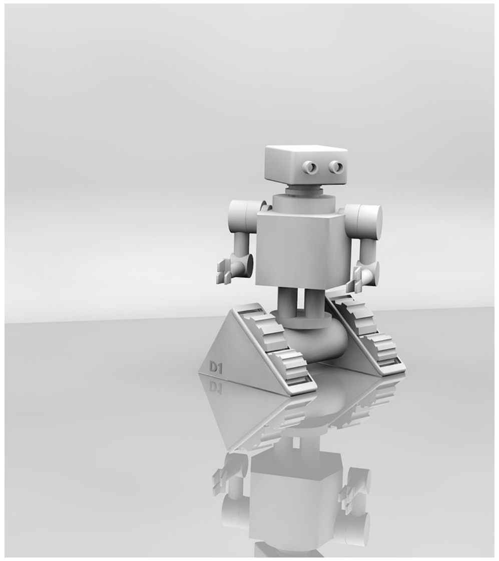


CPSC 481
Artificial Intelligence

Dr. Mira Kim
Mira.kim@fullerton.edu

What we will cover this week

- Overview of ML application development

Steps of Machine Learning Project

- Step 1. Setting up the Goal
- Step 2. Preparing the Dataset
- Step 3. Comprehending the Dataset
- Step 4. Cleaning up the Dataset
- Step 5. Training with ML Algorithm
- Step 6. Tuning the ML model
- Step 7. Deploying the ML Model

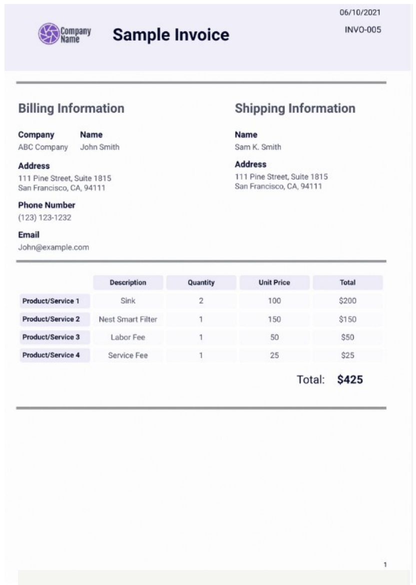
Step 1. Setting up the Goal

Setting up the goal

- Observe the Target Problem
  - What exactly is the business objective?
  - How does the group expect to use and benefit from the model?
  - What does the current solution look like?
- Select ML Algorithm
  - Supervised or Unsupervised?
  - Classification, Regression, Recommendation?
- Decide the Performance Measures
  - Select appropriate performance measures

Example – Document Processing software

- What does the current solution look like? What are the pain points?
  - Rule-based approach of having to use location-based templates
- How do we benefit from the model? What are we trying to improve?
  - Use ML-based approach such as Semi-supervised for higher accuracy and reduced human involvement



Company Name **Sample Invoice**

06/10/2021
INVO-005

Billing Information

Company ABC Company
Name John Smith

Address
111 Pine Street, Suite 1815
San Francisco, CA, 94111

Phone Number
(123) 123-1232

Email
John@example.com

Shipping Information

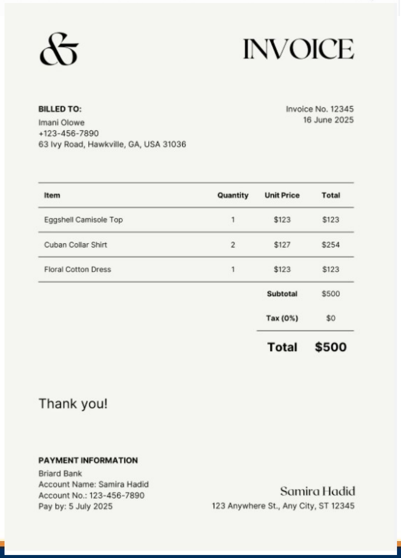
Name
Sam K. Smith

Address
111 Pine Street, Suite 1815
San Francisco, CA, 94111

	Description	Quantity	Unit Price	Total
Product/Service 1	Sink	2	100	$200
Product/Service 2	Nest Smart Filter	1	150	$150
Product/Service 3	Labor Fee	1	50	$50
Product/Service 4	Service Fee	1	25	$25

