19ECCN1702 - Machine Learning

UNIT 2- MACHINE LEARNING FRAMEWORK

Unit I INTRODUCTION **9 Hours**

Introduction to Machine Learning - Types of Machine Learning systems - Challenges in Machine Learning - Overfitting and Under fitting - Testing and Validating the model - Bias and Variance

Unit II MACHINE LEARNING FRAMEWORK **9 Hours**

Problem Formulation - Get the data - analyze and visualize the data - Prepare the data for ML algorithms - sample complexity - Hypothesis space - Model evaluation and Improvement: Cross validation - Grid search - Evaluation Metrics - Kernel functions

Unit III SUPERVISED LEARNING **9 Hours**

Linear and Logistic Regression – Eigen Values and Eigen vectors - Naïve Bayes Classifier: Maximum Likelihood, Minimum Description Length – Gradient Descent - Decision Trees - Ensembles of Decision Trees - Support Vector Machine(SVM)

Unit IV UNSUPERVISED LEARNING 9 Hours

Clustering: k-Means clustering- Agglomerative Clustering - DBSCAN- Gaussian Mixtures- precision and recall - Collaborative filtering and Content Filtering

Unit V NEURAL NETWORK AND DEEP LEARNING 9 Hours

Biological Neuron - Logical computation with Neuron - Perceptron - Sigmoid and softmax functions - Multi Layer Perceptron(MLP) with Back propagation - Regression MLPs - Classification MLPs - Fine Tuning NN models - Convolutional Neural Network: Architecture of Visual cortex - Convolutional Layers - Stacking Multiple Feature Maps- CNN architectures

Course Outcomes	**Cognitive Level**
At the end of this course, students will be able to:	
CO1:Describe the types and challenges in Machine learning for exploring the machine learning concepts	Understand
CO2:Illustrate the machine learning framework for implementation of machine learning projects	Apply
CO3:Interpret the supervised learning techniques for classification	Apply
CO4:Demonstrate the un-supervised learning methods for clustering and classification	Apply
CO5:Construct the Neural network and deep learning models for classification	Apply

Text Book(s):

T1. AurélienGéron," Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow", Second edition, O'Reilly Media, Inc,2019

T2. Andreas C. Müller and Sarah Guido, "Introduction to Machine Learning with Python A Guide for Data Scientists", First Edition,O'Reilly,2017

Reference Book(s):

R1. Ethem Alpaydin, "Introduction to Machine Learning 3e (Adaptive Computation and Machine Learning Series)", 3rd Edition, MIT Press, 2014

R2. Jason Bell, "Machine learning - Hands on for Developers and Technical Professionals",1st Edition, Wiley, 2014

R3. Peter Flach, "Machine Learning: The Art and Science of Algorithms that Make Sense of Data", 1st Edition, Cambridge University Press, 2012.

Web References:

1. https://www.kaggle.com/kanncaa1/machine-learning-tutorial-for-beginners
2. https://nptel.ac.in/courses/106/106/106106139/
3. https://archive.ics.uci.edu/ml/datasets.php

To get data

- Popular open data repositories
  - —UC Irvine Machine Learning Repository
  - —Kaggle datasets
  - —Amazon's AWS datasets
- Meta portals (they list open data repositories):
  - —*http://dataportals.org/*
  - —*http://opendatamonitor.eu/*
  - —*http://quandl.com/*
- Other pages listing many popular open data repositories:
  - —Wikipedia's list of Machine Learning datasets
  - —Quora.com question
  - —Datasets subreddit

End-to-End ML Project Steps

- Look at the big picture
- Get the data
- Discover and visualize the data to gain insights
- Prepare the data for ML algorithms
- Select a model and train it
- Fine-tune your model
- Present your solution to the test set
- Launch, monitor, and maintain your system

The task

To build a model of housing prices in California using the California census data. This data has metrics such as the population, median income, median housing price, and so on for each district of California.

