# Advanced Artificial Intelligence

# Lab 08

# Outline

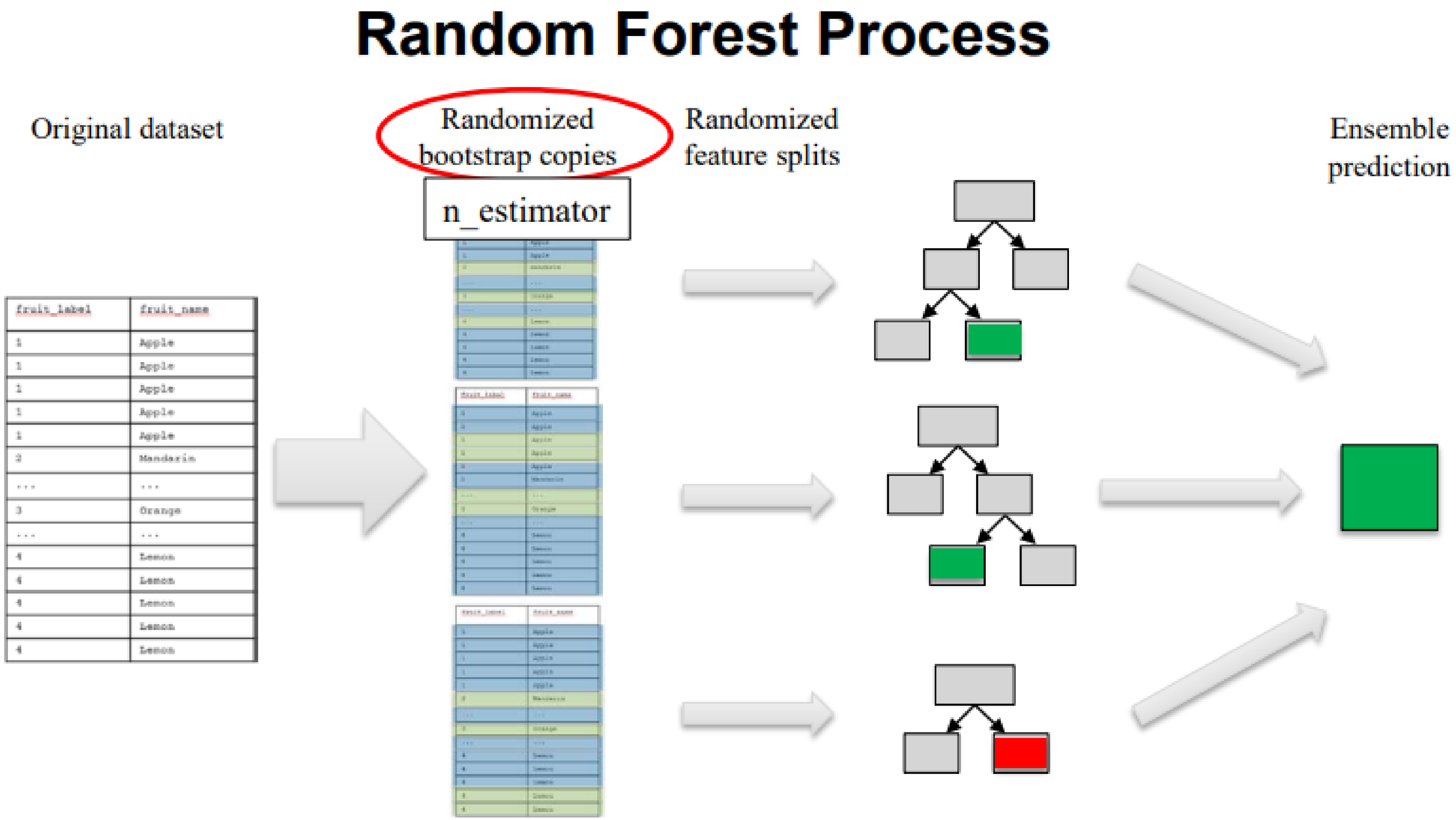
- More introduction to ensemble learning
- Exercise

