Gépi tanulás python alapokon

Modell hangolás

2024.10.21.

Jónás Dániel
data scientist

Eddigi lépések

- Célváltozó kiválasztása
- Bemenő változók kiválasztása
- Train-test adathalmaz létrehozása
- Modell típus kiválasztása
- Fit
- Predict

```
y = df.loc[:, target_variable]
X = df.loc[:, input_variables]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
clf.predict(X_test)
```

Hiperparaméter

```
y = df.loc[:, target_variable]
X = df.loc[:, input_variables]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
clf.predict(X_test)
```

Hiperparaméter

```
y = df.loc[:, target_variable]
X = df.loc[:, input_variables]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
clf.predict(X_test)
```

Hiperparaméter

sklearn.tree.DecisionTreeClassifier ¶

class sklearn.tree.DecisionTreeClassifier(*, *criterion='gini'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *class_weight=None*, *ccp_alpha=0.0*) [source]

```
y = df.loc[:, target_variable]
X = df.loc[:, input_variables]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
clf.predict(X_test)
```

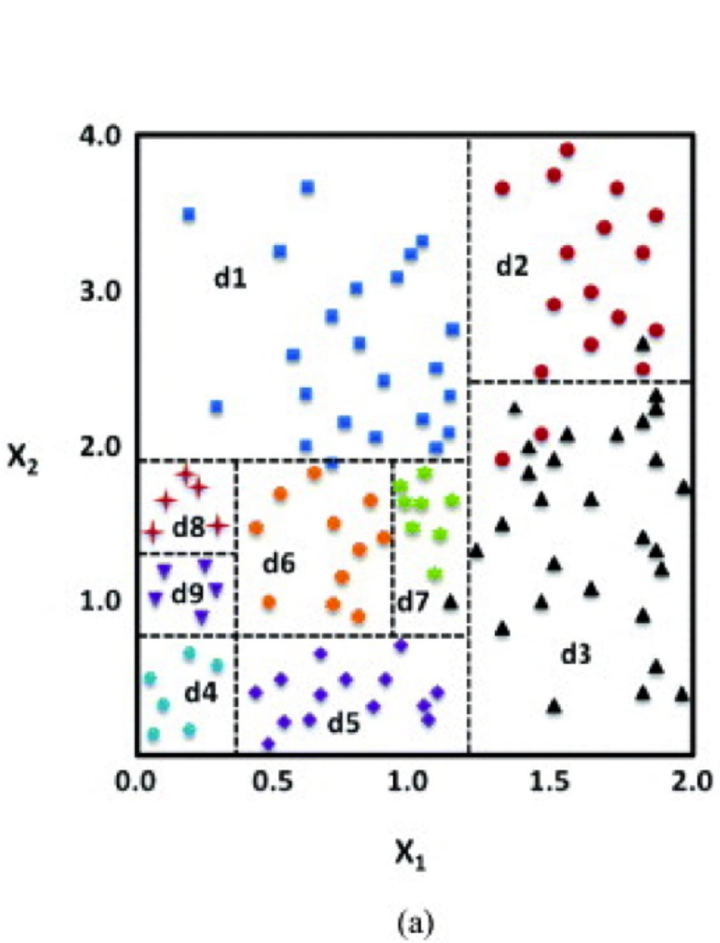
Hiperparaméter

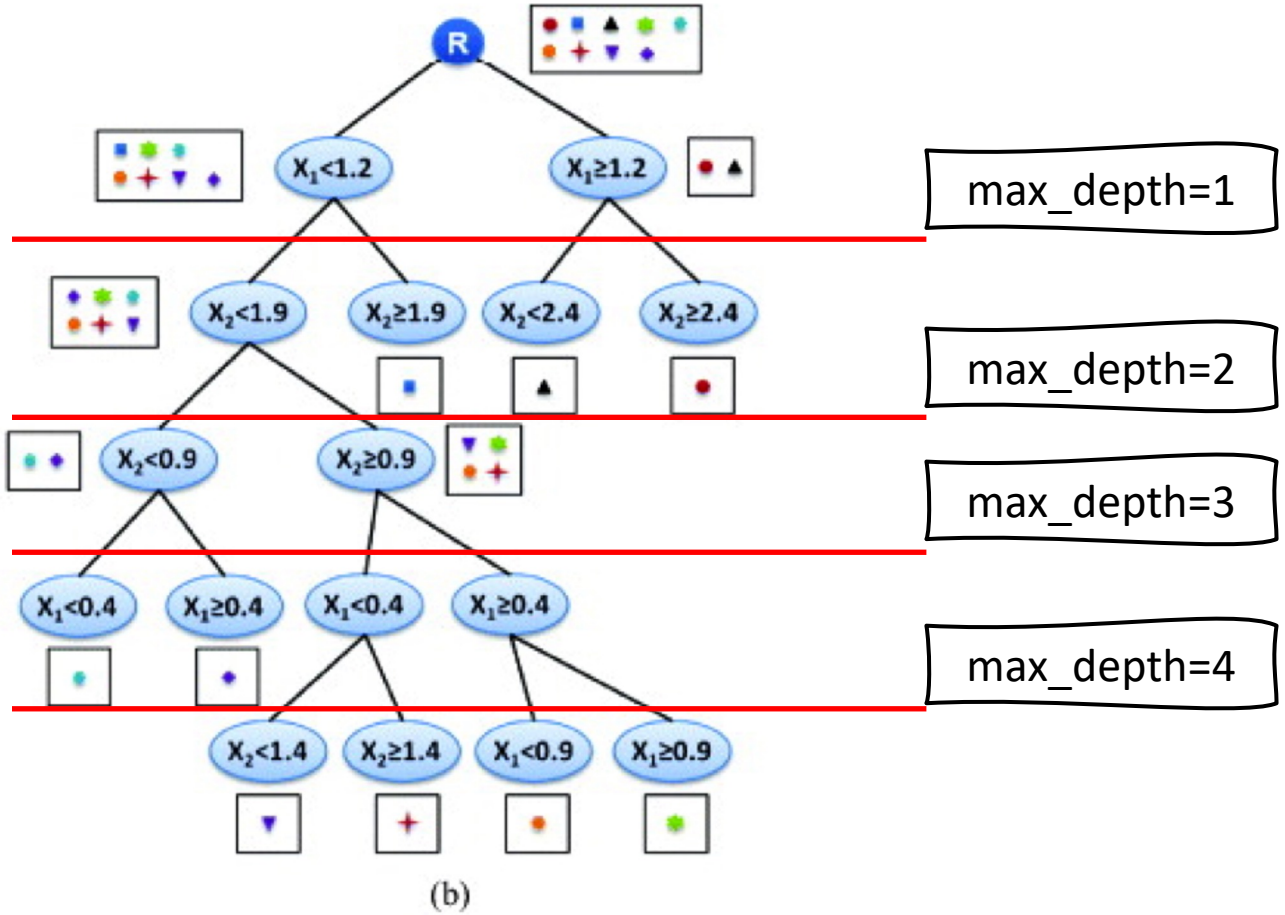
sklearn.tree.DecisionTreeClassifier ¶

class sklearn.tree.DecisionTreeClassifier(*, *criterion='gini'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *class_weight=None*, *ccp_alpha=0.0*) [source]

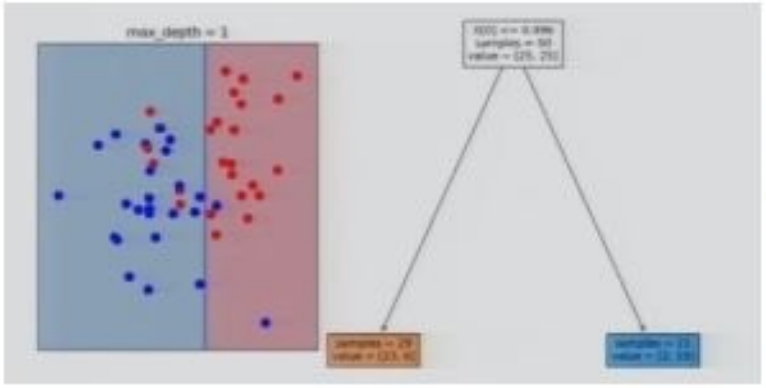
```
y = df.loc[:, target_variable]
X = df.loc[:, input_variables]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)
clf = DecisionTreeClassifier(max_depth=3)
clf.fit(X_train, y_train)
clf.predict(X_test)
```