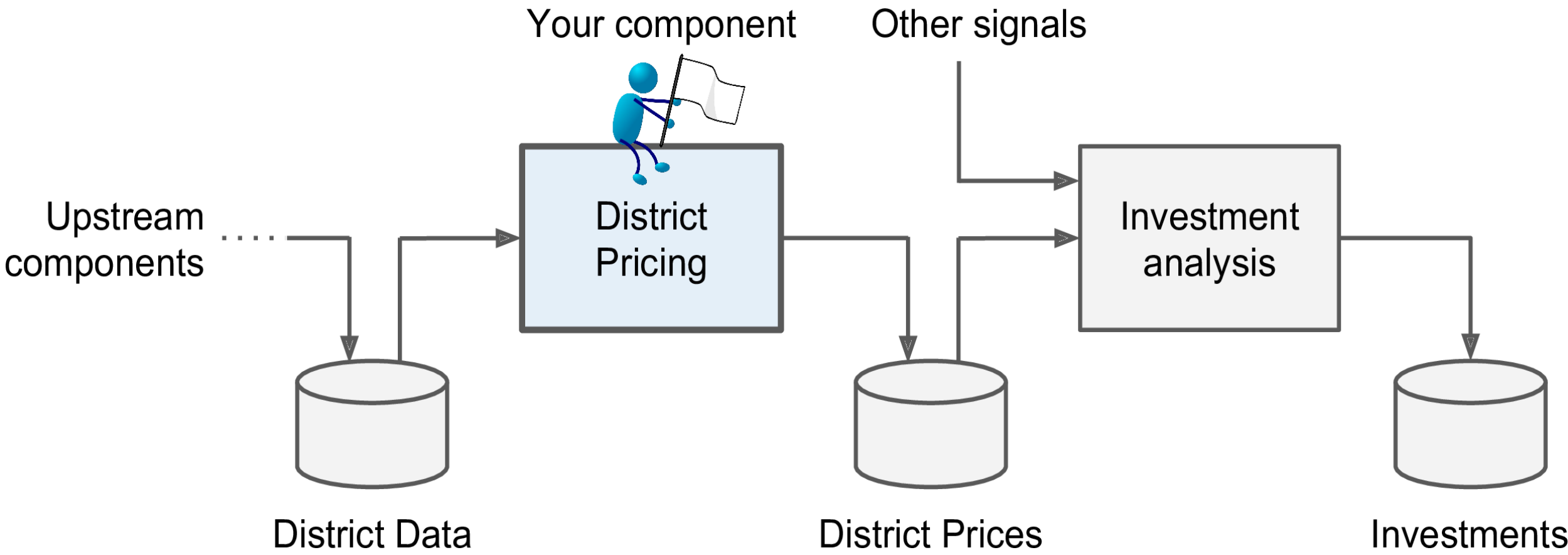
Your model should learn from this data and be able to predict the median housing price in any district, given all the other metrics.

Problem Framing

This is important because it will determine how you frame the problem, what algorithms you will select, what performance measure you will use to evaluate your model, and how much effort you should spend tweaking it.

The model's output (a prediction of a district's median housing price) will be fed to another Machine Learning system along with many other *signals*. This downstream system will determine whether it is worth investing in a given area or not

A Machine Learning pipeline for real estate investments



Problem Framing

Formal problem framing is the critical beginning for solving an ML problem, as it forces us to better understand both the problem and the data in order to design and build a bridge between them.

At a high level, ML problem framing consists of two distinct steps,

1. Determining whether ML is the right approach for solving a problem.
2. Framing the problem in ML terms

Frame the problem: supervised? classification/regression? batch/online? etc

- supervised learning task since you are given *labeled* training examples (each instance comes with the expected output, i.e., the district's median housing price).
- Regression task
- Multiple regression
- Univariate regression

Pipeline

Pipeline:

A sequence of data processing *components* is called a data *pipeline*. Pipelines are very common in Machine Learning systems, since there is a lot of data to manipulate and many data transformations to apply.

Problem framing ensures that an ML approach is a good solution to the problem before beginning to work with data and train a model

Performance Measure

Typical performance measure for regression problems is the Root Mean Square Error (RMSE).

