**Seminar 6**

**Put the data files in the same folder as your Jupyter Notebook files if there are data files.**

**We learn coding by imitation. Therefore, we start by copying example codes and run them. Based on the outputs, comments, and the codes, we understand what the codes need and what the codes produce. Then we can modify the codes and apply them to new data for solving new problems.**

1. **Try machine learning models: Start a new Jupyter Notebook and copy the following codes one (line/part) by one, followed by press the keys Shift + Enter.**

# Machine Learning

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

## Ensemble strategies

### Random Forests and Extra-Trees

#### Classification

from sklearn import datasets

iris = datasets.load\_iris()

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(iris.data,

iris.target, test\_size=0.2, random\_state=0)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

import numpy as np

from sklearn.model\_selection import cross\_val\_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import ExtraTreesClassifier

hypothesis = RandomForestClassifier(random\_state=101)

scores = cross\_val\_score(hypothesis, X\_train, Y\_train,

cv=3, scoring='accuracy')

print ("RandomForestClassifier -> cross validation accuracy: mean = %0.3f std = %0.3f" % (np.mean(scores), np.std(scores)))

hypothesis = ExtraTreesClassifier(random\_state=101)

scores = cross\_val\_score(hypothesis, X\_train, Y\_train, cv=3,

scoring='accuracy')

print ("ExtraTreesClassifier -> cross validation accuracy: mean = %0.3f std = %0.3f" % (np.mean(scores), np.std(scores)))

#### Regression

from sklearn.datasets import fetch\_california\_housing

cali = fetch\_california\_housing()

cali

import numpy as np

feature\_names = np.array(cali.feature\_names)

feature\_names

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(cali.data,

cali.target, test\_size=0.2, random\_state=0)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

import numpy as np

from sklearn.ensemble import RandomForestRegressor

hypothesis = RandomForestRegressor(n\_estimators=30, random\_state=101)

scores = cross\_val\_score(hypothesis, X\_train, Y\_train, cv=3,

scoring='neg\_mean\_absolute\_error')

print ("RandomForestClassifier -> cross validation accuracy (neg\_mean\_absolute\_error): mean = %0.3f std = %0.3f" % (np.mean(scores), np.std(scores)))

hypothesis.fit(X\_train, Y\_train)

Y\_pred = hypothesis.predict(X\_test)

from sklearn.metrics import mean\_absolute\_error

print ("MAE", mean\_absolute\_error(Y\_test, Y\_pred))

# Feature importance

# feature\_importances\_ only works for tree based models

importance = hypothesis.feature\_importances\_

indices = np.argsort(importance)[::-1]

range1 = range(len(importance[indices]))

import matplotlib.pyplot as plt

plt.figure()

plt.title("Random Forest importance")

plt.barh(range1,importance[indices][::-1])

plt.yticks(range1, feature\_names[indices][::-1])

plt.ylim([-1, len(range1)])

plt.show()

1. **Try case – IPO case - 1: Start a new Jupyter Notebook and copy the following codes one (line/part) by one, followed by press the keys Shift + Enter.**

# IPO analysis and trading strategies

# An IPO is the process whereby a private company becomes a public company.

# Public offerings raise capital for the company and give the general public an

# opportunity to invest in the company by buying its shares.

# Though there are variations in how this occurs, in a typical offering, a company

# enlists the help of one or more investment banks to underwrite their offering.

# This means that the banks make a guarantee to the company that they will

# purchase all of the shares being offered at the IPO price on the day of the IPO.

# The underwriters, of course, do not intend to keep all of the shares themselves.

# With the help of the offering company, they go on what's called a roadshow to

# drum up interest from institutional clients. These clients put in a subscription

# for the shares, which indicates their interest in buying shares on the day of the

# IPO. This is a non-binding contract, as the price of the offering is not finalized

# until the day of the IPO. The underwriter will then set the offer price, given the

# level of interest expressed.

# What is interesting from our perspective is that research has consistently shown

# a systematic underpricing of IPOs. There are a number of theories as to why this

# happens, and why this level of underpricing seems to vary over time, but studies

# have shown that billions of dollars are left on the table every year.

# In an IPO, money left on the table, is the difference between the offering price

# of shares and the first day's closing price.

# While you can occasionally get in on the deal through your broker and receive the

# IPO at its offering price, in nearly all instances, you, as a member of the general

# public, will have to purchase the IPO at the (typically higher) opening price.

## Load data

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('IPO\_clean.csv')

df.head()

# The y is the target, which is 1 when the '1st Day Open to Close % Chg' is higher than a threshold, i.e., 2.5%.

## EDA

summary\_by\_year\_1 = df.groupby('Year')['1st Day % Chg'].describe()

summary\_by\_year\_1

# We get a quick summary of the performance of the stocks over the past 18 years.

# From the table, we can see the extraordinary average return of the IPO market in

# 2000. At over 35%, it is more than double any other year on the list. Also

# notable is the fact that every year has had a positive average return for first-day

# performance.

# The important point about these numbers is that they ('1st Day % Chg') are not the first-day

# performance that the general investing public could expect to receive on that

# first-day. Only investors who got in on the offering could expect to see these

# numbers.

# The first-day return that the general public could expect to receive would be the

# difference between the opening price and the closing price ('1st Day Open to Close % Chg').

summary\_by\_year\_2 = df.groupby('Year')['1st Day Open to Close % Chg'].describe()

summary\_by\_year\_2

# The y is the target, which is 1 when the '1st Day Open to Close % Chg' is higher than a threshold, i.e., 2.5%.

summary\_by\_year\_3 = df.groupby('Year')['y'].sum()

summary\_by\_year\_3