



Pengenalan Metode Ensemble

Oleh: Muhammad Mirza Fahmi

Intuition

"Persatuan adalah Kekuatan". sebuah ungkapan lama yang menjadi inspirasi yang mengatur **metode ensemble** dalam mesin pembelajar.

Metode ini berbasis pada hipotesis yang mengkombinasikan beberapa model secara bersamaan yang seringkali menghasilkan model yang jauh lebih kuat.



Definisi

Ensemble Learning merupakan paradigma mesin pembelajar yang beberapa model (sering disebut “pembelajar lemah”) dilatih untuk menyelesaikan **masalah yang sama** dan **dikombinasikan** untuk mendapat hasil yang lebih baik.



Pembelajar Lemah

Pembelajar Lemah secara istilah berarti model yang performanya sedikit lebih baik daripada model naif/sederhana.

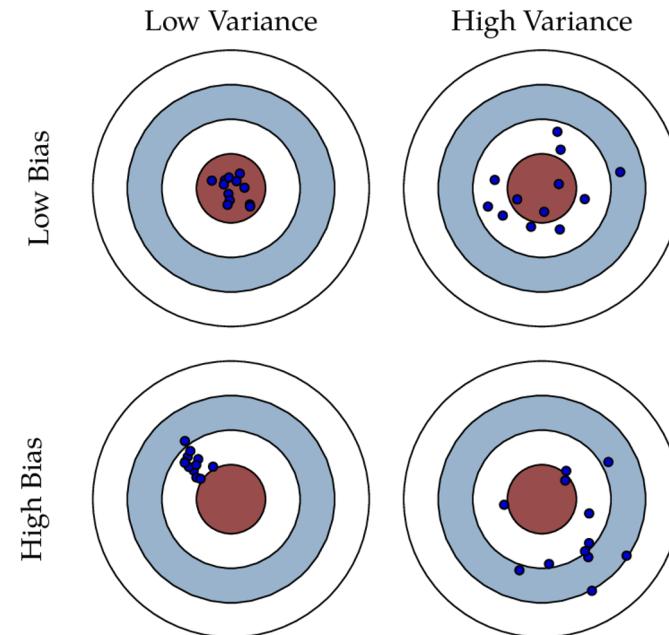
Contoh pembelajar lemah adalah **decision tree**. Karena kelemahan dari *tree* dapat dikendalikan oleh kedalaman *tree* selama konstruksi.



Mengapa Menggunakan Metode Ensemble

Seringkali, model-model dasar ini memiliki performa yang tidak terlalu baik karena mempunyai *bias* atau *variance tinggi*.

Karenanya, ide berkenaan dengan metode ensemble berusaha untuk **mengurangi** *bias* dan atau *variance* dengan cara mengombinasikannya yang bertujuan untuk membuat *strong learner*.





**Teknik-teknik
Ensemble
Sederhana**

Hard Voting

Teknik ini menggunakan **beberapa model** yang telah dilatih untuk membuat prediksi. Setiap prediksi dianggap sebagai sebuah vote (pilihan).

Dari keseluruhan prediksi, kita ambil pilihan terbanyak atau mayoritas sebagai prediksi akhir.

Averaging

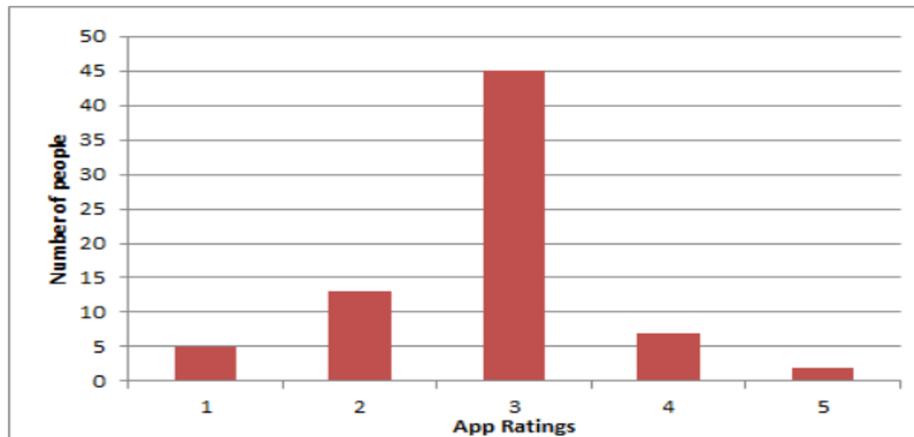
Serupa dengan yang sebelumnya, hanya saja yang dihitung adalah **rata-rata** jumlah votenya yang diambil sebagai prediksi akhir.

$$A = \frac{1}{n} \sum_{i=1}^n a_i = \frac{a_1 + a_2 + \dots + a_n}{n}$$

Soft Voting (Weighted Averaging)

Kelanjutan dari teknik pengambilan rata-rata prediksi, yaitu setiap model diberikan **bobot** tertentu bergantung pada **signifikansinya** terhadap hasil prediksi.

$$W = \frac{\sum_{i=1}^n w_i X_i}{\sum_{i=1}^n w_i}$$



Hands-on Lab

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.model_selection import RepeatedStratifiedKFold, cross_val_score

from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import VotingClassifier
```

```
# get a voting ensemble of models
def get_voting(voting_method):
    # define the base models
    models = list()
    models.append(('knn1', KNeighborsClassifier(n_neighbors=1)))
    models.append(('knn3', KNeighborsClassifier(n_neighbors=3)))
    models.append(('knn5', KNeighborsClassifier(n_neighbors=5)))
    models.append(('knn7', KNeighborsClassifier(n_neighbors=7)))
    models.append(('knn9', KNeighborsClassifier(n_neighbors=9)))
    # define the voting ensemble
    ensemble = VotingClassifier(estimators=models, voting=voting_method)
    return ensemble
```

Hands-on Lab

```
# get a list of models to evaluate
def get_models(voting_method):
    models = dict()
    models['knn1'] = KNeighborsClassifier(n_neighbors=1)
    models['knn3'] = KNeighborsClassifier(n_neighbors=3)
    models['knn5'] = KNeighborsClassifier(n_neighbors=5)
    models['knn7'] = KNeighborsClassifier(n_neighbors=7)
    models['knn9'] = KNeighborsClassifier(n_neighbors=9)
    models[f'{voting_method}_voting'] = get_voting(voting_method)
    return models
```

```
# Create dataset
X, y = make_classification(n_samples=10000, n_features=5)
```

```
# evaluate a give model using cross-validation
def evaluate_model(model, X, y):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')
    return scores
```

Hands-on Lab

```
# get the models to evaluate
models = get_models(voting_method='hard')
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, X, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, np.mean(scores), np.std(scores)))
```

```
>knn1 0.933 (0.007)
>knn3 0.946 (0.007)
>knn5 0.948 (0.006)
>knn7 0.947 (0.007)
>knn9 0.947 (0.007)
>hard_voting 0.948 (0.007)
```

```
# get the models to evaluate
models = get_models(voting_method='soft')
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, X, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, np.mean(scores), np.std(scores)))
```

```
>knn1 0.933 (0.007)
>knn3 0.946 (0.007)
>knn5 0.948 (0.006)
>knn7 0.947 (0.007)
>knn9 0.947 (0.007)
>soft_voting 0.946 (0.007)
```

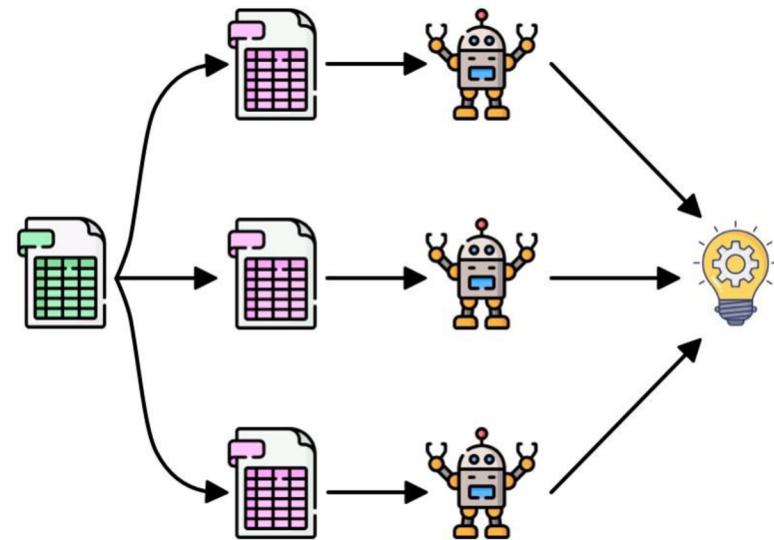


**Teknik-teknik
Ensemble
Lanjutan**

Metode Bagging

Metode ini sangat populer dalam kategori pembelajaran ensemble, biasanya digunakan dalam algoritma seperti **Random Forest**.

Bagging atau *Bootstrap Aggregating* mampu meningkatkan akurasi tidak hanya dengan merata-ratakan tapi juga membuat banyak model **yang tidak berkorelasi** dengan cara memberikan data latih yang berbeda-beda.

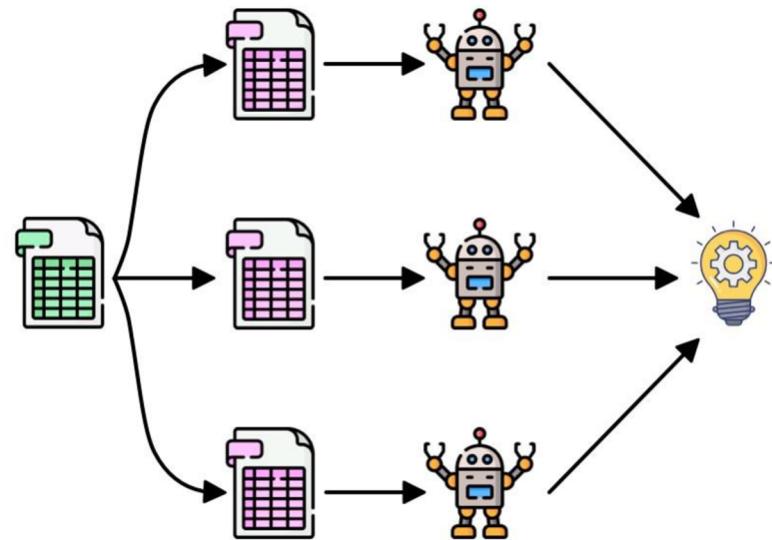


Metode Boosting

Metode ini proses pembelajarannya bertingkat yang berfokus untuk **meminimalkan** kesalahan.

Metodenya bersifat sekuensial, artinya setiap estimator bergantung pada estimator sebelumnya untuk meningkatkan hasil prediksi.

Metode boosting yang paling populer adalah **adaptive boosting** dan **gradient boosting**.



Quiz

Apa yang membedakan antara metode weighted average dan bagging dalam pembelajaran ensemble?

Kenapa sebuah model disebut sebagai *weak learner*?

Sebutkan Teknik-teknik untuk membuat prediksi akhir pada metode ensemble?