Total: **$425**

1



INVOICE

BILLED TO:
Imani Olowe
+123-456-7890
63 Ivy Road, Hawkville, GA, USA 31036

Invoice No. 12345
16 June 2025

Item	Quantity	Unit Price	Total
Eggshell Camisole Top	1	$123	$123
Cuban Collar Shirt	2	$127	$254
Floral Cotton Dress	1	$123	$123
		Subtotal	$500
		Tax (0%)	$0
		Total	**$500**

Thank you!

PAYMENT INFORMATION
Briard Bank
Account Name: Samira Hadid
Account No.: 123-456-7890
Pay by: 5 July 2025

Samira Hadid
123 Anywhere St., Any City, ST 12345

Step 2. Preparing the Dataset

Preparing the dataset

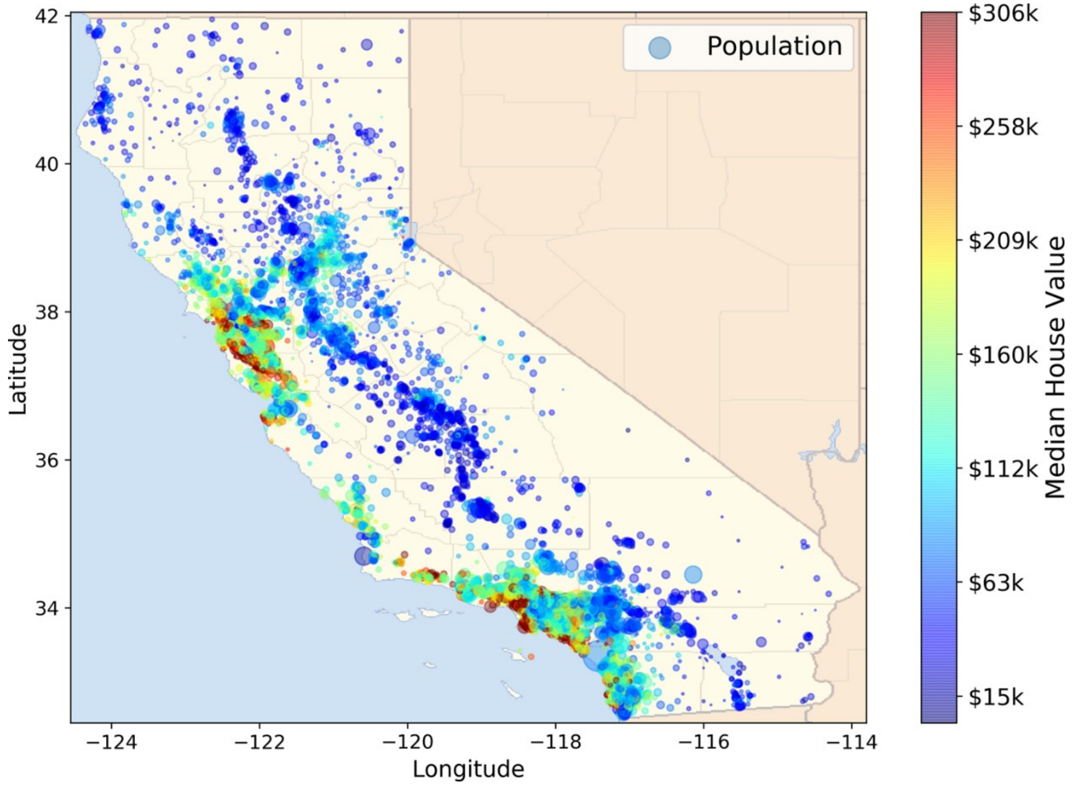
- Locate the dataset used for training the model
- Download the dataset
- Read the dataset into the program