Döntési fa – max_depth

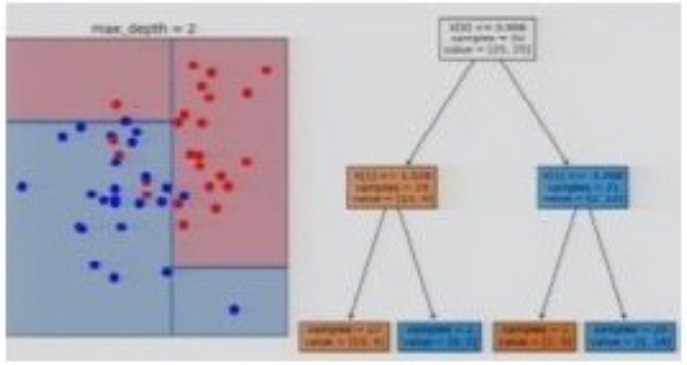




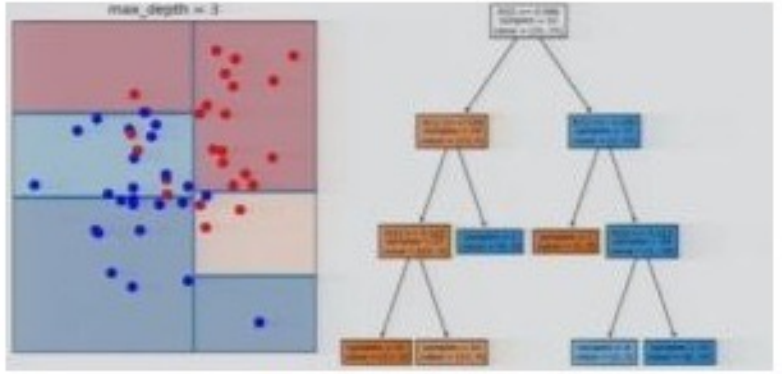
Döntési fa – max_depth



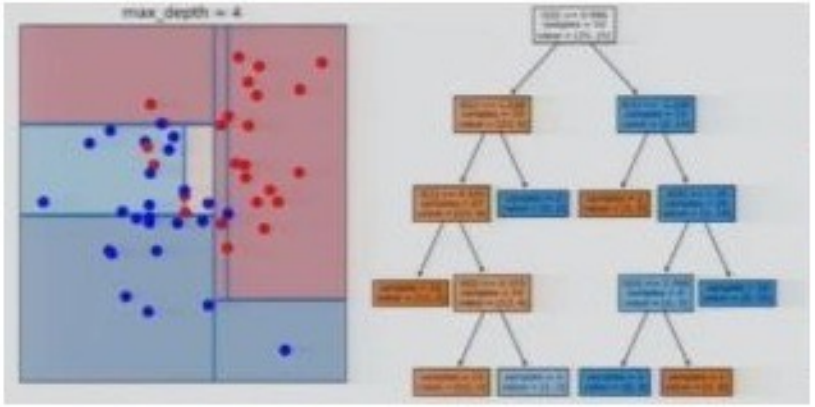
Max Depth = 1



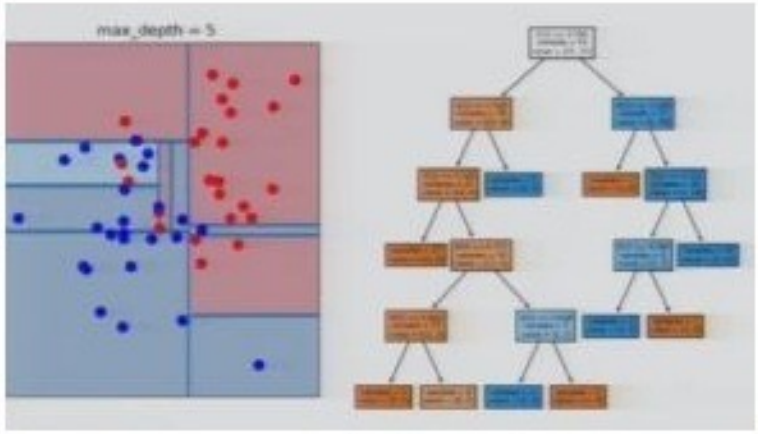
Max Depth = 2



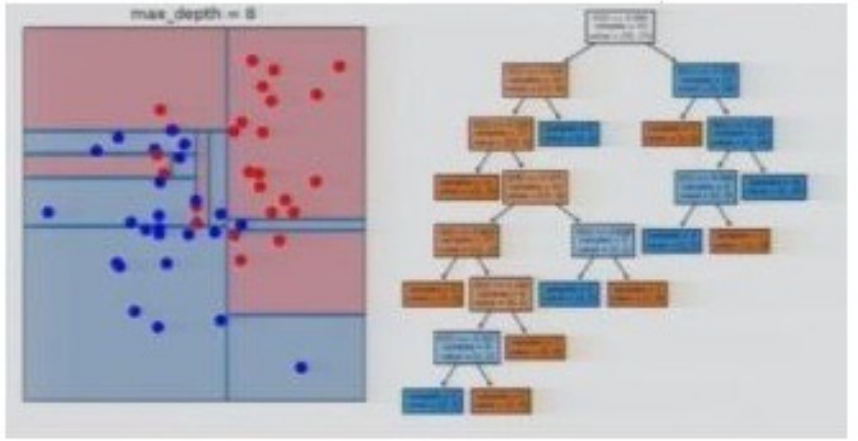
Max Depth = 3



Max Depth = 4



Max Depth = 5



Max Depth = 8

