse_test = mean_squared_error(y_pred_test, y_test)
print(mse_train)
print(mse_test)
```

```
0.5556284248376379
0.3857388225897835
```

Comparison with random guess

```
random_guess_train = np.random.normal(
                        loc = np.mean(y_train),
                        scale = np.std(y_train),
                        size =  y_train.shape[0])
```

```
random_guess_test = np.random.normal(
                        loc = np.mean(y_test),
                        scale = np.std(y_test),
                        size = y_test.shape[0])
```

```
mse_first_train = mean_squared_error(random_guess_train, y_train)
mse_first_test = mean_squared_error(random_guess_test, y_test)
print(mse_first_train)
print(mse_first_test)
```

```
2.6862254154194662
2.6724862045770768
```

Look ahead

Implementation of **random forests**

Tasks: Iris plants classification
California housing price prediction
Handwritten digit classification