DOLab

live class

Hyperparameter Tuning

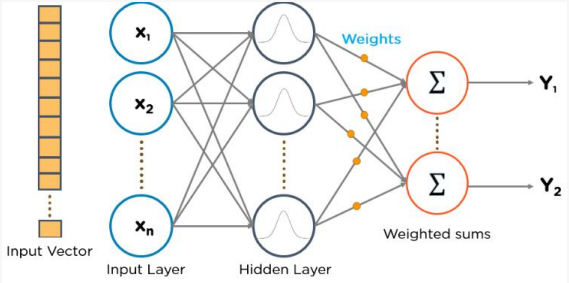
Outline

- Overview of hyperparameter tuning
- Grid search
- Randomized

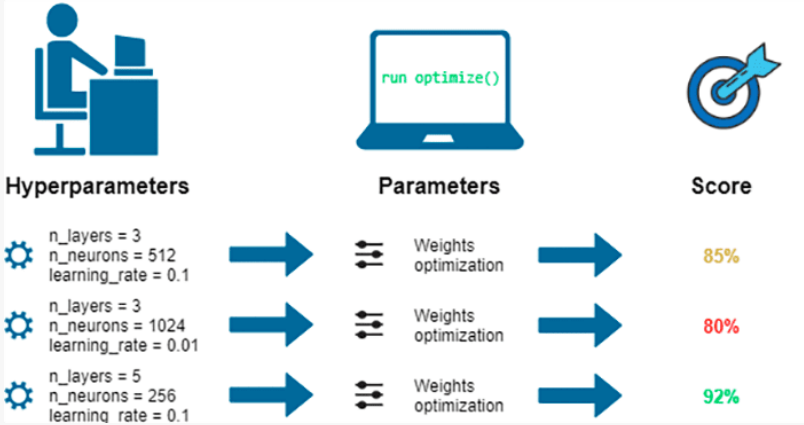
Overview of hyperparameter tuning

Kenapa perlu ketahui ini?

1. Saat ini algoritma semakin kompleks, sehingga hyperparameter dipilih banyak



2. Tuning memerlukan waktu yang banyak



Apa itu **Parameter?**

1. Nilai yang dihasilkan oleh model machine learning selama proses pembelajaran *(output)*
2. Tidak bisa disetting secara manual
3. Ini akan diketahui dari algoritma yang kita pilih

Contoh: Linear Modelling

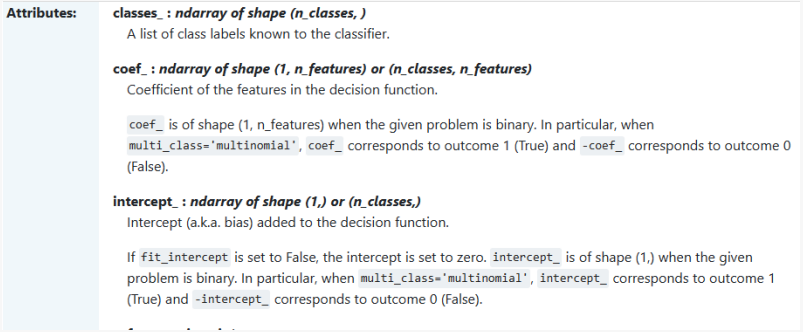
```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
print(log_reg.coef_)
```

```
array([[-0.0025, -0.0022, 0.0044,
0.00031, cont]])
```

Koefisien adalah Parameter, yang tidak kita tetapkan tapi dihasilkan selama pembelajaran

Cara cari **Parameter?**

1. Pahami algoritma yang kita gunakan
2. Jika perlu baca dokumentasi algoritma yang kita pilih (*attributes section bukan parameter section*)



Attributes:

classes_ : *ndarray of shape (n_classes,)*
A list of class labels known to the classifier.

coef_ : *ndarray of shape (1, n_features) or (n_classes, n_features)*
Coefficient of the features in the decision function.

`coef_` is of shape (1, n_features) when the given problem is binary. In particular, when `multi_class='multinomial'`, `coef_` corresponds to outcome 1 (True) and `-coef_` corresponds to outcome 0 (False).