Döntési fa – max_depth

sklearn.tree.DecisionTreeClassifier ¶

class sklearn.tree.DecisionTreeClassifier(*, *criterion='gini'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *class_weight=None*, *ccp_alpha=0.0*) [source]

max_depth : *int, default=None*

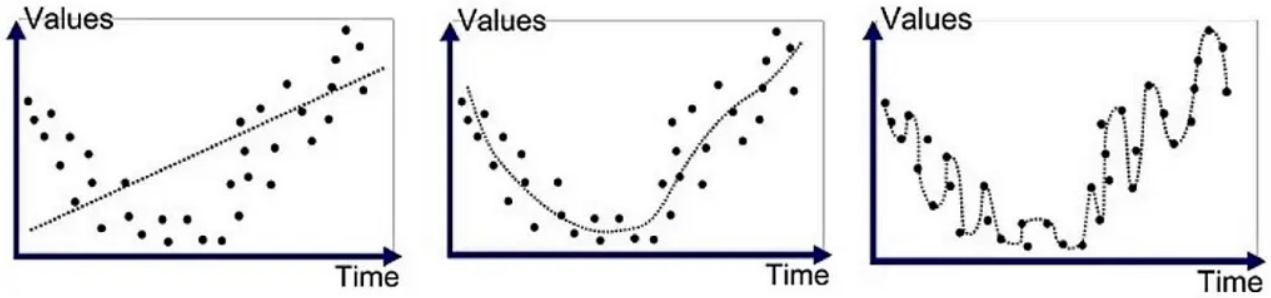
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*

