Staking an blending

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv("E:\heart.csv")
```

In [3]:
```
df
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

In [4]:
```
X = df.drop(columns=['target'])
y = df['target']
```

In [5]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_st
```

In [6]:
```
print(X_train.shape)
```

```
(242, 13)
```

In [7]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
```

In [8]:
```
estimators = [
    ('rf', RandomForestClassifier(n_estimators=10, random_state=42)),
    ('knn', KNeighborsClassifier(n_neighbors=10)),
    ('gbdt',GradientBoostingClassifier())
]
```

In [9]:
```
from sklearn.ensemble import StackingClassifier

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

```
    cv=10
)
```

In [10]:
```
clf.fit(X_train, y_train)
```

Out[10]:
```
StackingClassifier(cv=10,
                   estimators=[('rf',
                                RandomForestClassifier(n_estimators=10,
                                                       random_state=42)),
                               ('knn', KNeighborsClassifier(n_neighbors=1
0)),
                               ('gbdt', GradientBoostingClassifier())],
                   final_estimator=LogisticRegression())
```

In [11]:
```
y_pred = clf.predict(X_test)
```

In [12]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[12]:
```
0.8688524590163934
```