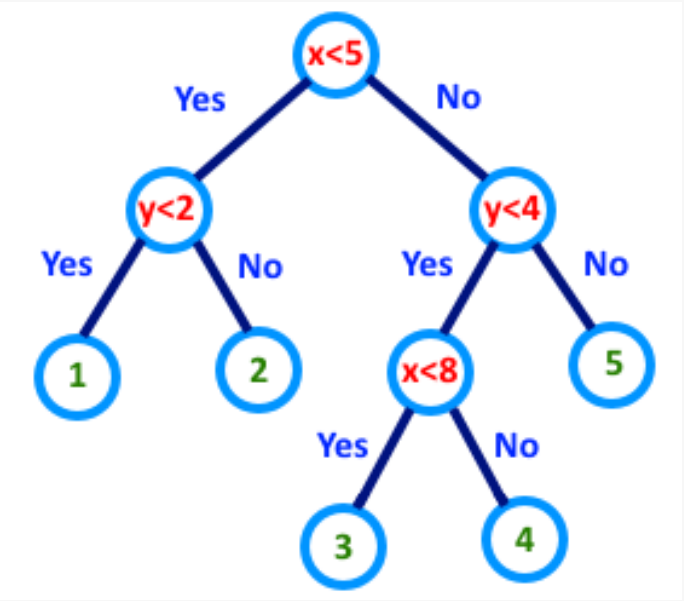
intercept_ : *ndarray of shape (1,) or (n_classes,)*
Intercept (a.k.a. bias) added to the decision function.

If `fit_intercept` is set to False, the intercept is set to zero. `intercept_` is of shape (1,) when the given problem is binary. In particular, when `multi_class='multinomial'`, `intercept_` corresponds to outcome 1 (True) and `-intercept_` corresponds to outcome 0 (False).

Sources : Scikit-Learn (Logistic Regression)

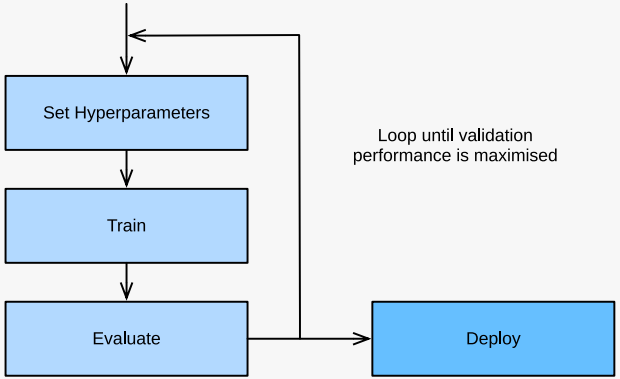
Jika non-linear, **Bagaimana?**

Random Forest tidak memiliki koefisien tapi berdasarkan cabang tangkai (node) untuk memisahkan feature dan value yang akan di split.



Apa itu **Hyperparameter?**

Sesuatu yang di **setting** sebelum pemodelan dan **tidak dipelajari algoritma**



Cara cari **Hyperparameter?**

1. Pahami algoritma yang kita gunakan
2. Jika perlu baca dokumentasi algoritma yang kita pilih (*parameter section*)

Parameters:

penalty : *{'l1', 'l2', 'elasticnet', None}, default='l2'*

Specify the norm of the penalty:

- `None`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

Sources : Scikit-Learn (Logistic Regression)

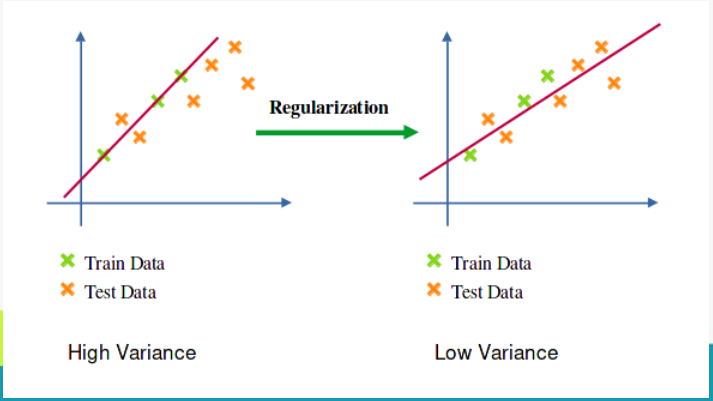
Bedah satu Hyperparameter.

