**Seminar 7**

**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

### Bagging

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 BaggingClassifier

from sklearn.neighbors import KNeighborsClassifier

hypothesis = BaggingClassifier(KNeighborsClassifier())

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

scoring='accuracy')

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

### Boost Models

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)

from sklearn.ensemble import AdaBoostClassifier

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

import numpy as np

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

scores = cross\_val\_score(hypothesis, X\_train, Y\_train, cv=3, scoring='accuracy')

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

from sklearn.ensemble import GradientBoostingClassifier

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

import numpy as np

hypothesis = GradientBoostingClassifier(max\_depth=5, n\_estimators=30, random\_state=101)

scores = cross\_val\_score(hypothesis, X\_train, Y\_train, cv=3, scoring='accuracy')

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

1. **Try Case – Firm Credit 2: Open and run the Jupyter Notebook “Seminar 3.2 Case-Firm-Credit-1” in the previous seminar. Click the menu “Cell -> Run All” and wait for it finishes running. In the end of the file, copy the following codes one (line/part) by one, followed by press the keys Shift + Enter.**

## Classification with multiple labels

### Data Preprocessing

# Make data preprocessing: taking the 'Creditscoreindicator' as the target variable 'Y'; keeping remaining numerical data (except ‘Creditscore’) as 'X'; filling missing values by feature means; using 20% of data as the test set and '101' as the random state; doing StandardScaler to X.

data=data\_all.drop(columns=['Creditscore', 'SMEindicator'])

target = data.pop('Creditscoreindicator')

data=data.fillna(data.mean())

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data, target, test\_size=0.2, random\_state=101)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

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

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

### AdaBoost

# Use Ada Boost Classification (with 3 estimators and a random state of 101) to fit the data. Use the test set to obtain a classification report.

from sklearn.ensemble import AdaBoostClassifier

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

import numpy as np

hypothesis = AdaBoostClassifier(n\_estimators=3, random\_state=101)

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

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

from sklearn.metrics import classification\_report

print (classification\_report(Y\_test, Y\_pred))

### Feature importance

feature\_names=data.columns

importance = hypothesis.feature\_importances\_

indices = np.argsort(importance)

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

plt.figure()

plt.title("AdaBoostClassifier Feature Importance")

plt.barh(range1,importance[indices])

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

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

plt.show()