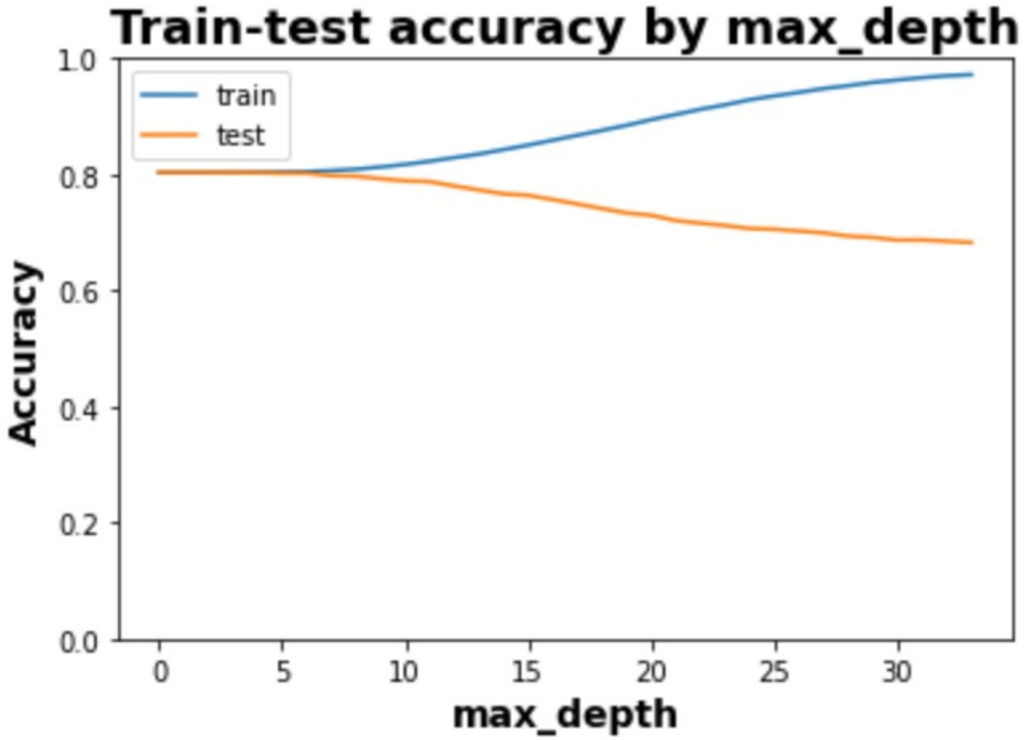
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Túltanulás

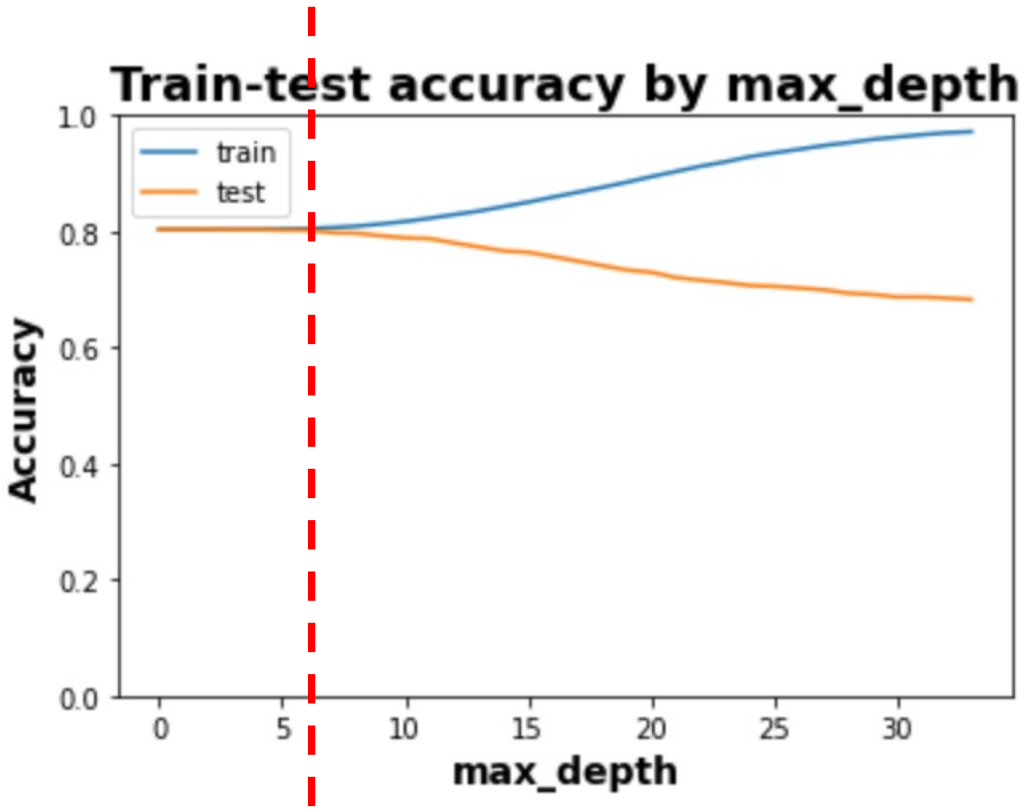


- A modell túl jól illeszkedik a tanuló mintára
- Max_depth paraméter növelése például a túltanulás veszélyével jár