Tipe data dan defaultnya

{'l1', 'l2', 'elasticnet', None}, default='l2'

Definisi

teknik yang digunakan untuk mengendalikan kompleksitas model dan mengurangi risiko overfitting



Setting

Contoh: Linear Modelling

```
log_reg = LogisticRegression(penalty = 'lasso')
log_reg.get_params()
```

```
{'C': 1.0, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'lasso', 'random_state': None, 'solver': 'lbfgs', 'tol': 0.0001, 'verbose': 0, 'warm_start': False}
```

Catatan tentang Hyperparameter

Ada beberapa parameter yang tidak membantu pemodelan

Contoh: Random Forest

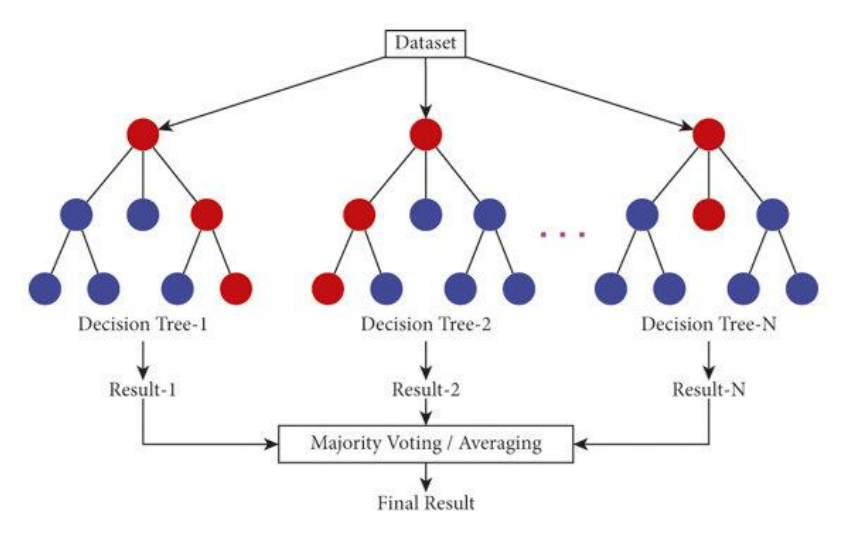
`n_job` : Mempercepat waktu pemodelan

`random_state` : Pemilihan acak data

`verbose` : Menampilkan informasi proses

Beberapa parameter tidak perlu digunakan dalam data training

Kuncinya adalah membaca paper atau literasi-literasi tertentu, untuk menentukan mana important dan tidak



Mana parameter yang perlu dahulu di *tune*?

1. Tergantung modelling yang kita gunakan
2. Perhatikan parameter yang conflict

Contoh: Linear Modelling

solver merupakan variabel yang conflict dengan penalty

Warning: The choice of the algorithm depends on the penalty chosen. Supported penalties by solver:

- 'lbfgs' - ['l2', None]
- 'liblinear' - ['l1', 'l2']
- 'newton-cg' - ['l2', None]
- 'newton-cholesky' - ['l2', None]
- 'sag' - ['l2', None]
- 'saga' - ['elasticnet', 'l1', 'l2', None]

Automating Hyperparameter

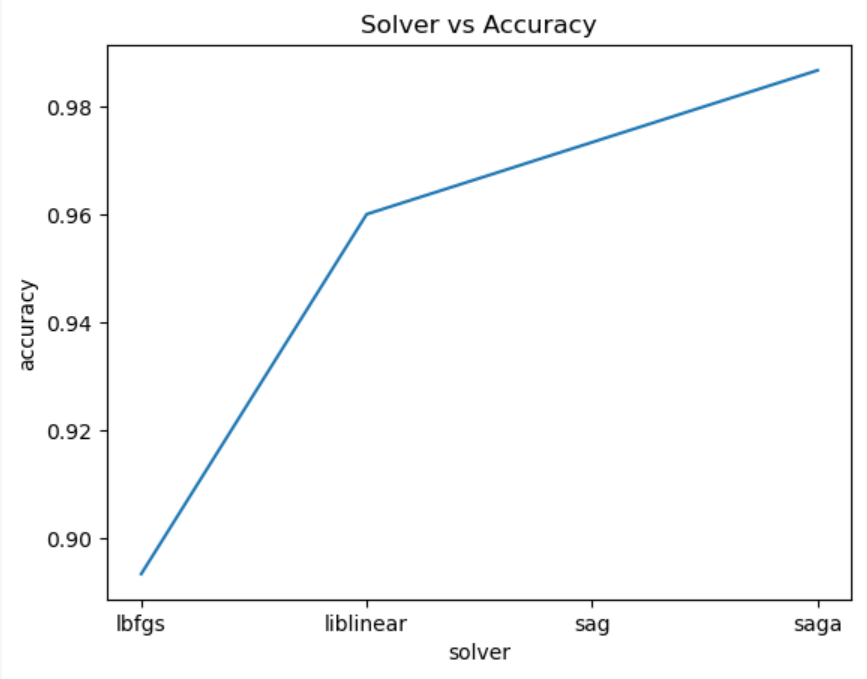
Contoh: Linear Modelling

```
solver = ['lbfgs', 'liblinear', 'liblinear', 'sag', 'saga']
accuracy = []
for i in solver:
  model = LogisticRegression(solver = i)
  logreg = model.fit(X_train, y_train)
  pred = logreg.predict(X_test)
  acc = accuracy_score(y_test, pred)
  accuracy.append(acc)
```

Learning Curves

Contoh: Linear Modelling

```
plt.plot(solver, accuracy)
plt.gca().set(xlabel = 'solver', ylabel = 'accuracy', title = 'Solver vs Accuracy')
plt.show()
```



Grid search

Masih ingat ini?

Automating Hyperparameter

Contoh: Linear Modelling

```
solver = ['lbfgs', 'liblinear', 'liblinear', 'sag', 'saga']
accuracy = []
for i in solver:
  model = LogisticRegression(solver = i)
  logreg = model.fit(X_train, y_train)
  pred = logreg.predict(X_test)
  acc = accuracy_score(y_test, pred)
  accuracy.append(acc)
```

Akan menjadi pertanyaan apabila parameternya lebih dari satu yang digunakan. Akan dibuat nested loop kan?

solver dengan max_iter

Catatan: Parameter tidak hanya linear relationship tapi exponential antar parameter

```
def func_iter(solver, max_iter):
  model = LogisticRegression(solver = solver,
            max_iter = max_iter)
  pred = model.fit(X_train, y_train).predict(
          X_test)
  return([solver, max_iter, accuracy_score(
          y_test, pred)])

result = []
for solv in solver:
  for iter in max_iter:
    result.append(func_iter(solver, max_iter))
```

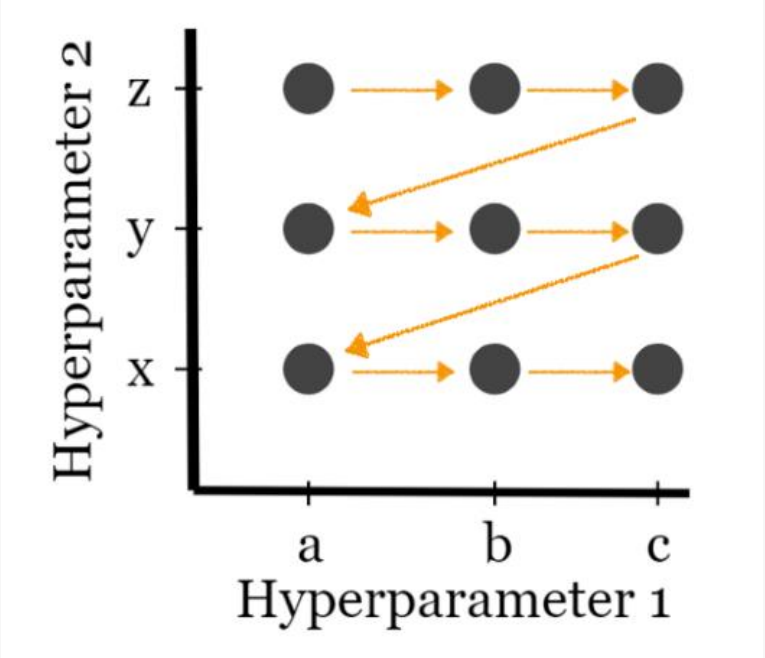
Masih ingat ini?

Jika terdapat:
1. 5 value parameter untuk parameter 1
2. 10 value parameter untuk parameter 2

Secara total terdapat 50 iterasi value parameter

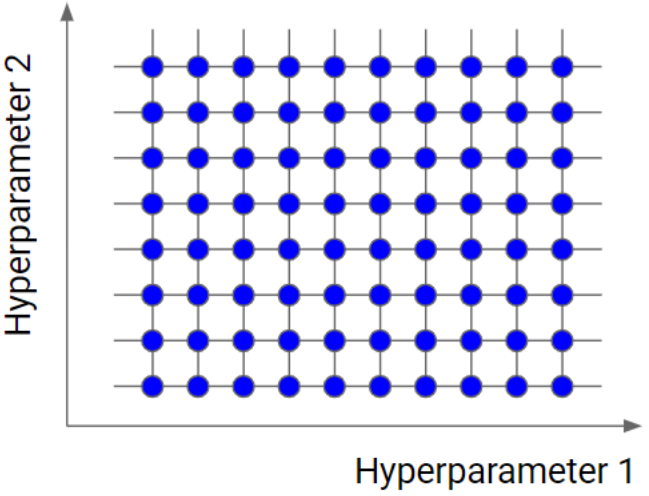
Bagaimana jika cross validation untuk 10 kali?
Bisa 500 iterasi value parameter

LALU, bagaimana jika lebih dari 2 parameter?
Pada akhirnya kita tidak bisa nested loop secara berkala, tidak efektif



Apa itu **Grid Search?**

Mencoba value parameter dalam algoritma dari hyperparameter yang kita gunakan dan menemukan mana yang terbaik



Tutorial dalam Scikit-Learn

`GridSearchCV` dan selanjutnya melakukan

1. Penentuan algoritma yang dipilih
2. Penentuan hyperparameter akan dituning
3. Range value dalam hypeparameter
4. Pilih mana hasil yang terbaik

Input umum digunakan dari GridSearchCV

Diantaranya terbagi:

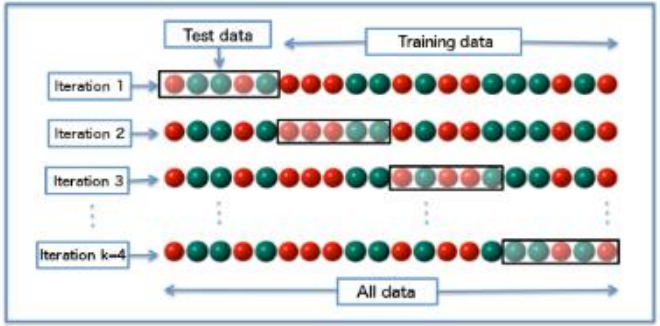
estimator : algoritma yang digunakan (contoh, logistic regression). PILIH **satu.**

param_grid : hyperparameter dengan value yang digunakan (contoh, max_iter). Bentuk **dict.**

cv : bagaimana kita membuat cross validation (masukan **k**, misal 5 fold, 10 fold dan disesuaikan)

scoring : apa yang akan di evaluasi dalam model

Catatan mengecek evaluasi metrik:
from sklearn import metrics
metrics.SCORERS.keys()



Memahami Output dari

Dimana terdapat 3 hasil:

1. Result log

   `cv_results_`

2. Best result

   `best_index`

   `best_params` ★

   `best_score` ★

   `best_estimators` ★

   `_`

Diakses dengan

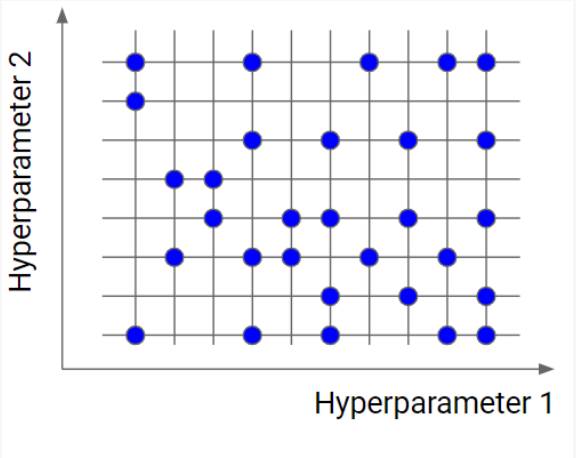
`Grid_search_object.propery`

1. best_params_ : untuk mendapatkan parameter terbaik
2. best_scores_ : untuk mendapatkan score terbaik
3. best_estimators_ : untuk memilih parameter terpilih

Random search

Apa itu **Random Search?**

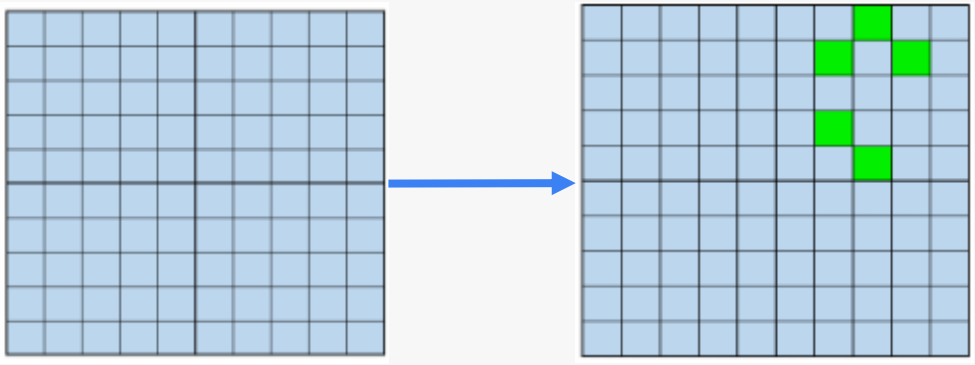
Ini masih sama dengan GridSearch, kita masih mendefine parameter, maupun estimator. Hanya yang membedakan adalah pengambilan grid secara random



Lalu **ini berfungsi?**

Berdasarkan Bengio & Bergstra (2012), terdapat dua alasan

1. Tidak semua hyperparameter itu penting
2. Probability Trick



Best params dengan warna hijau.

Hanya 5 dari 100 atau 0.05 peluang berhasil atau (1 – 0.05) peluang tidak berhasil

Dan untuk setiap perulangan, memiliki peluang untuk peluang tidak berhasil
(1 – 0.05) ^ n
n : jumlah percobaan

Lalu berapakah perulangan percobaan atau (n) agar mendapatkan peluang 95% diberhasil?

1. Jika peluang tidak berhasil adalah (1 – 0.05) ^ n
2. Maka perlu (1 – peluang tidak berhasil) atau sama dengan (1 – (1 – 0.05) ^ n)
3. Dari hasil tersebut didapati n >= 59 dari (1 – (1 – 0.05) ^ n) >= 0.95

Dengan uji coba sedikit, maka probalitas akan tinggi untuk mendapatkan berhasil tanpa perlu semua spot dianalisa

Tutorial dalam Scikit-Learn

`RandomizedSearchCV` dan selanjutnya melakukan

1. Penentuan algoritma yang dipilih
2. Penentuan hyperparameter akan dituning
3. Range value dalam hypeparameter
4. **Penetuan sampel yang akan diambil**
5. Pilih mana hasil yang terbaik

Bedanya GridSearchCV dengan RandomizedSearchCV

```
sklearn.model_selection.GridSearchCV(estimator, param_grid,
        scoring=None, fit_params=None,
        n_jobs=None,
        refit=True, cv='warn', verbose=0,
        pre_dispatch='2*n_jobs',
        error_score='raise-deprecating',
        return_train_score='warn')
```

```
sklearn.model_selection.RandomizedSearchCV(estimator,
        param_distributions, n_iter=10,
        scoring=None, fit_params=None,
        n_jobs=None, refit=True,
        cv='warn', verbose=0,
        pre_dispatch='2*n_jobs',
        random_state=None,
        error_score='raise-deprecating',
        return_train_score='warn')
```

Terdapat dua perbedaan:

`n_iter` Jumlah sampel untuk pengambilan parameter

`param_distributions` ini sedikit beda dengan param_grid, berguna sebagai distribution sampling

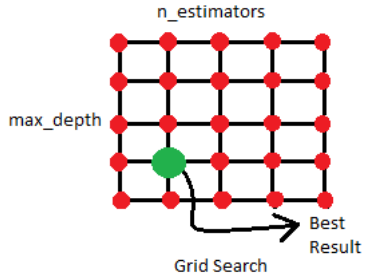
(defaultnya equal, semua kombinasi memiliki peluang yang sama)

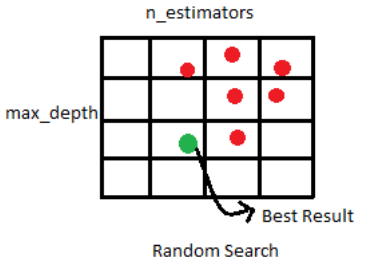
Conclusion

Kesamaan GridSearchCV dengan RandomizedSearchCV

1. Keduanya digunakan untuk automating hyperparamater
2. Penentuan grid dimana berisikan parameter termasuk dengan value yang dipilih
3. Penentuan mekanisme cross-validation dan scoring

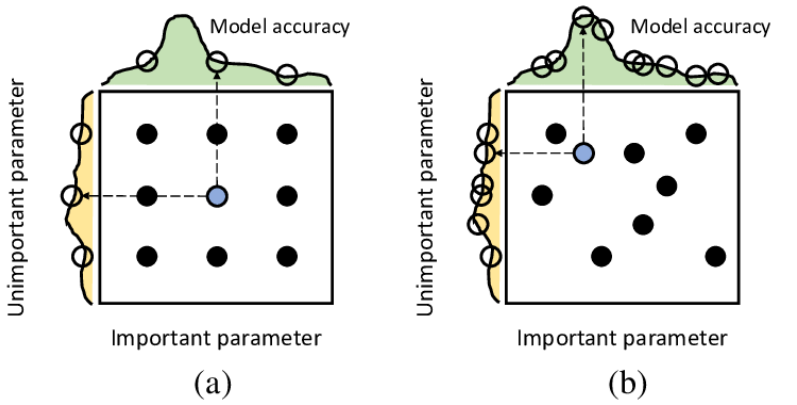
Ingat, model terbaik akan muncul dari GRID yang kita pilih





Perbedaan GridSearchCV dengan RandomizedSearchCV

1. GridSearch – Mencari mendalam untuk semua kombinasi di grid/ruang sampel dan tanpa sampling metodologi (Membutuhkan proses yang panjang, namun memastikan pasti dapat parameter terbaik)
2. RandomizedSearch – Mencari subpengumpulan secara acak dengan menentukan bagaimana bentuk pengambilannya (bentuk defaultnya adalah *uniform*) (Membutuhkan proses yang singkat namun belum pasti mendapatkan parameter terbaik)



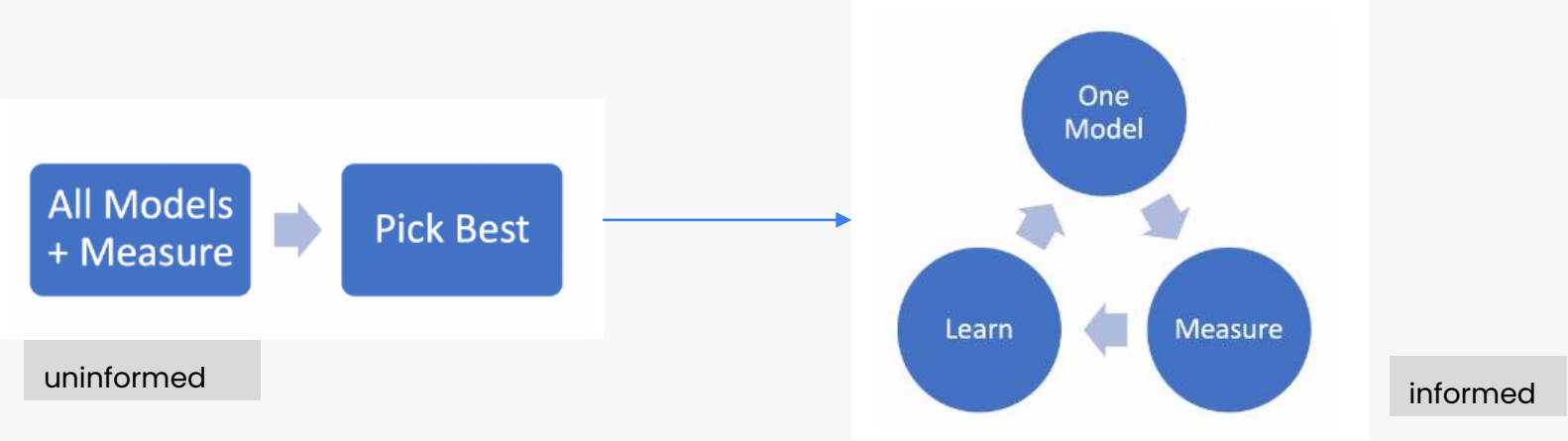
Lalu antara GridSearchCV dengan RandomizedSearchCV mana yang dipilih?

1. Semakin banyak **data** yang dimiliki, lebih baik menggunakan **RandomizedSearch**
2. Semakin banyak **hyperparameter** yang dimiliki, lebih baik menggunakan **RandomizedSearch**
3. Jika kamu tidak mempunyai waktu banyak untuk menunggu, lebih baik menggunakan **RandomizedSearch**

Ingat, randomizedSearch memperbesar peluang mendapatkan hasil yang baik, walaupun bukan terbaik yah!

Sebelumnya

Saat kita menggunakan GridSearchCV dengan RandomizedSearchCV adalah pencarian tanpa informasi (artinya setiap iterasi hyperparameter tidak belajar dari iterasi sebelumnya)



uninformed

informed

Informed Search

Adapun tutorialnya sebagaimana berikut:

1. Lakukan random search
2. Melihat area yang bagus untuk diobservasi lebih lanjut
3. Melakukan Grid Search ke area tersebut
4. Lanjutkan sampai mendapatkan hasil yang optimal

Keuntungan Informed Search

1. Melakukan optimalisasi dari `GridSearchCV` dan `RandomizedSearchCV`
   Dimana mencari secara luas dengan **Random Search**, kemudian secara detail dengan **Grid Search**
2. Proses penggunaan akan lebih efisien dan saling menginformasi setiap tahapan yang dilakukan

Jika

Kita memiliki hyperparameter sebagai berikut:

`max_depth_list` dari 1 sampai dengan 65

`min_sample_list` dari 3 sampai dengan 17

`learn_rate_list` dari 0.01 sampai dengan 150 dengan 150 value

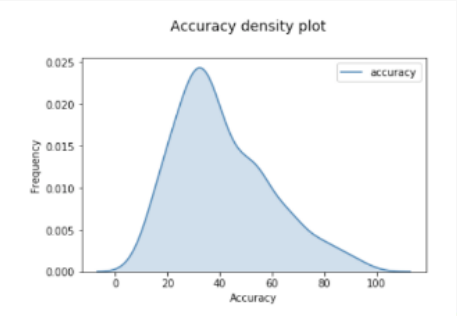
Maka secara total ada 134400 iterasi loh.

`RandomizedSearchCV`

Pilih 500 kombinasi

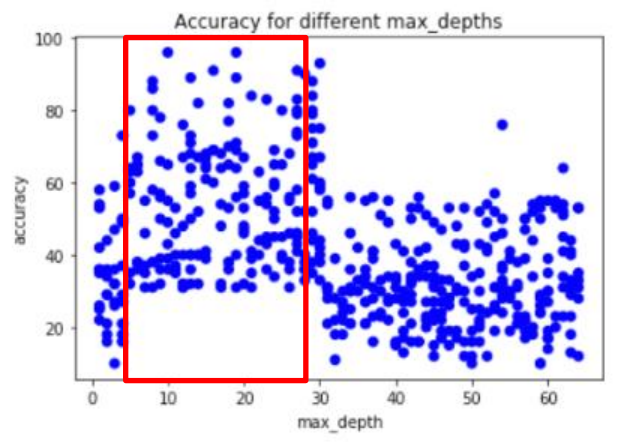
max_depth	min_samples_leaf	learn_rate	accuracy
10	7	0.01	96
19	7	0.023355705	96
30	6	1.038389262	93
27	7	1.11852349	91
16	7	0.597651007	91

Top Result



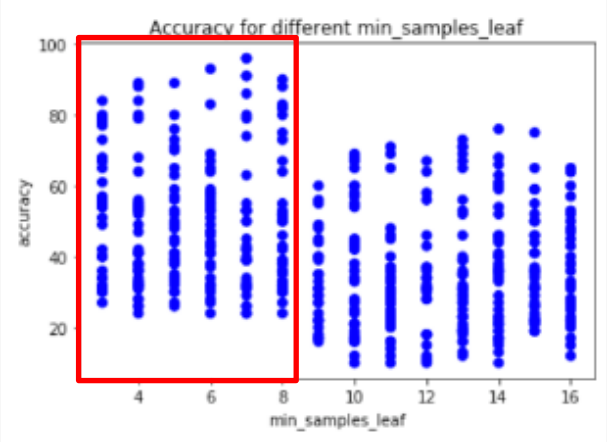
Temukan Kesimpulan

max_depth_list



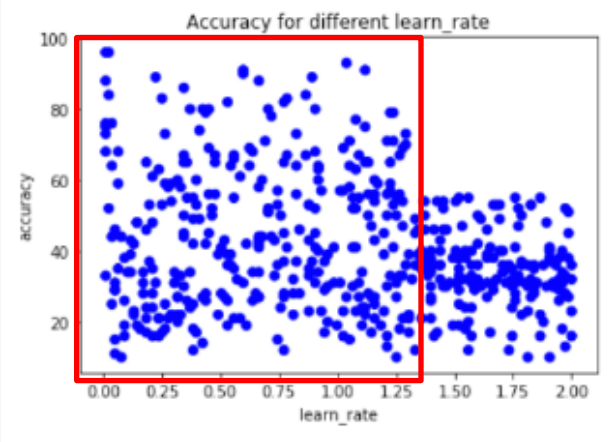
dari 8 sampai dengan 30

min_sample_list



Kurang dari 8

learn_rate_list



Kurang dari 1.3


Thanks!