Example Dataset

- ‘California Housing Prices’ Data Set
  - Median House Prices for California Districts Derived from 1990 Census
  - Contains information about 20,640 districts in California
- Features used in the Dataset
  - longitude
  - latitude
  - housing_median_age
  - total_rooms
  - total_bedrooms
  - median_income
  - ocean_proximity
  - …

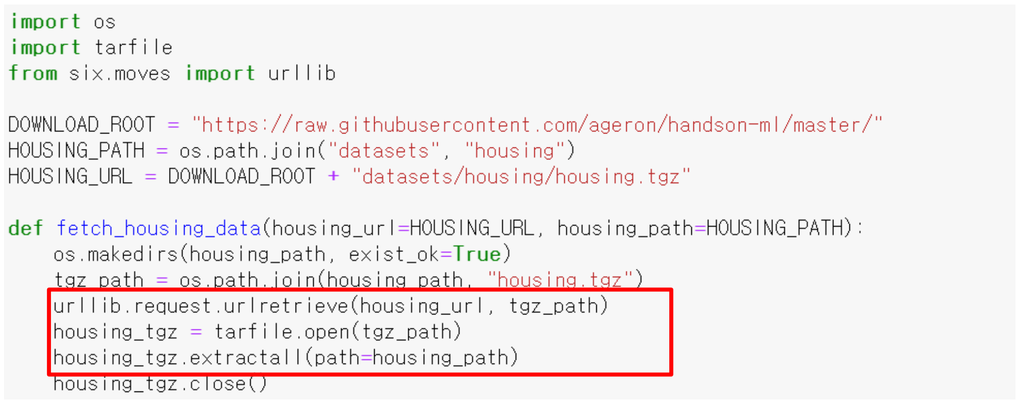
Example Dataset

- Distribution of House Prices by considering
  - longitude
  - latitude



Downloading the Dataset

- Format of the Example Dataset
  – housing.tgz file
- Code to Extract the housing.tgz file

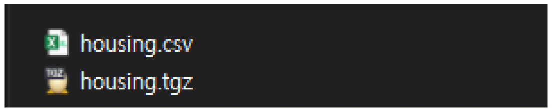


```
import os
import tarfile
from six.moves import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    os.makedirs(housing_path, exist_ok=True)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
```

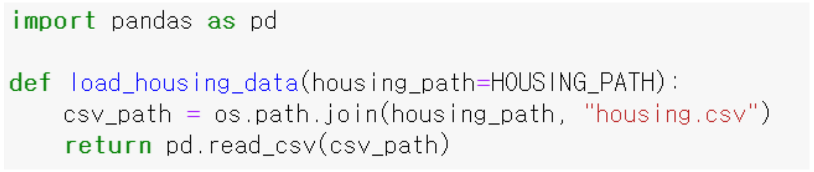
  – Results
    - To extract housing.csv from housing.tgz file



```
housing.csv
housing.tgz
```