# Random Forests

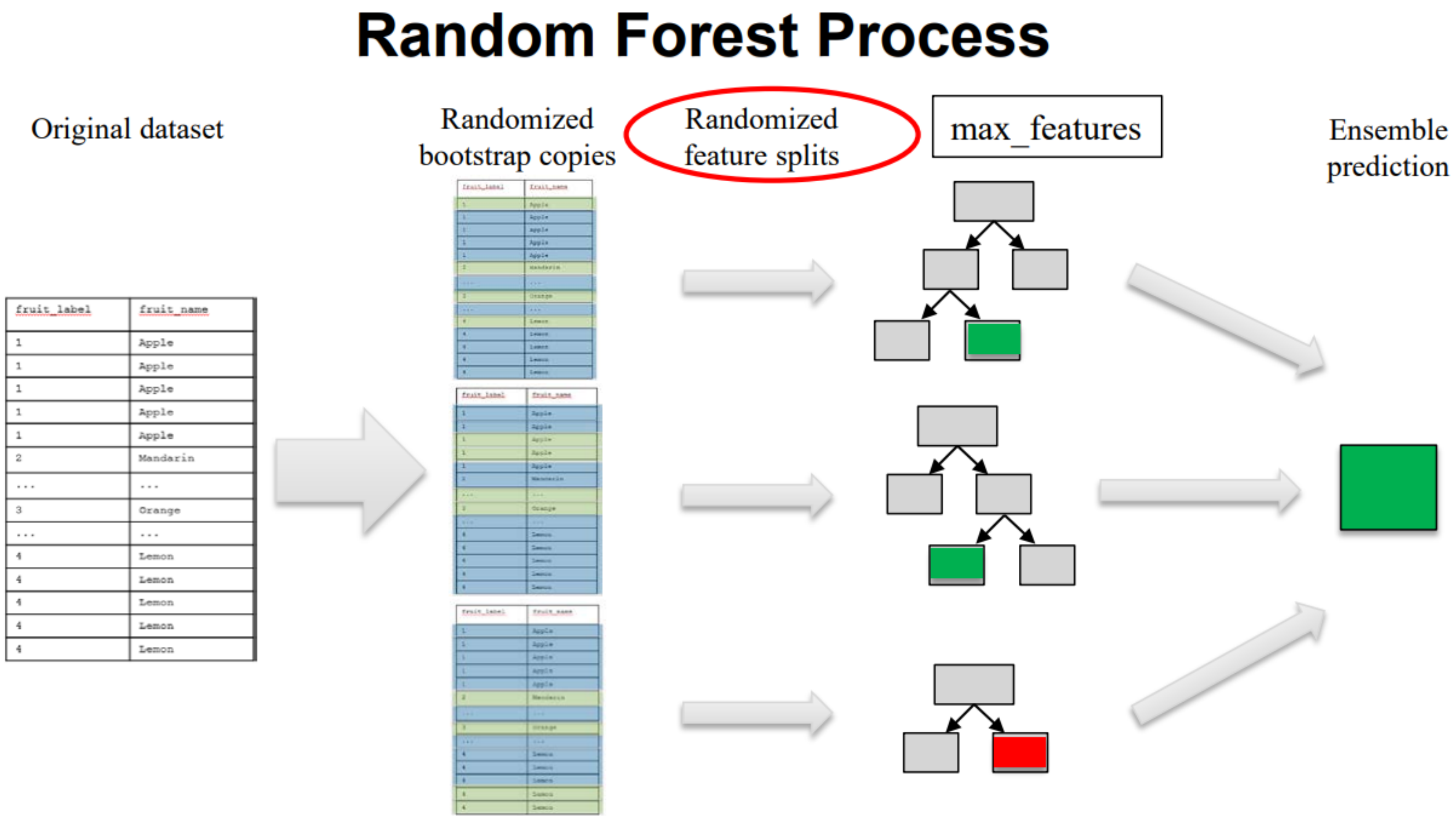
## Random Forests

- An ensemble of trees, not just one tree.
- Widely used, very good results on many problems.
- `sklearn.ensemble` module:
  - *Classification:* `RandomForestClassifier`
  - *Regression:* `RandomForestRegressor`
- One decision tree → Prone to overfitting.
- Many decision trees → More stable, better generalization
- Ensemble of trees should be diverse: introduce random variation into tree-building.