Hiperparaméter optimalizálás

- Találjuk meg azt a max_depth mértéket, aminél a legideálisabb a train – test pontosság aránya



Hiperparaméter optimalizálás

- Manuálisan
- Grid search
- Random search

Hiperparaméter optimalizálás

Manuálisan

```
For i in range(1, 30):
    clf = DecisionTreeClassifier(max_depth=i)
    clf.fit(X_train, y_train)
    pred = clf.predict(X_test)
    scoring...
```

```
criterion_list = [”gini”, „entropy”]
For crit in criterion_list:
    For i in range(1, 30):
        clf = DecisionTreeClassifier(criterion=crit, max_depth=i)
        clf.fit(X_train, y_train)
        pred = clf.predict(X_test)
        scoring...
```

Hiperparaméter optimalizálás

Grid search

- A megadott hiperparaméter értékek közül minden lehetőséget kipróbál
- Futási idő++

Hiperparaméter optimalizálás

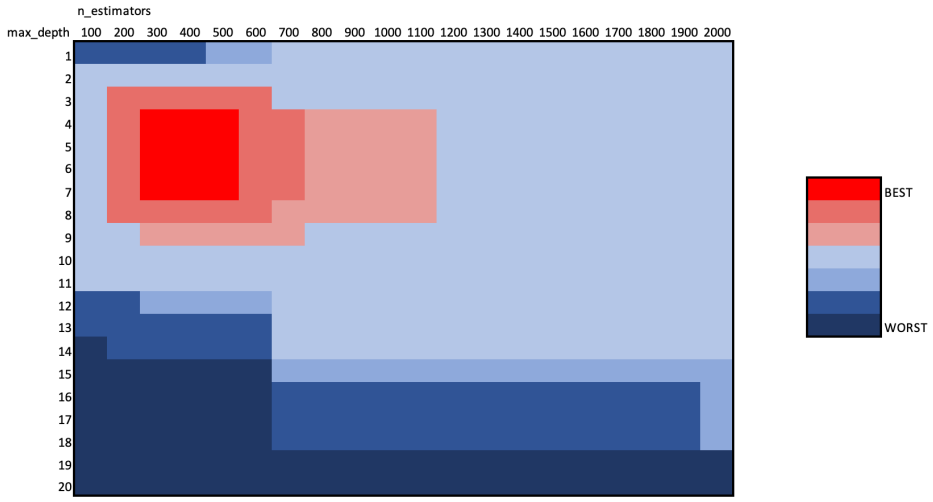
Grid search

- 1. grid search importálása
- 2. hiperparaméter-tér megadása
- 3. grid search definialasa
- 4. grid search fitelése

```
from sklearn.model_selection import GridSearchCV

param_grid = [{
„criterion_list”: [”gini”, „entropy”],
„max_depth”: [1, 5, 10, 20]
}]
cv = GridSearchCV(model, param_grid, cv=5)
cv.fit(X, y)
cv.best_params_
cv.best_score_
```