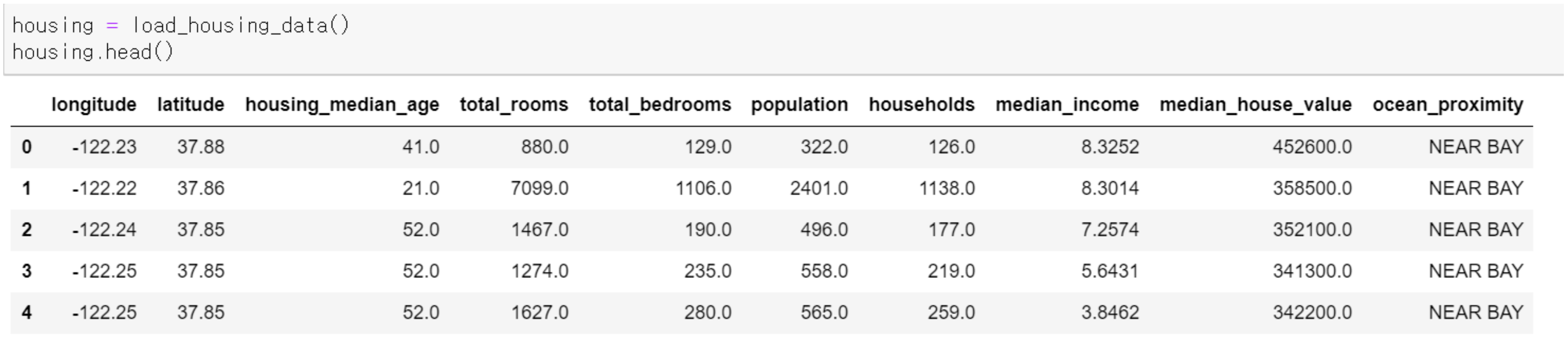
Dataset Preparation

- To load data to csv file
  - To apply dataframe type in *Pandas* library



```
import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

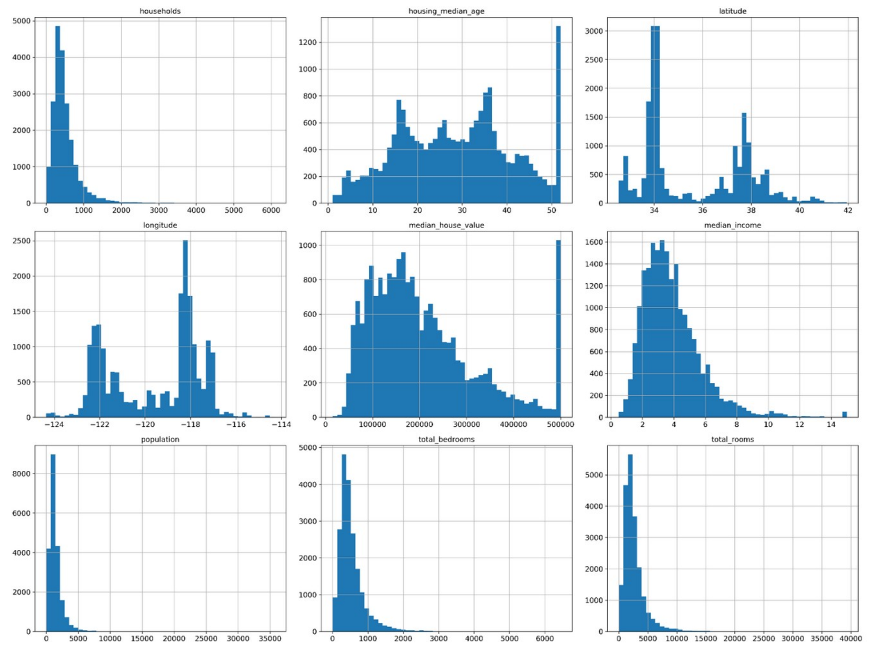


```
housing = load_housing_data()
housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

Dataset Preparation

- To visualize distribution of numerical attributes using graphs
  - Result of hist() Method