$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

It gives an idea of how much error the system typically makes in its predictions, with a higher weight for large errors

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

- *m* is the number of instances in the dataset
- **x***(i)* is a vector of all the feature values (excluding the label) of the *i th* instance in the dataset, and
- *y(i)* is its label (the desired output value for that instance).
- **X** is a matrix containing all the feature values (excluding labels) of all instances in the dataset
- h is hypothesis

Example

If there are 2000 districts, then m =2000

If the attributes are longitude(118.29°,),latitude(33.91°),inhabitants($156,400) with medium income of $38,372 and the median house value is $156,400 (ignoring the other features for now),

then:

$$\mathbf{x}^{(1)} = \begin{pmatrix} -118.29 \\ 33.91 \\ 1,416 \\ 38,372 \end{pmatrix} \qquad y^{(1)} = 156,400$$

Example

X is a matrix containing all the feature values (excluding labels) of all instances in the dataset. There is one row per instance and the *ith* row is equal to the transpose of **x**(*i*), noted (**x**(*i*))*T*

$$\mathbf{X} = \begin{pmatrix} \left(\mathbf{x}^{(1)}\right)^T \\ \left(\mathbf{x}^{(2)}\right)^T \\ \vdots \\ \left(\mathbf{x}^{(1999)}\right)^T \\ \left(\mathbf{x}^{(2000)}\right)^T \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1,416 & 38,372 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

h is your system's prediction function, also called a hypothesis

When your system is given an instance's feature vector x(i), it outputs a predicted value $\hat{y}(i) = h(x(i))$ for that instance

RMSE(X,h) is the cost function measured on the set of examples using your hypothesis h.

Mean Absolute Error

$$\text{MAE}(\mathbf{X}, h) = \frac{1}{m} \sum_{i=1}^{m} \left| h\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right|$$

For example, suppose that there are many outlier districts. In that case, you may consider using the *Mean Absolute Error*

Both the RMSE and the MAE are ways to measure the distance between two vectors: the vector of predictions and the vector of target values.

Check the Assumptions

What if the downstream system actually converts the prices into categories ?

(e.g., “cheap,” “medium,” or “expensive”)

Scientific Python

Python modules:

- ✓ Jupyter
- ✓ NumPy
- ✓ Pandas,
- ✓ Matplotlib
- ✓ Scikit-Learn

Creating Isolated Environment

```
python3 -m pip install --user -U virtualenv
virtualenv my_env
source my_env/bin/activate
```

virtualenv is **a tool used to create isolated Python environments**.

It creates a folder which contains all the necessary executables to use the packages that a Python project would need.

Get the data-Set up

```
# Python ≥3.5 is required
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn ≥0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)
```

Get the data-Set up

```
# Where to save the figures
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "end_to_end_project"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

Get the data-Download the Data

```
import os
import tarfile
from six.moves import urllib
DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml2/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):
if not os.path.isdir(housing_path):
os.makedirs(housing_path)
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()
```

Now when you call fetch_housing_data(),

it creates a *datasets/housing* directory in your workspace, downloads the *housing.tgz* file, and extracts the *housing.csv* from it in this directory.

Get the data-Download the Data

The function creates the `./datasets/housing` directory if it does not exist yet (`os.makedirs(...)`), then it downloads the `housing.tgz` file at the URL `"https://raw.githubusercontent.com/ageron/handson-ml/master/housing.tgz"` and saves it in the `./datasets/housing` directory (`urlretrieve(...)`), then it uncompresses it (`extractall()`).

Get the data-Download the Data

Include Pandas. This function returns a Pandas Data Frame object containing all the data.

```
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)
```

Quick Look at the Data Structure

```
In [5]: housing = load_housing_data()
        housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatio
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

The info() method is useful to get a quick description of the data, in particular the total number of rows, and each attribute's type and number of non-null values

```
In [6]: housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

There are 20,640 instances in the dataset. Notice that the total_bed rooms attribute has only 20,433 non-null values, meaning that 207 districts are missing this feature.