# Random Forests



# Random Forests



# Random Forests

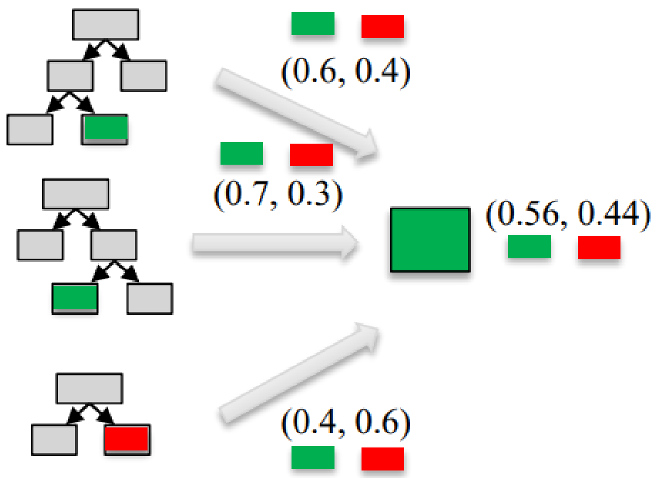
## Random Forest max_features Parameter

- **Learning is quite sensitive to max_features.**
- **Setting max_features = 1 leads to forests with diverse, more complex trees.**
- **Setting max_features = <close to number of features> will lead to similar forests with simpler trees.**

# Random Forests

## Prediction Using Random Forests

1. **Make a prediction for every tree in the forest.**
2. **Combine individual predictions**
   - *Regression: mean of individual tree predictions.*
   - *Classification:*
     - *Each tree gives probability for each class.*
     - *Probabilities averaged across trees.*
     - *Predict the class with highest probability.*