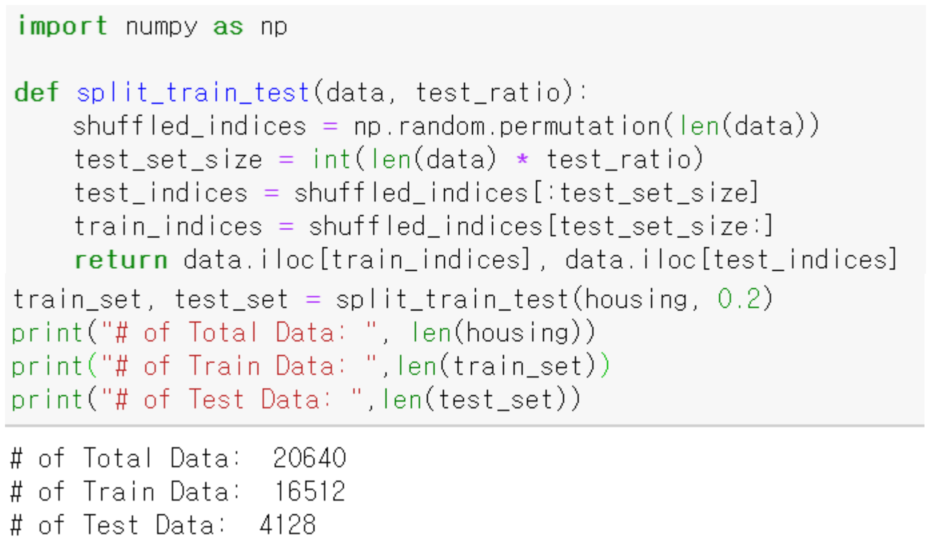


Splitting Dataset

- Training Set
  - Set of data instances used for training the model.
  - Often, 80% of the whole dataset
- Test Set
  - Set of data instances used for evaluating the model performance.
  - Often, 20% of the whole dataset
- To split train set and test set
  - Providing method in Scikit-Learn
    - train_test_split()

Splitting Dataset

- split_train_test(data, test_ratio): ndarray, ndarray
  - To pick some instances randomly in dataset typically 20% of dataset
    - data: ndarray, dataset
    - test_ratio: float, rate of test set size on dataset, range is 0 to 1, to set as 0.2
    - Return: ndarrayes, train set and test set
    - ☾ It can't return same result every time.



```
import numpy as np

def split_train_test(data, test_ratio):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]
train_set, test_set = split_train_test(housing, 0.2)
print("# of Total Data: ", len(housing))
print("# of Train Data: ",len(train_set))
print("# of Test Data: ",len(test_set))
```

```
# of Total Data:  20640
# of Train Data:  16512
# of Test Data:  4128
```

Step 3. Comprehending the Dataset

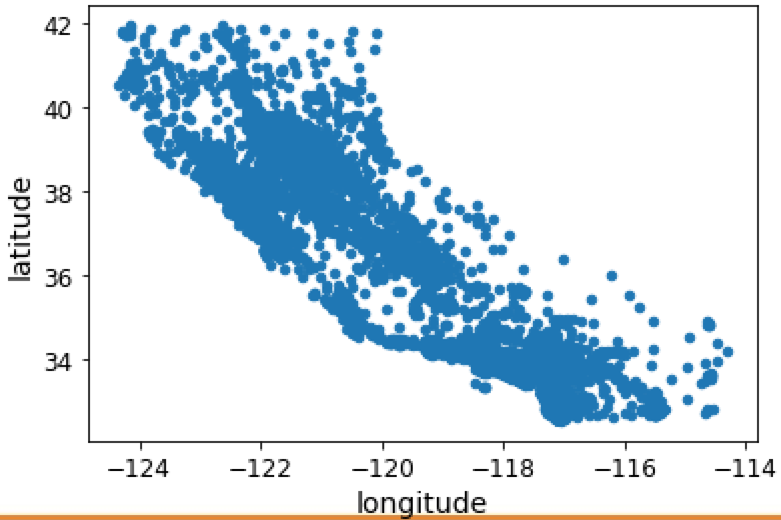
Comprehending the Dataset

- Observe the content of the dataset
- Visualize the content of the dataset using charts
- Apply statistics on the dataset

Visualizing Dataset

- plot() Method
  - To make plots of DataFrame using matplotlib
- Example) House Distribution
  - To apply longitude and latitude data in dataframe

```
housing.plot(kind="scatter", x="longitude", y="latitude")
```