```
>>> housing["ocean_proximity"].value_counts()
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

Summary of each numerical attribute

```
In [8]: housing.describe()
```

Out[8]:

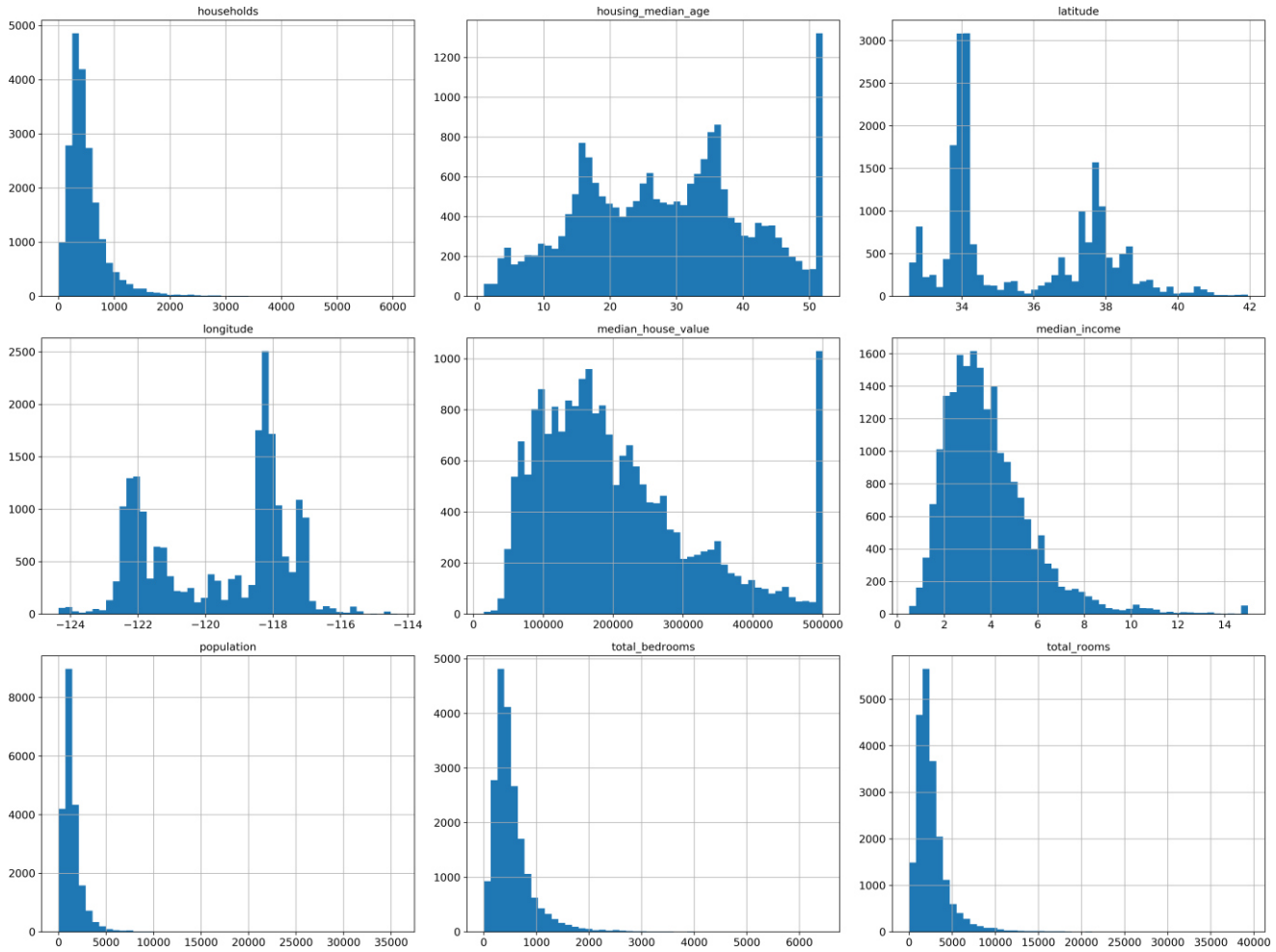
	longitude	latitude	housing_median_age	total_rooms	total_bedr
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000(
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553
std	2.003532	2.135952	12.585558	2181.615252	421.38507(
min	-124.350000	32.540000	1.000000	2.000000	1.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.00000(
50%	-118.490000	34.260000	29.000000	2127.000000	435.00000(
75%	-118.010000	37.710000	37.000000	3148.000000	647.00000(
max	-114.310000	41.950000	52.000000	39320.000000	6445.0000(

HISTOGRAM

A histogram shows the number of instances (on the vertical axis) that have a given value range (on the horizontal axis).