Hiperparaméter optimalizálás



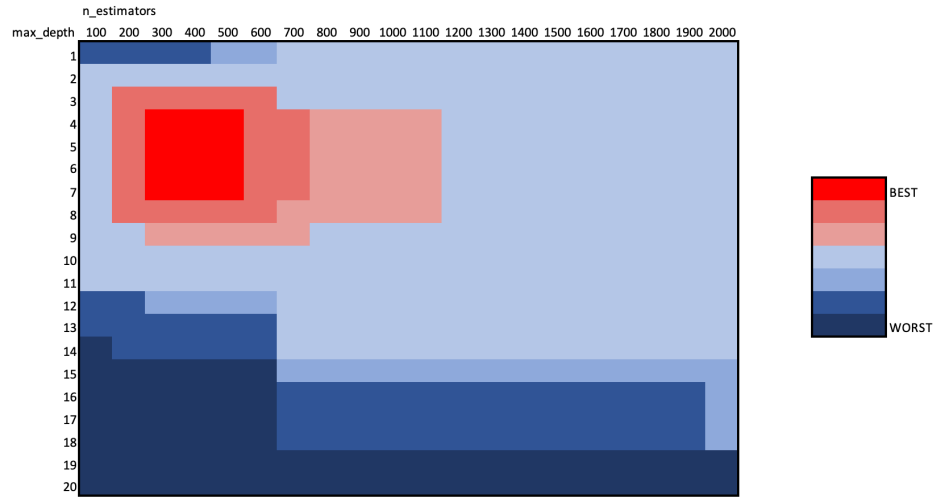
Random search

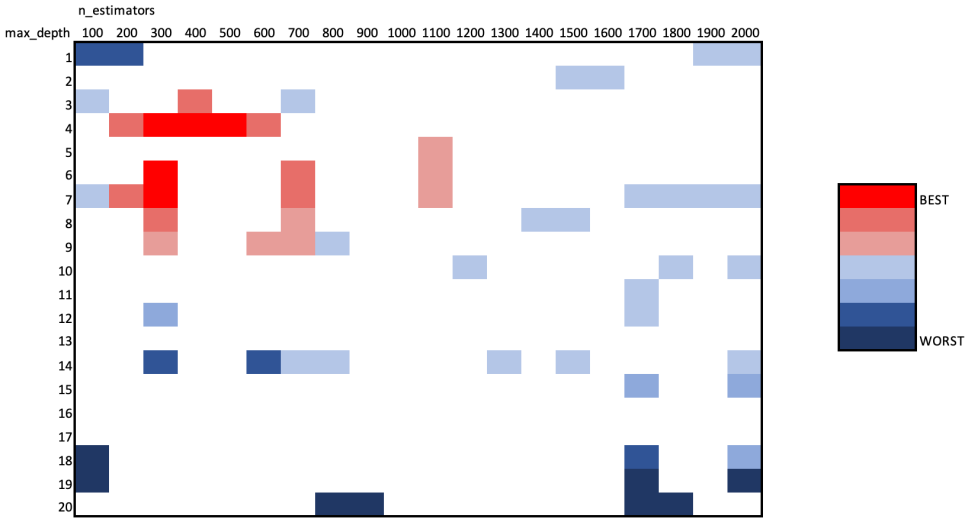
- A megadott hiperparaméter értékek közül véletlenszerűen néhányat kipróbál
- Kevesebb futási idő
- Nem biztos, hogy megtalálja a legjobb megoldást
- Elég jó megoldást talál

Hiperparaméter optimalizálás

Random search

- A megadott hiperparaméter értékek közül véletlenszerűen néhányat kipróbál
- Kevesebb futási idő
- Nem biztos, hogy megtalálja a legjobb megoldást
- Elég jó megoldást talál





Hiperparaméter optimalizálás

Random search

- 1. random search importálása
- 2. hiperparaméter-tér megadása
- 3. random search definialasa
  - iterációk számának megadása
- 4. random search fitelése

```
from sklearn.model_selection import RandomSearchCV

param_dist = [{
„criterion_list”: [”gini”, „entropy”],
„max_depth”: [1, 5, 10, 20]
}]
cv = RandomSearchCV(model, param_dist, n_iter=5, cv=5)
cv.fit(X, y)
cv.best_params_
cv.best_score_
```

Train – Test

Train – test split

- Külön halmazon tanítunk
- Külön halmazon mérünk eredményt
- Azért tesszük, hogy lássuk, a modellünk mennyire jól általánosít, azaz, hogy teljesít eddig még nem látott adaton
- Teszt halmaz és validációs halmaz fogalmának meghatározása zavaros

Keresztvalidáció

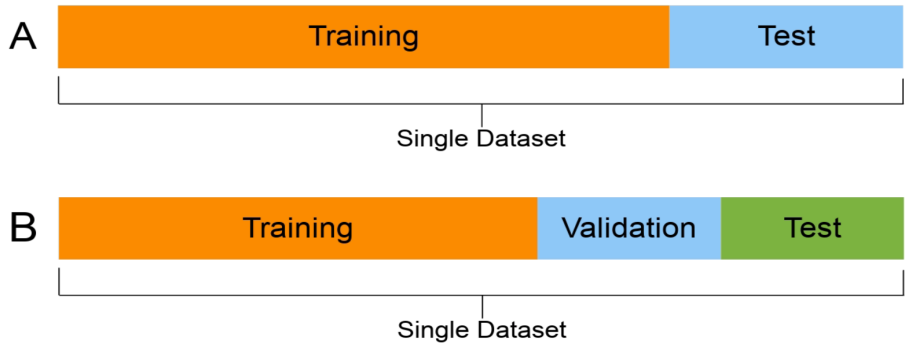
CV – cross validation

- Az egész adathalmazra tudunk predikciót készíteni
- Hiperparaméter-optimalizálás során ezeken a teszthalmazokon optimalizálunk
- Ezek a halmazok így szintén megtévesztő eredményeket adhatnak