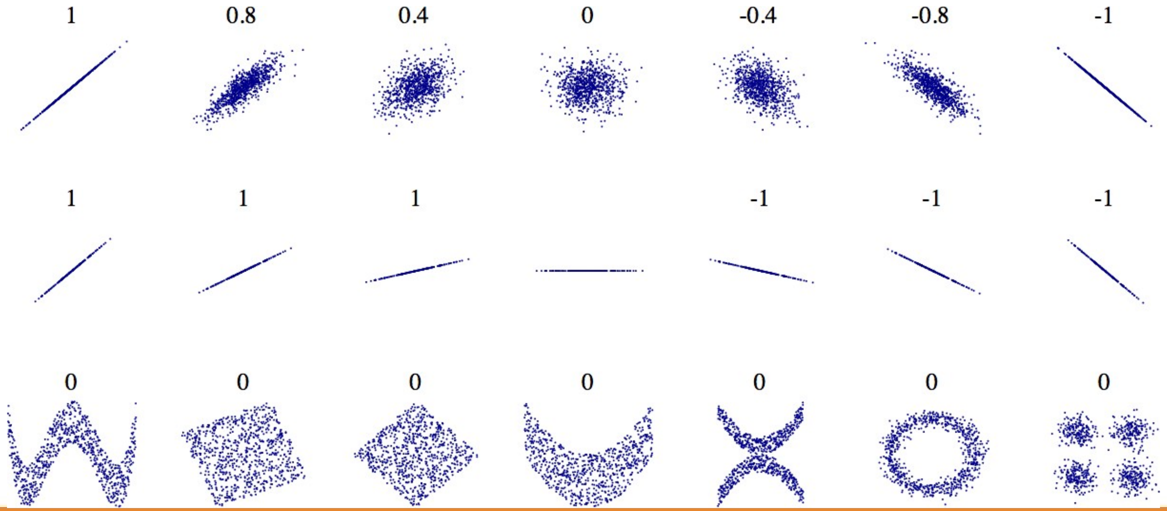


Finding Correlations

- Correlation Coefficient
  - Definition
    - Strength of the relationship between the relative movements of two variables
    - To only measure linear correlations
  - Features
    - To only measure linear correlation (if x goes up, y goes up or down.)
    - To miss out on nonlinear relationships
    - Correlation Coefficient Range: -1 to 1
    - As close to 1, there is a strong positive correlation.
    - As close to -1, there is a strong negative correlation.

Finding Correlations

- Correlation Coefficient (Cont.)
  - Standard Correlation Coefficient of Various Dataset
    - To show various plots along with the correlation coefficient between their horizontal and vertical axes



Finding Correlations

- corr():[] method
  - The method computes correlation coefficient between every pair of attributes.
- Example
  - Correlation coefficient is computed between “median_house_ value” and each attributes in the housing dataset.
  - Relation with median_income has the strongest positive correlation.
  - Relation with latitude is weak negative correlation.

```
corr_matrix = housing.corr()
corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```

Step 4. Cleaning up Dataset

Cleaning up Dataset

- To remove erroneous data elements from training set
  - Bias
  - Noise
  - Outlier
  - Ambiguity
  - Missing Features

Data Purification

- Why purify?
  - To generate ML models with high performance
- Step to prepare the data
  - Step 1. Data Cleaning
  - Step 2. Handling Text and Categorical Attributes
  - Step 3. Custom Transformers
  - Step 4. Feature Scaling
  - Step 5. Transformation Pipelines

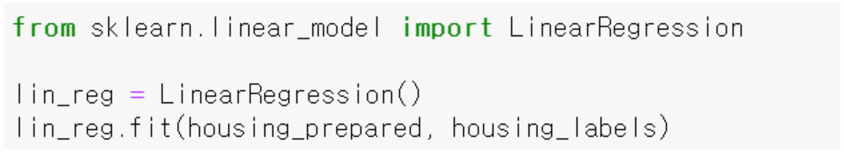
Step 5. Training with ML Algorithm

Training with ML Algorithm

- Choosing ML Algorithm
- Training the Model
- Predicting with the Model

Training and Selecting Model

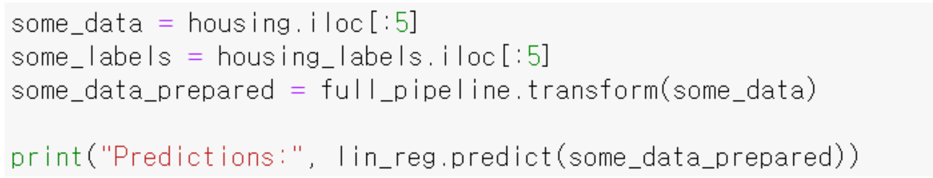
- Training
  - This is to generate a knowledge model using a training set.
  - fit(x, y)
    - x: training set
    - y: Target values