You can either plot this one attribute at a time, or you can call the hist() method on the whole dataset, and it will plot a histogram for each numerical attribute

Histogram


households
housing_median_age
latitude
longitude
median_house_value
median_income
population
total_bedrooms
total_rooms

Create a Test Set

```
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(housing, test_size=0.2,
random_state=42)
```

Prepare the Data for Machine Learning Algorithms

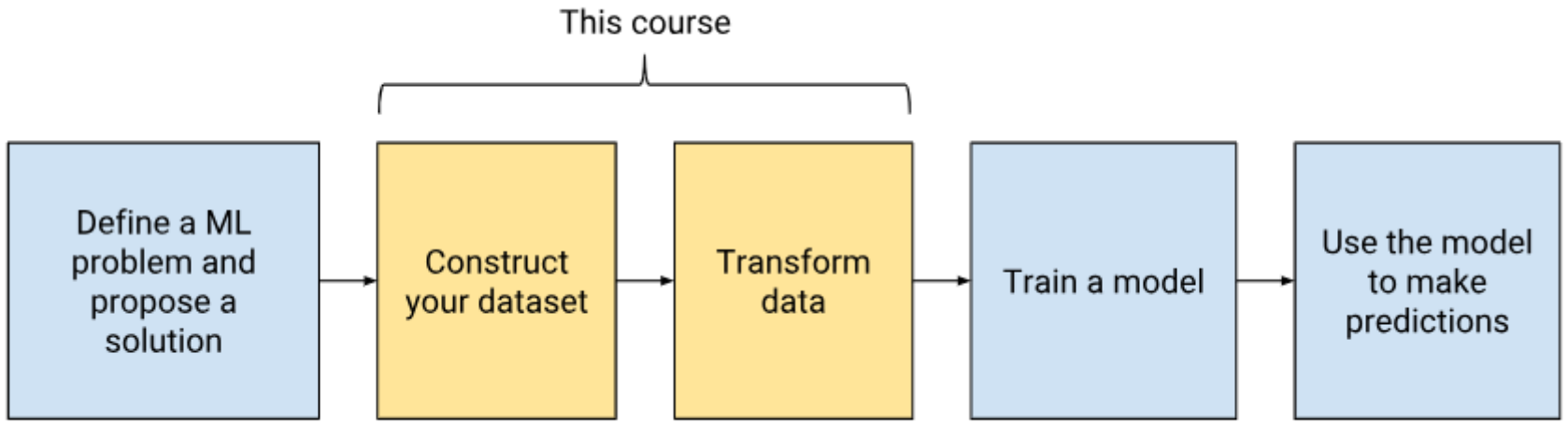
- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction

Data Preparation

The broader philosophy of data preparation is to discover how to best expose the underlying structure of the problem to the learning algorithms.

This is the guiding light.

Data Preparation and Feature Engineering in ML



Why is data Preparation important ?

- It helps to provide reliable prediction outcomes in various analytics operations.
- It helps identify data issues or errors and significantly reduces the chances of errors.
- It increases decision-making capability.
- It reduces overall project cost (data management and analytic cost).
- It helps to remove duplicate content to make it worthwhile for different applications.
- It increases model performance.

- DATA CLEANING –removing noise and inconsistencies in the data
- DATA INTEGRATION involves merging data from different sources
- DATA NORMALIZATION is the process of organizing database to remove redundancy
- DATA REDUCTION is the process of reducing data size by eliminating redundant features

Handling Text and Categorical Attributes

```
>>> housing_cat = housing[["ocean_proximity"]]
>>> housing_cat.head(10)
       ocean_proximity
17606        <1H OCEAN
18632        <1H OCEAN
14650       NEAR OCEAN
3230            INLAND
3555         <1H OCEAN
19480           INLAND
8879         <1H OCEAN
13685           INLAND
4937         <1H OCEAN
4861         <1H OCEAN
```

```
>>> housing_cat_1hot.toarray()
array([[1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       ...,
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

Once again, you can get the list of categories using the encoder's `categories_` instance variable:

```
>>> cat_encoder.categories_
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

Custom Transformers

- fit()
- (returning self),
- transform(),

And

- fit_transform().