Train – Test - Validation

Train – validation - test

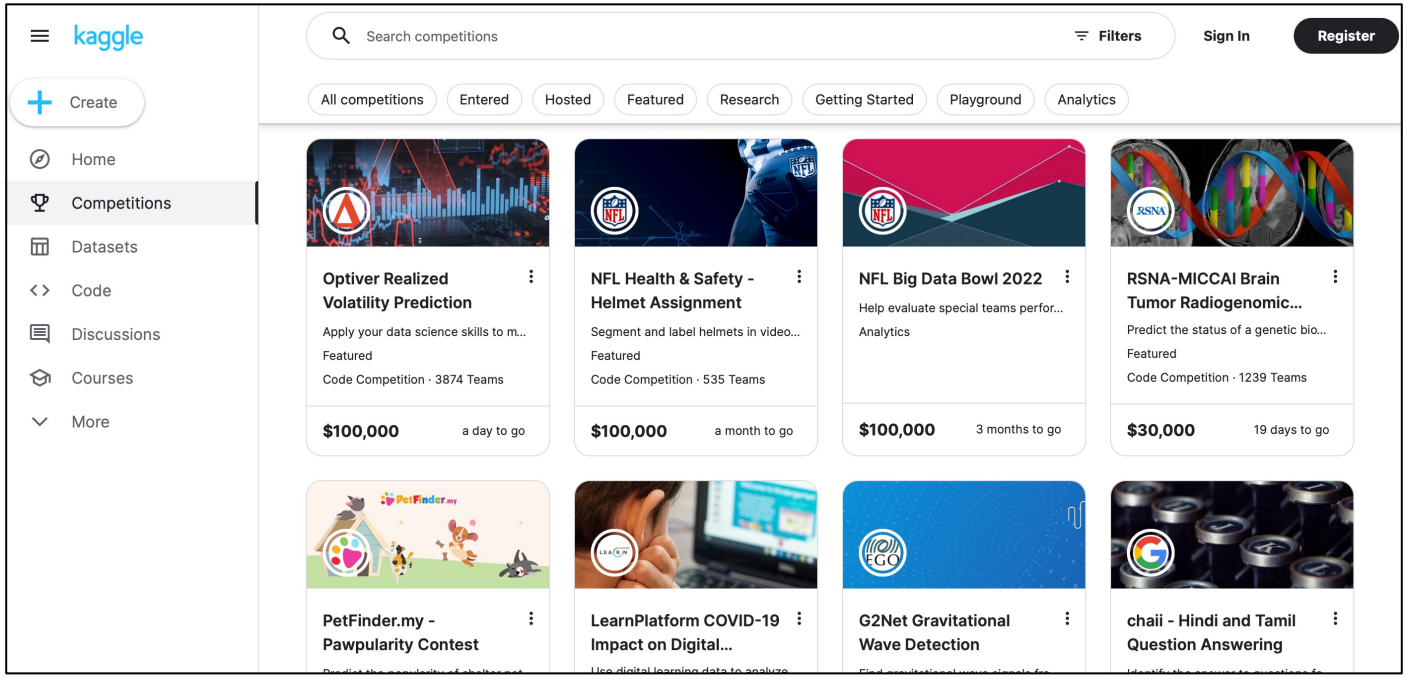
- Az adathalmaz egy részét már a feladat legelején leválasztjuk, és semmilyen lépés során nem használjuk fel a modellezés során
- A modellezés végeztével ezen a halmazunk állapítjuk meg modellünk jóságát



Train – Test - Validation

Kaggle

- Adatelemző versenyek platformja
- Kiértékelés során public és private leaderboard



Train – Test - Validation

Kaggle

- Ha túltanítjuk a modellünket a public leaderboard alapján, attól még az összesítés során könnyen lemaradhatunk