```python
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)
```

- Predicting
  - This is to make a prediction for a given observation, which can be a value in a test set.
  - predict(x): array
    - x: test set
    - return : array, detecting result



```python
some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)

print("Predictions:", lin_reg.predict(some_data_prepared))
```

Step 6. Fine-tuning the ML model

Tools for fine-tuning model

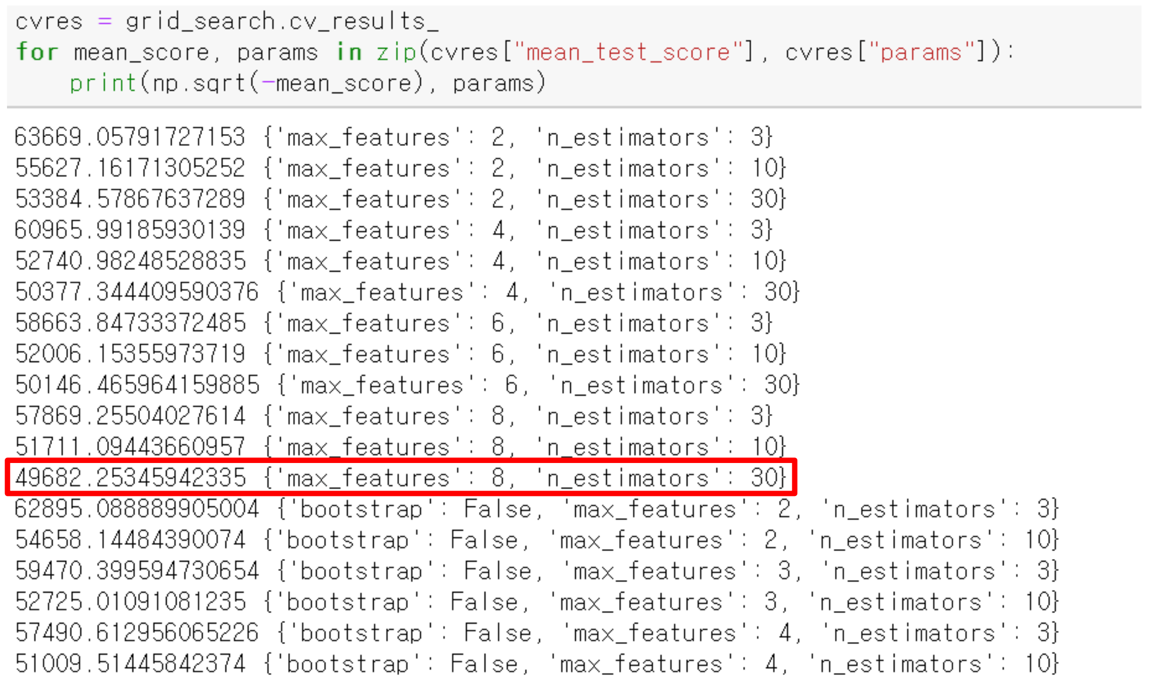
- **Grid Search:** Exhaustively tries every combination of hyperparameters.
- **Random Search:** Tries random combinations of hyperparameters; sometimes more efficient than grid search.
- **Bayesian Optimization:** Leverages probability to find the best model parameters.
- **Automated ML frameworks:** Tools like AutoML or cloud-based solutions offer automated hyperparameter tuning.

Fine-Tune Model

- Grid Search
  - Exhaustively tries every combination of hyperparameters to find best hyperparameters combination
  - Define a grid of hyperparameters and their respective ranges. Grid Search then trains a model for every possible combination of these parameters and evaluates each model
  - Pros: Ensures that you will find the best combination of parameters
  - Cons: Can be very time-consuming, as it grows exponentially with the number of parameters and the values they can take

Fine-Tune Model

- Grid Search
  - Result
    - The lowest error is when max_features is 8 and n_estimators is 30.
      - RMSE is 49,682.