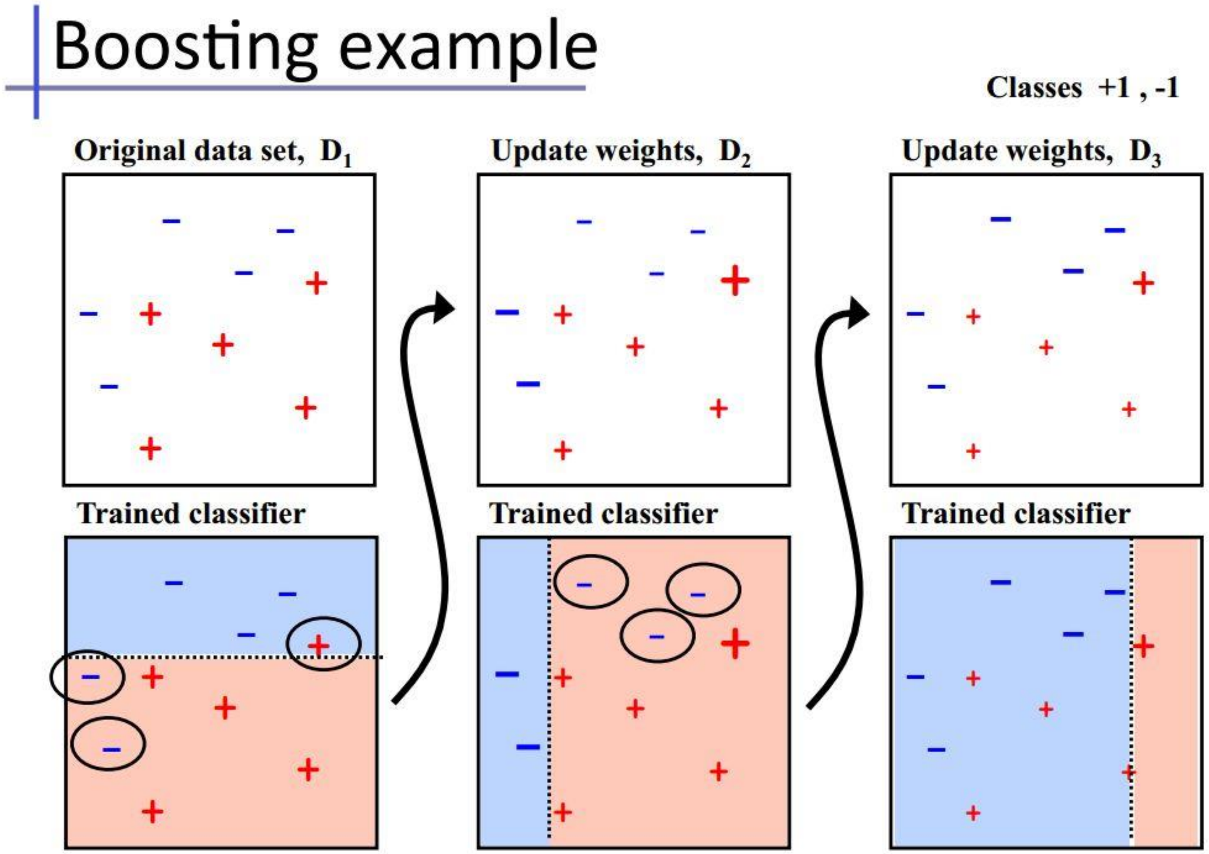


# Boosting

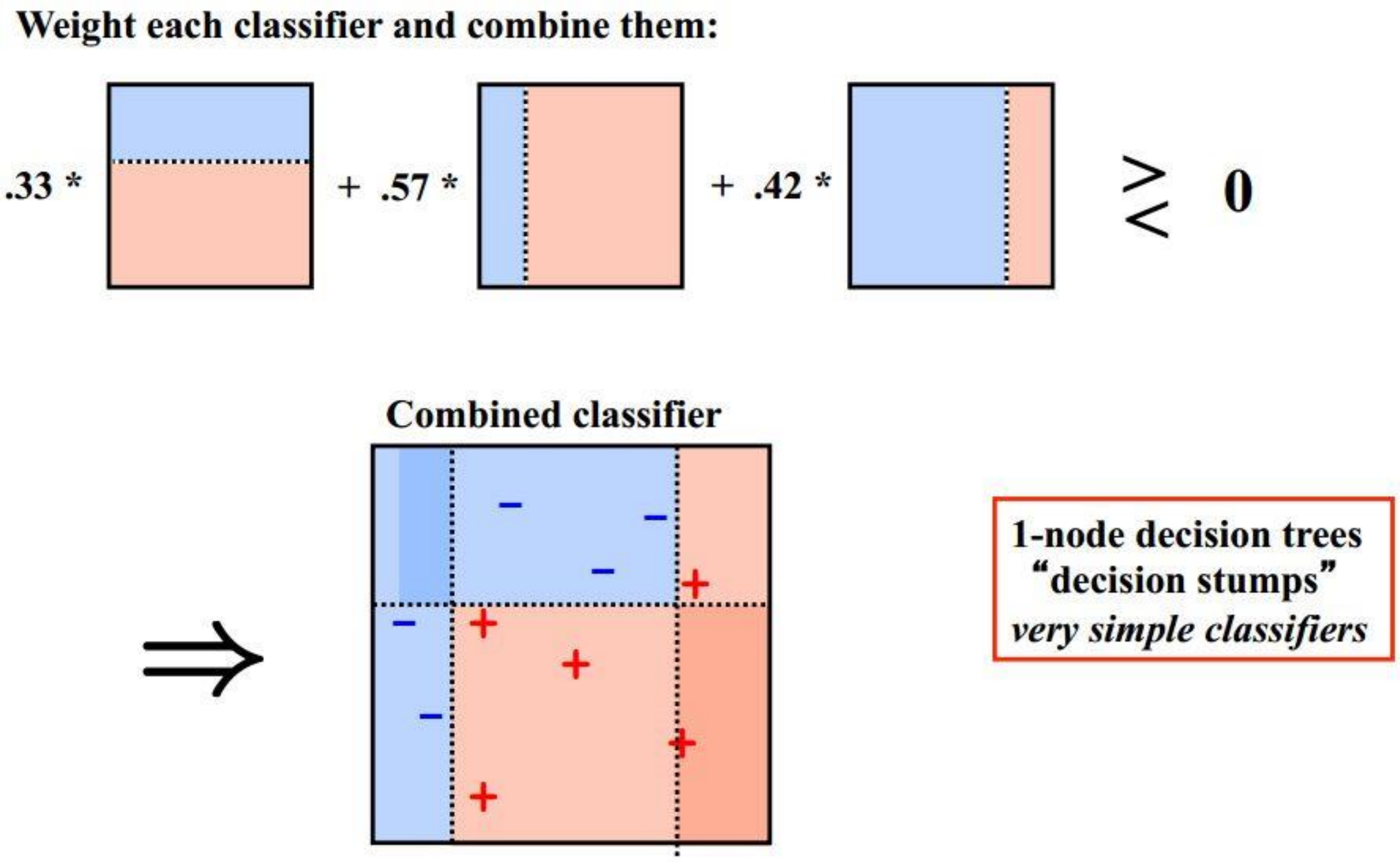
- **Pseudo-code**

**Input** the original data set $\mathscr{D}$
**Input** the number of bootstrap samples $k$
Number of training samples $N = |\mathscr{D}|$
Initialise the weights for samples $\mathbf{w} \leftarrow (\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N})$
**for** $i = 1$ to $k$ **do**
    Create a bootstrap sample $\mathscr{D}_i$ of size $N$ from $\mathscr{D}$ according to $\mathbf{w}$
    Train a base model on $\mathscr{D}_i$
    Increase the weights of incorrectly classified examples
    Reduce the weights of correctly classified examples
    Normalise $\mathbf{w}$
**end for**
Aggregating the trained base classifiers and use it as final ensemble model

# Boosting Example



# Boosting example



# Boosting: GBDT

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

rfc = GradientBoostingClassifier(learning_rate=1.0)
para ={'n_estimators':[88,99,100,111,122,133],'max_depth':[7,8,9,10,13,14]}
gscv = GridSearchCV(rfc,param_grid=para,cv=2)
gscv.fit(X_train,y_train)
predict = gscv.predict(X_test)
score = gscv.score(X_test,y_test)
# print(predict)
print(score)
