Prepare Data

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

In [ ]:
```
df = pd.read_csv("employee_churn_data_clearned.csv")
df
```

Out[ ]:

	department_0	department_1	department_2	department_3	department_4	departme
0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	
2	0.0	1.0	0.0	0.0	0.0	
3	0.0	0.0	1.0	0.0	0.0	
4	0.0	0.0	1.0	0.0	0.0	
...	...	...	...	...	...	
9535	0.0	0.0	0.0	0.0	0.0	
9536	0.0	0.0	1.0	0.0	0.0	
9537	0.0	0.0	0.0	0.0	0.0	
9538	0.0	0.0	0.0	0.0	0.0	
9539	0.0	0.0	1.0	0.0	0.0	

9540 rows × 19 columns

In [ ]:
```
from sklearn.model_selection import train_test_split, GridSearchCV,cross_val_score,
```

In [ ]:
```
X = df.loc[:,df.columns != 'left']
y = df['left']
X_train ,X_test,y_train,y_test  = train_test_split(X,y,test_size=0.3,random_state=0
```

Voting Classifier

In [ ]:
```
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

Default parameters

In [ ]:
```
clf1 = KNeighborsClassifier()
clf2 = SVC()


voting_clf = VotingClassifier(
    estimators=[('SVC', clf2), ('KNeighborsClassifier', clf1)],
    voting='hard'
    )

score = cross_val_score(voting_clf, X_train, y_train, cv=5)
print("train score : ", score.mean())

voting_clf.fit(X_train, y_train)
print("test score : ", voting_clf.score(X_test, y_test))
```

```
train score :  0.8162620825764202
test score :  0.8074772886093641
```

With hyper parameters

In [ ]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
grid = {
    # "n_neighbors": [5, 10, 20],
    "n_neighbors": [20],
    "algorithm": ['auto'],
    # "leaf_size": [10, 20, 30, 40, 50],
    "leaf_size": [10],
    # "p": [0.01, 0.1, 1, 2, 10, 100],
    "p": [0.01, 0.1],
}
gcv_knb = GridSearchCV(KNeighborsClassifier(), grid)

grid = {
    "gamma" : ["scale","auto"],
}

gcv_svc = GridSearchCV(SVC(),grid)
```

In [ ]:
```
voting_clf = VotingClassifier(
    estimators=[('gcv_svc', gcv_svc), ('gcv_knb', gcv_knb)],
    voting='hard'
    )

score = cross_val_score(voting_clf, X_train, y_train, cv=5)

print("train score : ", score.mean())
voting_clf.fit(X_train, y_train)
print("test score : ", voting_clf.score(X_test, y_test))
```

```
train score :  0.8313859920608222
test score :  0.8245981830887491
```

Stacking Classifier

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

default parameters

In [ ]:
```
stack_clf = StackingClassifier(
    estimators=[('SVC', SVC()), ('KNeighborsClassifier', KNeighborsClassifier())],
    stack_method='predict',
    cv=10
    )

score = cross_val_score(stack_clf, X_train, y_train, cv=5)

print("train score : ", score.mean())
stack_clf.fit(X_train, y_train)
print("test score : ", stack_clf.score(X_test, y_test))
```

```
train score :  0.8168613335127498
test score :  0.8151642208245982
```

With hyper parameter

In [ ]:
```
grid = {
    # "n_neighbors": [5, 10, 20],
    "n_neighbors": [20],
    "algorithm": ['auto'],
    # "leaf_size": [10, 20, 30, 40, 50],
    "leaf_size": [10],
    # "p": [0.01, 0.1, 1, 2, 10, 100],
    "p": [0.01, 0.1],
}
gcv_knb = GridSearchCV(KNeighborsClassifier(), grid)

grid = {
    "gamma" : ["scale","auto"],
}

gcv_svc = GridSearchCV(SVC(),grid)
```

In [ ]:
```
stack_clf = StackingClassifier(
    estimators=[('gcv_svc', gcv_svc), ('gcv_knb', gcv_knb)],
    stack_method='auto'
    )

score = cross_val_score(stack_clf, X_train, y_train, cv=5)

print("train score : ", score.mean())
stack_clf.fit(X_train, y_train)
print("test score : ", stack_clf.score(X_test, y_test))
```

```
train score :  0.8382744623113323
test score :  0.8343815513626834
```

Baseline

In [ ]:
```
from sklearn.dummy import DummyClassifier
dummy_clf = DummyClassifier(strategy='most_frequent')
dummy_clf.fit(X_train, y_train)
dummy_clf.score(X_train, y_train)
```

Out[ ]: 0.7136867325546571

In [ ]:
```
dummy_clf = DummyClassifier(strategy='prior')
dummy_clf.fit(X_train, y_train)
dummy_clf.score(X_train, y_train)
```

Out[ ]: 0.7136867325546571

In [ ]:
```
dummy_clf = DummyClassifier(strategy='stratified')
dummy_clf.fit(X_train, y_train)
dummy_clf.score(X_train, y_train)
```

Out[ ]: 0.5847559149445942

In [ ]:
```
dummy_clf = DummyClassifier(strategy='uniform')
dummy_clf.fit(X_train, y_train)
dummy_clf.score(X_train, y_train)
```

Out[ ]: 0.4916142557651992