```python
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
63669.05791727153 {'max_features': 2, 'n_estimators': 3}
55627.16171305252 {'max_features': 2, 'n_estimators': 10}
53384.57867637289 {'max_features': 2, 'n_estimators': 30}
60965.99185930139 {'max_features': 4, 'n_estimators': 3}
52740.98248528835 {'max_features': 4, 'n_estimators': 10}
50377.344409590376 {'max_features': 4, 'n_estimators': 30}
58663.84733372485 {'max_features': 6, 'n_estimators': 3}
52006.15355973719 {'max_features': 6, 'n_estimators': 10}
50146.465964159885 {'max_features': 6, 'n_estimators': 30}
57869.25504027614 {'max_features': 8, 'n_estimators': 3}
51711.09443660957 {'max_features': 8, 'n_estimators': 10}
49682.25345942335 {'max_features': 8, 'n_estimators': 30}
62895.088889905004 {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
54658.14484390074 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
59470.399594730654 {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
52725.01091081235 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
57490.612956065226 {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
51009.51445842374 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
```

Step 7. Deploying the ML Model

Overview of the Step

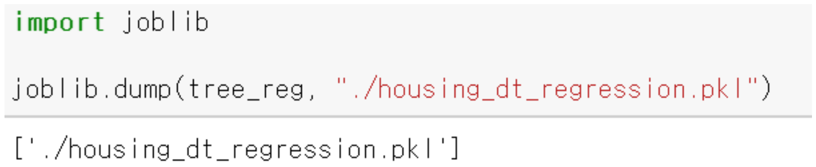
- Installing the Model
- Getting ready for making predictions

Methods to Deploy ML Model

- To store trained model to local
  - To use *Joblib* library
  - To be able to make prediction by calling *predict()* method
- To apply the stored model in server
  - To use query through REST API
- To deploy the model in the cloud
  - To be able to use Google Cloud AI Platform (Google Cloud ML Engine)

jobilb library

- Code for Saving Trained Model in Disk
  - To save model whose name is “housing_dt_regression.pkl”

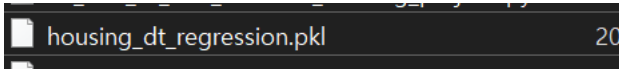


```
import joblib

joblib.dump(tree_reg, "./housing_dt_regression.pkl")
```

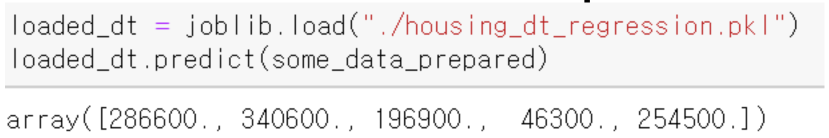
```
['./housing_dt_regression.pkl']
```

  - Result



housing_dt_regression.pkl 20

- Code for Loading Saved Model from Disk
  - To load saved file and predict *some_data_prepared*



```
loaded_dt = joblib.load("./housing_dt_regression.pkl")
loaded_dt.predict(some_data_prepared)
```

```
array([286600., 340600., 196900.,  46300., 254500.])
```

References

- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 3rd Edition