print(gscv.best_params_)

```

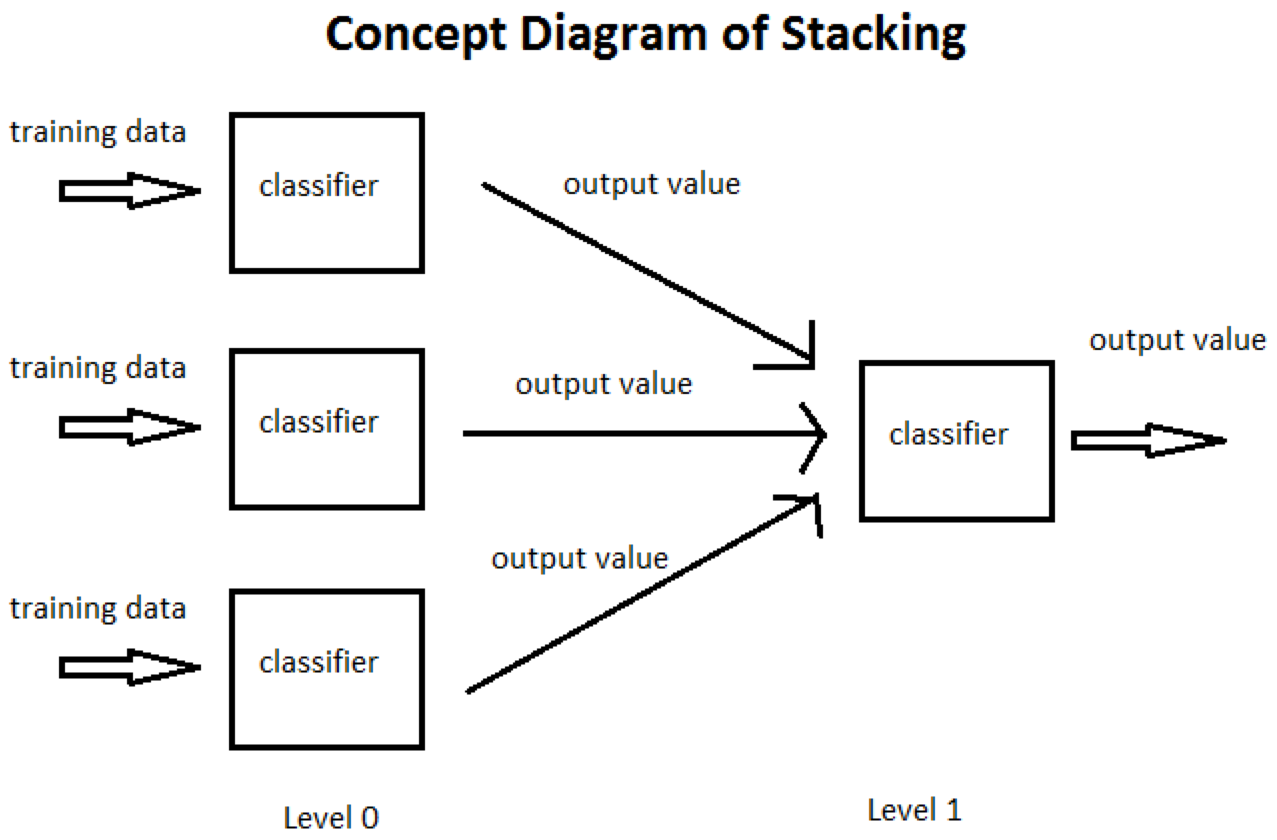
```
0.8977777777777778
{'max_depth': 7, 'n_estimators': 99}
```

# Stacking Algorithm

**Algorithm** Stacking

Input: training data $D = \{x_i, y_i\}_{i=1}^{m}$
Ouput: ensemble classifier $H$
*Step 1: learn base-level classifiers*
**for** $t = 1$ to $T$ **do**
    learn $h_t$ based on $D$
**end for**
*Step 2: construct new data set of predictions*
**for** $i = 1$ to $m$ **do**
    $D_h = \{x'_i, y_i\}$, where $x'_i = \{h_1(x_i), ..., h_T(x_i)\}$
**end for**
*Step 3: learn a meta-classifier*
learn $H$ based on $D_h$
return $H$

# Stacking Example



# Stacking Ensemble

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import StackingClassifier
X, y = load_iris(return_X_y=True)
estimators = [
    ('rf', RandomForestClassifier(n_estimators=122,max_depth=8, random_state=0)),
    ('ada',AdaBoostClassifier(n_estimators = 88, random_state=0),
    ('lr',LogisticRegression(random_state=1)),
    ( 'knn',KNeighborsClassifier(n_neighbors=4))

)
]
clf = StackingClassifier(
    estimators=estimators, final_estimator=LogisticRegression()
)

clf.fit(X_train, y_train).score(X_test, y_test)
```

0.9777777777777777

# Weighted Average Probabilities (Soft Voting)

To illustrate this with a simple example, let’s assume we have 3 classifiers and a 3-class classification problems where we assign equal weights to all classifiers: w1=1, w2=1, w3=1.

The weighted average probabilities for a sample would then be calculated as follows:

| classifier | class 1 | class 2 | class 3 |
|---|---|---|---|
| classifier 1 | w1 * 0.2 | w1 * 0.5 | w1 * 0.3 |
| classifier 2 | w2 * 0.6 | w2 * 0.3 | w2 * 0.1 |
| classifier 3 | w3 * 0.3 | w3 * 0.4 | w3 * 0.3 |
| weighted average | 0.37 | 0.4 | 0.23 |

# Weighted Average Probabilities (Soft Voting)

```python
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier

clf1 = LogisticRegression(random_state=1)
clf2 =  KNeighborsClassifier(n_neighbors=3)
clf3 = GaussianNB()

eclf = VotingClassifier(
    estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3)],
    voting='soft',weights=[2, 2, 1])

for clf, label in zip([clf1, clf2, clf3, eclf], ['Logistic Regression', 'Random Forest', 'Naive Bayes', 'Voting Ensemble']):
    clf.fit(X_train, y_train)
    score = clf.score(X_test,y_test)
    print("Accuracy: %0.4f, [%s] " % (score,label))
```

```
Accuracy: 0.9711, [Logistic Regression]
Accuracy: 0.9667, [Random Forest]
Accuracy: 0.7778, [Naive Bayes]
Accuracy: 0.9756, [Voting Ensemble]
```

# Exercise

- To implement the instances of the Section `Ensemble methods` in Sklearn and try different models as base learners for stacking algorithm in the above example.
- https://scikit-learn.org/stable/modules/ensemble.html#