Feature Scaling

There are two common ways to get all attributes to have the same scale:

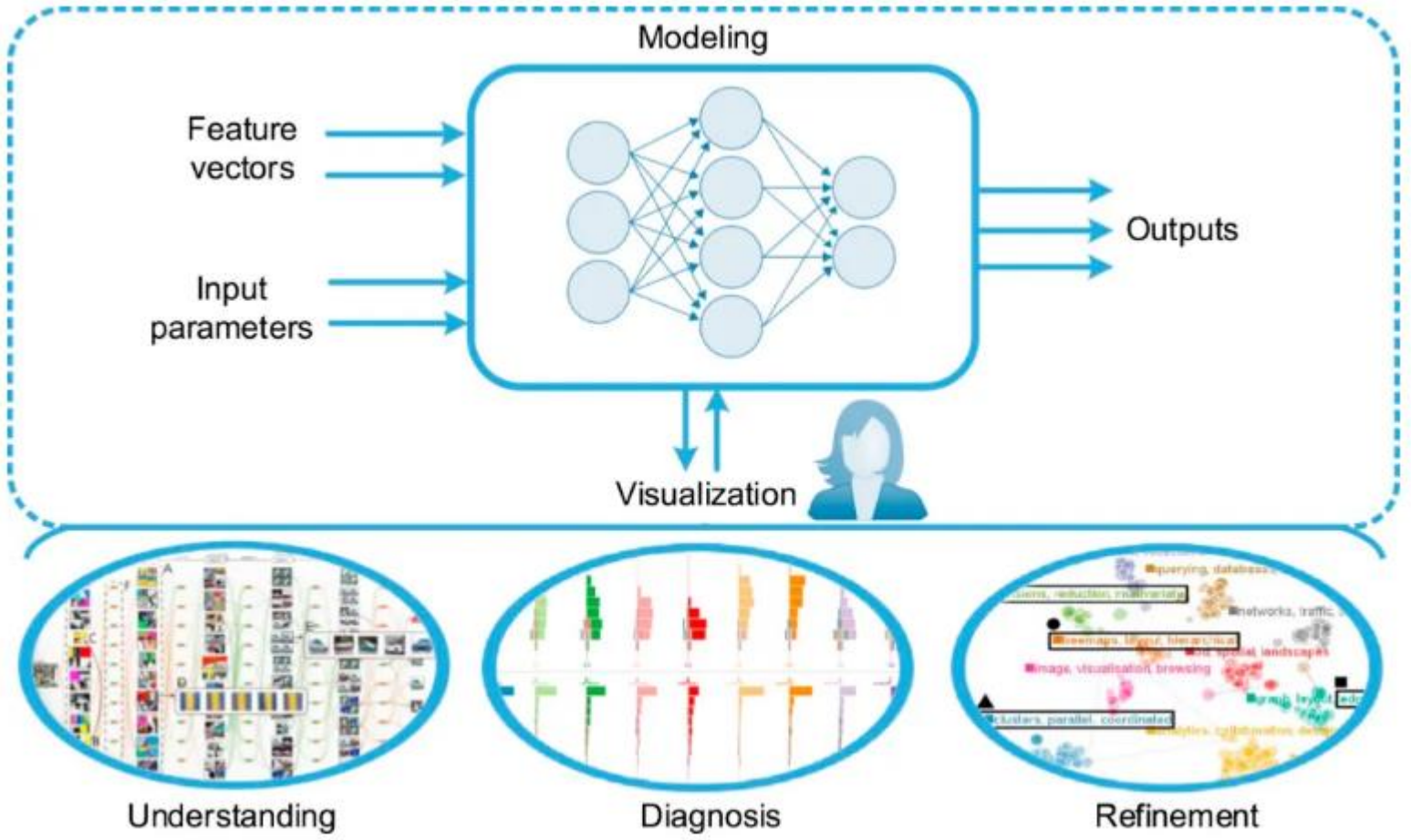
- *min-max scaling*
- *standardization*

Transformation Pipelines

One definition of an ML pipeline is **a means of automating the machine learning workflow by enabling data to be transformed and correlated into a model that can then be analyzed to achieve outputs**.

This type of ML pipeline makes the process of inputting data into the ML model fully automated.

Visualizing Machine Learning Models



Need for Visualization

Why do we want to visualize models?

- **Explainability**
- **Debugging & improvements**
- **Comparison & selection**
- **Teaching concepts**

Who should use visualization ?

