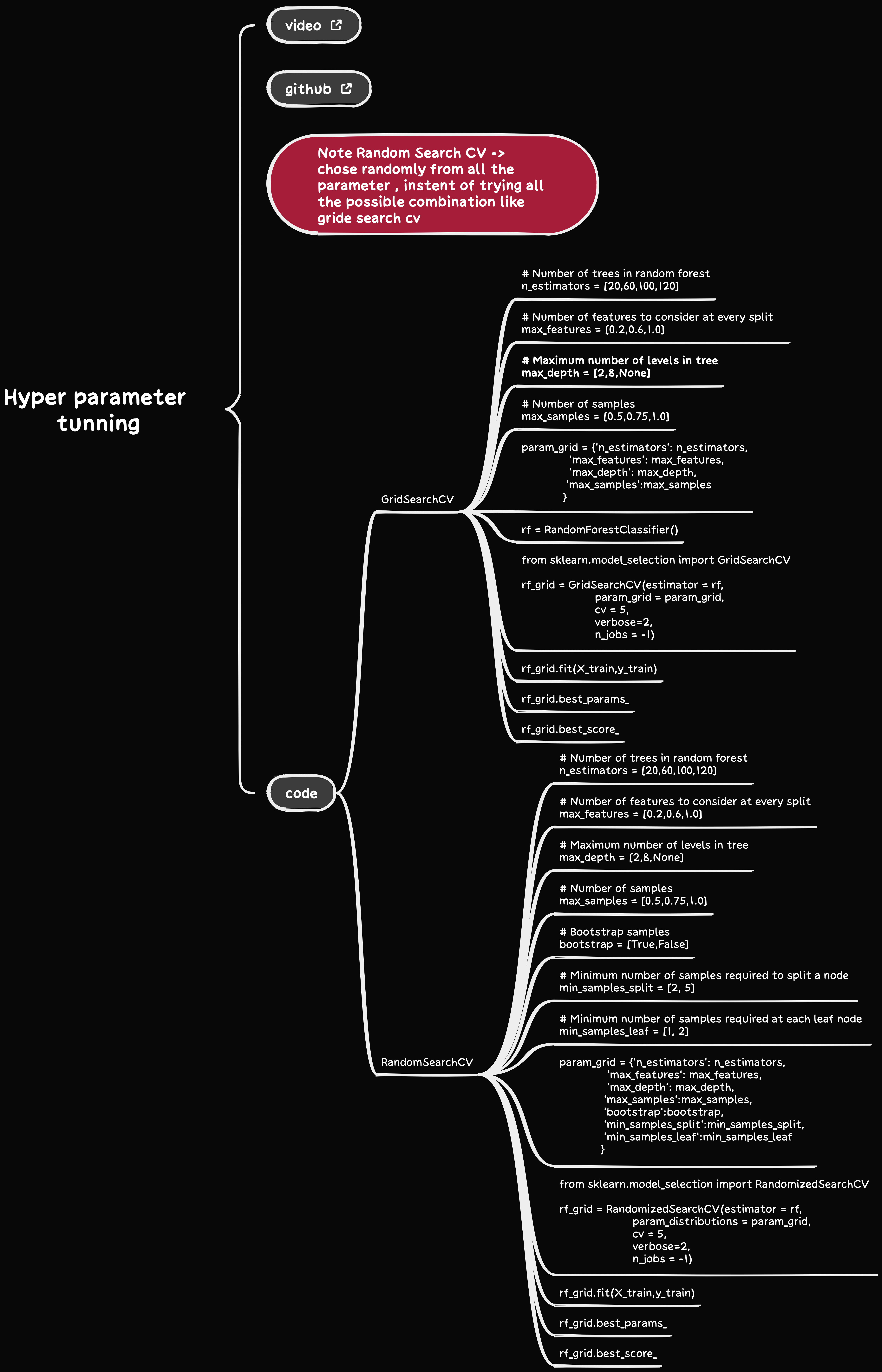

Hyper parameter tunning

- video
- github
- Note Random Search CV -> chose randomly from all the parameter , instent of trying all the possible combination like gride search cv
- code
  - GridSearchCV

```
# Number of trees in random forest
n_estimators = [20,60,100,120]
```

```
# Number of features to consider at every split
max_features = [0.2,0.6,1.0]
```

```
# Maximum number of levels in tree
max_depth = [2,8,None]
```

```
# Number of samples
max_samples = [0.5,0.75,1.0]
```

```
param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
             'max_samples':max_samples
            }
```

```
rf = RandomForestClassifier()
```

```
from sklearn.model_selection import GridSearchCV

rf_grid = GridSearchCV(estimator = rf,
                       param_grid = param_grid,
                       cv = 5,
                       verbose=2,
                       n_jobs = -1)
```

```
rf_grid.fit(X_train,y_train)
```

```
rf_grid.best_params_
```

```
rf_grid.best_score_
```

  - RandomSearchCV

```
# Number of trees in random forest
n_estimators = [20,60,100,120]
```

```
# Number of features to consider at every split
max_features = [0.2,0.6,1.0]
```

```
# Maximum number of levels in tree
max_depth = [2,8,None]
```

```
# Number of samples
max_samples = [0.5,0.75,1.0]
```

```
# Bootstrap samples
bootstrap = [True,False]
```

```
# Minimum number of samples required to split a node
min_samples_split = [2, 5]
```

```
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2]
```

```
param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
             'max_samples':max_samples,
             'bootstrap':bootstrap,
             'min_samples_split':min_samples_split,
             'min_samples_leaf':min_samples_leaf
            }
```

```
from sklearn.model_selection import RandomizedSearchCV

rf_grid = RandomizedSearchCV(estimator = rf,
                       param_distributions = param_grid,
                       cv = 5,
                       verbose=2,
                       n_jobs = -1)
```

```
rf_grid.fit(X_train,y_train)
```

```
rf_grid.best_params_
```

```
rf_grid.best_score_
```