Metode Ensemble

Ansambel berarti sekelompok elemen yang dilihat secara keseluruhan daripada secara individual. Metode Ensemble membuat beberapa model dan menggabungkannya untuk menyelesaikannya. Metode ansambel membantu meningkatkan ketahanan/generalisasi model. Pada artikel ini, kita akan membahas beberapa metode dengan implementasinya dalam Python. Untuk ini, kami memilih **kumpulan data** dari repositori UCI.

Metode ansambel dasar

1. Metode rata-rata: Ini terutama digunakan untuk masalah regresi. Metode ini terdiri dari membangun beberapa model secara independen dan mengembalikan rata-rata prediksi semua model. Secara umum, output gabungan lebih baik daripada output individu karena varians berkurang.
Dalam contoh di bawah ini, tiga model regresi (regresi linier, xgboost, dan hutan acak) dilatih dan prediksinya dirata-ratakan. Keluaran prediksi akhir adalah pred_final.

```
# Mengimpor modul utilitas
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Mengimpor model pembelajaran mesin untuk prediksi
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.linear_model import LinearRegression

# Memuat kumpulan data kereta dalam Dataframe dari file train_data.csv
df = pd.read_csv("train_data.csv")

# mendapatkan data target dari dataframe
target = df["target"]

# Mendapatkan data kereta dari dataframe
train = df.drop("target")

# Memisahkan antara data pelatihan ke dalam himpunan data pelatihan dan validasi
X_train, X_test, y_train, y_test = train_test_split(
    train, target, test_size=0.20)
```

```
# Menginisialisasi semua objek model dengan parameter default
model_1 = LinearRegression()
model_2 = xgb.XGBRegressor()
model_3 = RandomForestRegressor()

# melatih semua model pada himpunan data pelatihan
model_1.fit(X_train, y_target)
model_2.fit(X_train, y_target)
model_3.fit(X_train, y_target)

# memprediksi output pada himpunan data validasi
pred_1 = model_1.predict(X_test)
pred_2 = model_2.predict(X_test)
pred_3 = model_3.predict(X_test)

# prediksi akhir setelah rata-rata prediksi dari semua 3 model
pred_final = (pred_1+pred_2+pred_3)/3.0

# mencetak kesalahan kuadrat rata-rata antara nilai riil dan nilai prediksi
print(mean_squared_error(y_test, pred_final))
```

2. Max voting: Ini terutama digunakan untuk masalah klasifikasi. Metode ini terdiri dari membangun beberapa model secara independen dan mendapatkan output individu yang disebut 'vote'. Kelas dengan suara maksimum dikembalikan sebagai output.
Dalam contoh di bawah ini, tiga model klasifikasi (regresi logistik, xgboost, dan hutan acak) digabungkan menggunakan sklearn VotingClassifier, model tersebut dilatih dan kelas dengan suara maksimum dikembalikan sebagai output. Keluaran prediksi akhir adalah pred_final. Harap dicatat bahwa ini adalah klasifikasi, bukan regresi, sehingga kerugiannya mungkin berbeda dari jenis metode ansambel lainnya.

```
# Mengimpor modul utilitas
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import log_loss

# Mengimpor model pembelajaran mesin untuk prediksi
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression

# mengimpor pengklasifikasi suara
from sklearn.ensemble import VotingClassifier

# Memuat kumpulan data kereta dalam Dataframe dari file train_data.csv
```

```
df = pd.read_csv("train_data.csv")

# mendapatkan data target dari dataframe
target = df["Weekday"]

# Mendapatkan data kereta dari dataframe
train = df.drop("Weekday")

# Memisahkan antara data pelatihan ke dalam himpunan data pelatihan dan
validasi
X_train, X_test, y_train, y_test = train_test_split(
    train, target, test_size=0.20)

# Menginisialisasi semua objek model dengan parameter default
model_1 = LogisticRegression()
model_2 = XGBClassifier()
model_3 = RandomForestClassifier()

# Membuat model akhir menggunakan pengklasifikasi pemungutan suara
final_model = VotingClassifier(
    estimators=[('lr', model_1), ('xgb', model_2), ('rf', model_3)],
voting='hard')

# melatih semua model pada himpunan data kereta
final_model.fit(X_train, y_train)

# memprediksi output pada himpunan data pengujian
pred_final = final_model.predict(X_test)

# kehilangan log pencetakan antara nilai aktual dan prediksi
print(log_loss(y_test, pred_final))
```

Mari kita lihat metode ansambel yang sedikit lebih canggih

Metode ansambel lanjutan

Metode ansambel banyak digunakan dalam pembelajaran mesin klasik. Contoh algoritma yang menggunakan bagging adalah meta-estimator hutan acak dan bagging dan contoh algoritma yang menggunakan boosting adalah GBM, XGBM, Adaboost, dll.

Sebagai pengembang model pembelajaran mesin, sangat disarankan untuk menggunakan metode ansambel. Metode ansambel digunakan secara luas di hampir semua kompetisi dan makalah penelitian.

1. Stacking: Ini adalah metode ansambel yang menggabungkan beberapa model (klasifikasi atau regresi) melalui meta-model (meta-klasifikasi atau meta-regresi). Model dasar dilatih pada himpunan data lengkap, kemudian meta-model dilatih pada fitur yang dikembalikan (sebagai output) dari model dasar. Model dasar dalam penumpukan biasanya berbeda. Meta-model membantu menemukan fitur dari model dasar untuk mencapai akurasi terbaik.

Algoritma:

1. *Pisahkan himpunan data kereta menjadi n bagian*
2. *Model dasar (katakanlah regresi linier) dipasang pada bagian n-1 dan prediksi dibuat untuk bagian ke-n. Ini dilakukan untuk setiap bagian n dari set kereta.*
3. *Model dasar kemudian dipasang pada seluruh himpunan data kereta.*
4. *Model ini digunakan untuk memprediksi himpunan data pengujian.*
5. *Langkah 2 hingga 4 diulang untuk model dasar lain yang menghasilkan serangkaian prediksi lain untuk set data kereta dan pengujian.*
6. *Prediksi pada kumpulan data kereta digunakan sebagai fitur untuk membangun model baru.*
7. *Model akhir ini digunakan untuk membuat prediksi pada himpunan data pengujian*

Penumpukan sedikit berbeda dari metode perakitan dasar karena memiliki model tingkat pertama dan kedua. Fitur penumpukan pertama kali diekstraksi dengan melatih himpunan data dengan semua model tingkat pertama. Model tingkat pertama kemudian menggunakan fitur

penumpukan kereta untuk melatih model daripada model ini memprediksi output akhir dengan fitur penumpukan pengujian.

```
# Mengimpor modul utilitas
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Mengimpor model pembelajaran mesin untuk prediksi
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.linear_model import LinearRegression

# mengimpor lib susun
from vecstack import stacking

# Memuat kumpulan data kereta dalam Dataframe dari file train_data.csv
df = pd.read_csv("train_data.csv")

# mendapatkan data target dari dataframe
target = df["target"]

# Mendapatkan data kereta dari dataframe
train = df.drop("target")

# Memisahkan antara data pelatihan ke dalam himpunan data pelatihan dan
validasi
X_train, X_test, y_train, y_test = train_test_split(
    train, target, test_size=0.20)


# menginisialisasi semua objek model dasar dengan parameter default
model_1 = LinearRegression()
model_2 = xgb.XGBRegressor()
model_3 = RandomForestRegressor()

# menempatkan semua objek model dasar dalam satu daftar
all_models = [model_1, model_2, model_3]

# menghitung fitur tumpukan
s_train, s_test = stacking(all_models, X_train, X_test,
                           y_train, regression=True, n_folds=4)

# menginisialisasi model tingkat kedua
final_model = model_1

# Menyesuaikan model tingkat kedua dengan fitur tumpukan
final_model = final_model.fit(s_train, y_train)

# memprediksi output akhir menggunakan penumpukan
```

```
pred_final = final_model.predict(X_test)

# mencetak kesalahan kuadrat rata-rata antara nilai riil dan nilai prediksi
print(mean_squared_error(y_test, pred_final))
```

2. Blending: Ini mirip dengan metode penumpukan yang dijelaskan di atas, tetapi alih-alih menggunakan seluruh kumpulan data untuk melatih model dasar, kumpulan data validasi dipisahkan untuk membuat prediksi.
Algoritma:

1. *Pisahkan himpunan data pelatihan menjadi himpunan data latihan, pengujian, dan validasi.*
2. *Sesuaikan semua model dasar menggunakan himpunan data latihan.*
3. *Buat prediksi tentang validasi dan uji himpunan data.*
4. *Prediksi ini digunakan sebagai fitur untuk membangun model tingkat kedua*
5. *Model ini digunakan untuk membuat prediksi pada pengujian dan meta-fitur*

```
# Mengimpor modul utilitas
import pandas as pd
from sklearn.metrics import mean_squared_error

# Mengimpor model pembelajaran mesin untuk prediksi
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.linear_model import LinearRegression

# mengimpor split uji kereta
from sklearn.model_selection import train_test_split

# Memuat kumpulan data kereta dalam Dataframe dari file train_data.csv
df = pd.read_csv("train_data.csv")

# mendapatkan data target dari dataframe
target = df["target"]

# Mendapatkan data kereta dari dataframe
train = df.drop("target")

#Splitting antara melatih data ke dalam himpunan data pelatihan dan validasi
X_train, X_test, y_train, y_test = train_test_split(train, target,
test_size=0.20)
```

```
# melakukan pengujian kereta dan pemisahan validasi
train_ratio = 0.70
validation_ratio = 0.20
test_ratio = 0.10

# melakukan pemisahan uji kereta
x_train, x_test, y_train, y_test = train_test_split(
    train, target, test_size=1 - train_ratio)

# melakukan pemisahan validasi pengujian
x_val, x_test, y_val, y_test = train_test_split(
    x_test, y_test, test_size=test_ratio/(test_ratio + validation_ratio))

# menginisialisasi semua objek model dasar dengan parameter default
model_1 = LinearRegression()
model_2 = xgb.XGBRegressor()
model_3 = RandomForestRegressor()

# melatih semua model pada himpunan data kereta

# Pelatihan Model Pertama
model_1.fit(x_train, y_train)
val_pred_1 = model_1.predict(x_val)
test_pred_1 = model_1.predict(x_test)

# Mengonversi ke Dataframe
val_pred_1 = pd.DataFrame(val_pred_1)
test_pred_1 = pd.DataFrame(test_pred_1)

# pelatihan model kedua
model_2.fit(x_train, y_train)
val_pred_2 = model_2.predict(x_val)
test_pred_2 = model_2.predict(x_test)

# Mengonversi ke Dataframe
val_pred_2 = pd.DataFrame(val_pred_2)
test_pred_2 = pd.DataFrame(test_pred_2)

# training third model
model_3.fit(x_train, y_train)
val_pred_3 = model_1.predict(x_val)
test_pred_3 = model_1.predict(x_test)

# Mengonversi ke Dataframe
val_pred_3 = pd.DataFrame(val_pred_3)
test_pred_3 = pd.DataFrame(test_pred_3)

# kumpulan data validasi gabungan bersama dengan semua data validasi yang
diprediksi (fitur meta)
df_val = pd.concat([x_val, val_pred_1, val_pred_2, val_pred_3], axis=1)
df_test = pd.concat([x_test, test_pred_1, test_pred_2, test_pred_3], axis=1)
```

```
# Membuat model akhir menggunakan fitur meta
final_model = LinearRegression()
final_model.fit(df_val, y_val)

# mendapatkan hasil akhir
final_pred = final_model.predict(df_test)

#printing kesalahan kuadrat rata-rata antara nilai riil dan nilai prediksi
print(mean_squared_error(y_test, pred_final))
```

3. Bagging: Ini juga dikenal sebagai metode bootstrapping. Model dasar dijalankan pada tas untuk mendapatkan distribusi yang adil dari seluruh himpunan data. Tas adalah bagian dari himpunan data bersama dengan pengganti untuk membuat ukuran kantong sama dengan seluruh himpunan data. Output akhir terbentuk setelah menggabungkan output semua model dasar.

Algoritma:

1. *Buat beberapa himpunan data dari himpunan data kereta dengan memilih pengamatan dengan penggantian*
2. *Jalankan model dasar pada setiap himpunan data yang dibuat secara independen*
3. *Gabungkan prediksi semua model dasar untuk masing-masing output akhir*

Bagging biasanya hanya menggunakan satu model dasar (XGBoost Regressor yang digunakan dalam kode di bawah ini).

```
# Mengimpor modul utilitas
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Mengimpor model pembelajaran mesin untuk prediksi
import xgboost as xgb

# mengimpor modul bagging
from sklearn.ensemble import BaggingRegressor

# Memuat kumpulan data kereta dalam Dataframe dari file train_data.csv
df = pd.read_csv("train_data.csv")

# mendapatkan data target dari dataframe
target = df["target"]
```

```
# Mendapatkan data kereta dari dataframe
train = df.drop("target")

# Memisahkan antara data pelatihan ke dalam himpunan data pelatihan dan
validasi
X_train, X_test, y_train, y_test = train_test_split(
    train, target, test_size=0.20)

# menginisialisasi model pengantongan menggunakan XGboost sebagai model
dasar dengan parameter default
model = BaggingRegressor(base_estimator=xgb.XGBRegressor())

# model pelatihan
model.fit(X_train, y_train)

# memprediksi output pada himpunan data pengujian
pred = model.predict(X_test)

# mencetak kesalahan kuadrat rata-rata antara nilai riil dan nilai prediksi
print(mean_squared_error(y_test, pred_final))
```

4. Boosting: Boosting adalah metode berurutan - bertujuan untuk mencegah model dasar yang salah memengaruhi output akhir. Alih-alih menggabungkan model dasar, metode ini berfokus pada membangun model baru yang bergantung pada model sebelumnya. Model baru mencoba menghapus kesalahan yang dibuat oleh model sebelumnya. Masing-masing model ini disebut pembelajar lemah. Model akhir (alias pembelajar kuat) dibentuk dengan mendapatkan rata-rata tertimbang dari semua pembelajar yang lemah.

Algoritma:

1. *Ambil bagian dari himpunan data kereta.*
2. *Melatih model dasar pada himpunan data tersebut.*
3. *Gunakan model ketiga untuk membuat prediksi pada seluruh himpunan data.*
4. *Hitung kesalahan menggunakan nilai prediksi dan nilai aktual.*
5. *Inisialisasi semua titik data dengan bobot yang sama.*
6. *Tetapkan bobot yang lebih tinggi ke titik data yang diprediksi dengan salah.*
7. *Buat model lain, buat prediksi menggunakan model baru sedemikian rupa sehingga kesalahan yang dibuat oleh model sebelumnya dapat dikurangi/diperbaiki.*

8. *Demikian pula, buat beberapa model--masing-masing model berturut-turut mengoreksi kesalahan model sebelumnya.*
9. *Model akhir (pembelajar kuat) adalah rata-rata tertimbang dari semua model sebelumnya (pembelajar lemah).*

```
# Mengimpor modul utilitas
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Mengimpor model pembelajaran mesin untuk prediksi
from sklearn.ensemble import GradientBoostingRegressor

# Memuat kumpulan data kereta dalam Dataframe dari file train_data.csv
df = pd.read_csv("train_data.csv")

# mendapatkan data target dari dataframe
target = df["target"]

# Mendapatkan data kereta dari dataframe
train = df.drop("target")

# Memisahkan antara data pelatihan ke dalam himpunan data pelatihan dan
validasi
X_train, X_test, y_train, y_test = train_test_split(
    train, target, test_size=0.20)

# menginisialisasi modul boosting dengan parameter default
model = GradientBoostingRegressor()

# melatih model pada himpunan data kereta
model.fit(X_train, y_train)

# memprediksi output pada himpunan data pengujian
pred_final = model.predict(X_test)

# mencetak kesalahan kuadrat rata-rata antara nilai riil dan nilai prediksi
print(mean_squared_error(y_test, pred_final))
```

Catatan: Scikit-learn menyediakan beberapa modul/metode untuk metode ansambel. Harap dicatat bahwa keakuratan suatu metode tidak menunjukkan satu metode lebih unggul dari yang lain. Artikel ini bertujuan untuk memberikan pengantar singkat tentang metode ansambel - bukan untuk membandingkan di antara mereka. Programmer harus menggunakan metode yang sesuai dengan data.