- **Data Scientists / Machine Learning Engineers**
- **Model users**

What can we visualize?

- **Model architecture**
- **Learned parameters**
- **Model metrics**

When is visualization most relevant?

- **During training**
- **After training**

Where is visualization applied?

- **Application domains & models**
- **Research & development**

Metrics to Evaluate your Machine Learning Algorithm

- *Classification Accuracy*
- *Logarithmic Loss*
- *Confusion Matrix*
- *Area under Curve*
- *F1 Score*
- *Mean Absolute Error*
- *Mean Squared Error*

Classification Accuracy

Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Logarithmic Loss

Logarithmic Loss or Log Loss, works by penalizing the false classifications. It works well for multi-class classification. When working with Log Loss, the classifier must assign probability to each class for all the samples. Suppose, there are N samples belonging to M classes, then the Log Loss is calculated as below :

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$

where,

y_ij, indicates whether sample i belongs to class j or not

p_ij, indicates the probability of sample i belonging to class j

Log Loss has no upper bound and it exists on the range [0, ∞).

Log Loss nearer to 0 indicates higher accuracy, whereas if the Log Loss is away from 0 then it indicates lower accuracy.

In general, minimising Log Loss gives greater accuracy for the classifier.

Confusion Matrix

Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model.

Lets assume we have a binary classification problem. We have some samples belonging to two classes : YES or NO. Also, we have our own classifier which predicts a class for a given input sample. On testing our model on 165 samples ,we get the following result.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

There are 4 important terms :

True Positives : The cases in which we predicted YES and the actual output was also YES.

True Negatives : The cases in which we predicted NO and the actual output was NO.

False Positives : The cases in which we predicted YES and the actual output was NO.

False Negatives : The cases in which we predicted NO and the actual output was YES.

Accuracy for the matrix can be calculated by taking average of the values lying across the “main diagonal” i.e

$$Accuracy = \frac{TruePositive + TrueNegative}{TotalSample}$$

$$\therefore Accuracy = \frac{100 \dot{+} 50}{165} = 0.91$$

Confusion Matrix forms the basis for the other types of metrics.

Area Under Curve

Area Under Curve(AUC) is one of the most widely used metrics for evaluation. It is used for binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. Before defining AUC, let us understand two basic terms :

- True Positive Rate (Sensitivity) : True Positive Rate is defined as TP/ (FN+TP). True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.

$$TruePositiveRate = \frac{TruePositive}{FalseNegative + TruePositive}$$

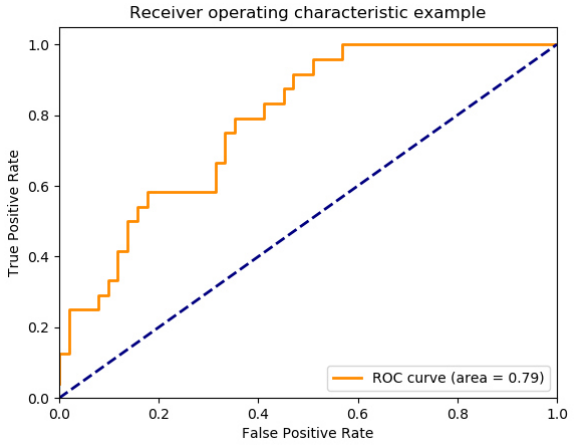
- True Negative Rate (Specificity) : True Negative Rate is defined as TN / (FP+TN). False Positive Rate corresponds to the proportion of negative data points that are correctly considered as negative, with respect to all negative data points.

$$TrueNegativeRate = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

False Positive Rate : False Positive Rate is defined as FP / (FP+TN). False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

$$FalsePositiveRate = \frac{FalsePositive}{TrueNegative + FalsePositive}$$

False Positive Rate and True Positive Rate both have values in the range [0, 1].
FPR and TPR both are computed at varying threshold values such as (0.00, 0.02, 0.04, …., 1.00) and a graph is drawn. AUC is the area under the curve of plot False Positive Rate vs True Positive Rate at different points in [0, 1].



As evident, AUC has a range of [0, 1]. The greater the value, the better is the performance of our model.

F1 Score

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).

High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as :

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

F1 Score tries to find the balance between precision and recall.

Precision : It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Recall : It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive)