```
https://colab.research.google.com/drive/1jrR2jWUN8JMm4DOstQkLOXmUH7TNEzRI?usp=sharing
```

Practical 7

Tutorial 1. Voting

The objectives of this practical are to learn:

- the core concepts of voting classifiers
- the core concepts of adaptive boosting
- how to tune the parameters in a AdaBoost classifier
- how to evaluate the performacen of a classifier using cross-validation.

In this tutorial, we form an ensemble of classifers that use different ML algorithms on the same dataset in a voting classifier.

Note that there is no data randomisation in a voting classifier.

Voting can be either hard or soft voting.

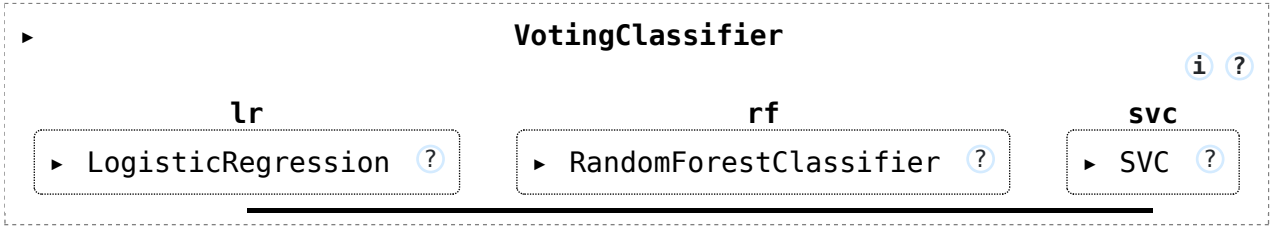
We will also use a performace metrics, accuracy_score, to evaluate the performace of each individual classifier and the voting ensemble classifier.

Fpr further information about the voting classifier, refer to:

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

```
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

X, y = make_moons(n_samples = 100, noise = 0.25)
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=1) # the default split is 75:25

log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()

voting_clf = VotingClassifier(estimators = [('lr', log_clf), ('rf', rnd_clf), ('svc',svm_clf)], voting = 'harc
voting_clf.fit(X_train, y_train)
```




```
for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
LogisticRegression 0.8
RandomForestClassifier 0.88
SVC 0.92
VotingClassifier 0.92
```

Tutorial 2. AdaBoost

In this tutorial, we use the iris dataset provided by scikit-learn. For more information, refer to:

In this tutorial, we use the iris dataset provided by scikit-learn. For more information, refer to:

https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html?highlight=iris%20dataset

https://en.wikipedia.org/wiki/Iris_flower_data_set

For cross validation in sklearn, refer to:

https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

We will focus on a boosting ensemble method called AdaBoost. For further information on the ensemble methods in sklearn used in this tutorial, refer to:

https://scikit-learn.org/stable/modules/ensemble.html

By default, weak learners in an AdaBoost classifier are decision stumps. Different weak learners can be specified through the base_estimator parameter, though the base classifier needs one of those that are able to return the probabilities with classes.

```
# import both the dataset and the classifiers
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_val_score

iris = load_iris()

X, y = iris.data, iris.target # use all features

# use the decision tree classifier
clf = DecisionTreeClassifier() # to full depth

scores = cross_val_score(clf, X, y, cv=5)
scores.mean()
```

For further information about the AdaBoostClassifier, refer to:

For further information about the AdaBoostClassifier, refer to:

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

```
# use the AdaBoost classifier with the default base classifier - DecisionTreeClassifier(max_depth=1)
clf = AdaBoostClassifier(n_estimators=100)

# the train-test split is done in each iteration of cross validation
scores = cross_val_score(clf, X, y, cv=5)
scores.mean()
```

```
# use the AdaBoost classifier with the logistic regression classifier

log_clf = LogisticRegression()

clf = AdaBoostClassifier(log_clf,
    n_estimators=100)

# the train-test split is done in each iteration of cross validation
scores = cross_val_score(clf, X, y, cv=5)
print(scores.mean())

# use the AdaBoost classifier with the DecisionTreeClassifier(max_depth=1) and a learning_rate

clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2),
    n_estimators=100, learning_rate=1.5)

# the train-test split is done in each iteration of cross validation
scores = cross_val_score(clf, X, y, cv=5)
scores.mean()
```

```
# use the AdaBoost classifier with the random forest classifier

rnd_clf = RandomForestClassifier()

clf = AdaBoostClassifier(rnd_clf
```

```
clf = AdaBoostClassifier(rnd_clf,
    n_estimators=100)

# the train-test split is done in each iteration of cross validation
scores = cross_val_score(clf, X, y, cv=5)
print(scores.mean())

# use the AdaBoost classifier with the random forest classifier and a learning_rate

clf = AdaBoostClassifier(rnd_clf,
    n_estimators=100, learning_rate=1.5)

# the train-test split is done in each iteration of cross validation
scores = cross_val_score(clf, X, y, cv=5)
scores.mean()
```

Exercise 1. Parameter tuning in the AdaBoostClassifier function

As we have seen in the above tutorials there are parameters in the AdaBoostClassifier function which need to be tuned. Study the user guide on the function to undertand what these paramters are for and how their values can affect the classification results.

- What does learning_rate represents? Run AdaBoostClassifier with the DecisionTreeClassifer as the base classifier on the iris dataset with learning_rate = 0.75 and compare the evaluation results of the ensemble with learning_rate = 1.5.

Use the remainder of this practical to work on either the individual assignment or the group project.

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
```

```
# Dữ liệu iris
iris = load_iris()
X, y = iris.data, iris.target

# Base classifier
base_clf = DecisionTreeClassifier(max_depth=1, random_state=42)

# AdaBoost với learning_rate=0.75
clf1 = AdaBoostClassifier(estimator=base_clf, n_estimators=100, learning_rate=0.75, random_state=42)
scores1 = cross_val_score(clf1, X, y, cv=5)
acc1 = scores1.mean()
print("Accuracy với learning_rate=0.75:", acc1)

# AdaBoost với learning_rate=1.5
clf2 = AdaBoostClassifier(estimator=base_clf, n_estimators=100, learning_rate=1.5, random_state=42)
scores2 = cross_val_score(clf2, X, y, cv=5)
acc2 = scores2.mean()
print("Accuracy với learning_rate=1.5:", acc2)
```

```
Accuracy với learning_rate=0.75: 0.96
Accuracy với learning_rate=1.5: 0.9466666666666667
```