"Meningkatkan Performa Ensembling Machine Learning dengan Penerapan Teknik Python Closure pada Batch Data Processor"

Elia Meylani Simanjuntak[1], Priska Silvia Ferantiana[2], Residen Nusantara[3], Anwar Muslim[4], Muhammad Zaky Zaiddan[5]
Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera
email : *elia.122450026@student.itera.ac.id*, *priska.122450053@student.itera.ac.id*, *residen.122450080@student.itera.ac.id*, *anwar.122450117@student.itera.ac.id*, *muhammad.122450119@student.itera.ac.id*

Abstrak

Batch data processor digunakan untuk mengumpulkan, memproses, dan mengelola data dalam satu waktu. *Batch data processor* terdiri atas *closure* dan *ensembling* dimana keduanya merujuk pada proses dari *batch data* tersebut. Pada analisis ini, kita akan membahas mengenai konsep dan implementasi dari *batch data processor closure* dan *ensembling*. *Ensemble learning* adalah cara untuk mencapai konsensus dalam prediksi dengan menggabungkan karakteristik penting dari dua atau lebih model. Beberapa strategi *ensembling* yang umum digunakan untuk *batch data processor* adalah *Voting, Baging (Bootstrap Aggregating), Boosting, Stacking (Meta-Learning), Blending*. Model ensemble adalah salah satu teknik yang sering digunakan dalam *machine learning* untuk meningkatkan performa prediksi. Ensemble model menggabungkan beberapa model pembelajaran mesin (misalnya, *decision tree*, *random forest, gradient boosting,* dll.) untuk membuat prediksi yang lebih akurat daripada model tunggal. Analisis performa model ensemble adalah proses evaluasi untuk mengukur seberapa baik ensemble model bekerja dalam memprediksi data baru. Penerapan teknik Python dalam proses pemrosesan data batch menawarkan manfaat besar dalam fleksibilitas pengelolaan state model, optimisasi kode, skalabilitas yang lebih baik, dan penanganan yang lebih mudah. Closure dapat meningkatkan fleksibilitas dan efisiensi dalam proses pemrosesan data batch. Closure dapat membantu mengelola state dan parameter dalam situasi ini, serta meningkatkan skalabilitas dan generalisasi pengolahan data batch.

Kata Kunci: *Batch data processor, Closure, Ensembling, Python*

PENDAHULUAN

1) Latar Belakang

Jumlah data yang dihasilkan oleh berbagai bidang saat ini terus meningkat seiring dengan perkembangan teknologi. Pengolahan dan analisis data secara efisien dan efektif menjadi tantangan utama yang harus dihadapi. Salah satu cara untuk mengatasi tantangan ini yaitu dengan menerapkan *batch data processor*.

Batch data processor digunakan untuk mengumpulkan, memproses, dan mengelola data dalam satu waktu. *Batch data processor* terdiri atas *closure* dan *ensembling* dimana keduanya merujuk pada proses dari *batch data* tersebut. Pada analisis ini, kita akan membahas mengenai konsep dan implementasi dari *batch data processor closure* dan *ensembling* (Liu, 2022). Kita akan mengetahui bagaimana konsep ini mengoptimalkan pengolahan data, meningkatkan efisien, dan menambah wawasan dari data tersebut.

2) Rumusan Masalah

a. Bagaimana konsep dasar dari *batch data processor*?
b. Bagaimana penerapan *closure* dan *ensembling* dari *batch data processor* dalam analisis data?
c. Bagaimana kontribusi *closure* dna *ensembling* dari *batch data processor* dalam analisis data?

3) Tujuan

a. Mengetahui konsep dasar dari *batch data processor*.
b. Mengimplementasikan *closure* dan *ensembling* dari *batch data processor* dalam analisis data.
c. Menganalisis kinerja konsep *closure* dan *ensembling* dari *batch data processor* dalam analisis data.

METODE

1) Konsep *Closure* dalam Python

Sebelum melihat *closure*, kita harus memahami fungsi bersarang dan variabel non-lokal. Dalam Python, fungsi bersarang dapat mengakses variabel dari lingkup yang melingkupinya. Variabel non-lokal ini tidak dapat diakses dalam Python di luar ruang lingkupnya. Contoh berikut dapat menunjukkan hal ini:

```
# Python program to illustrate
# nested functions
def outerFunction(text):

  def innerFunction():
    print(text)

  innerFunction()


if __name__ == '__main__':
  outerFunction('Hey!')
```

```
Hey!
```

Gambar 1. Fungsi bersarang dan Variabel non-lokal

Kode diatas menunjukkan bahwa InnerFunction() dapat diakses dengan mudah di dalam tubuh outerFunction, tetapi tidak di luarnya. Akibatnya, innerFunction() dianggap sebagai fungsi bersarang yang menggunakan teks sebagai variabel non-lokal di sini.

```
# Python program to illustrate
# closures
def outerFunction(text):

  def innerFunction():
    print(text)

  # Note we are returning function
  # WITHOUT parenthesis
  return innerFunction

if __name__ == '__main__':
  myFunction = outerFunction('Hey!')
  myFunction()
```

```
Hey!
```

Gambar 2. *Python Closures*

Seperti yang ditunjukkan pada kode di atas, *Closure* Python memungkinkan untuk memanggil fungsi di luar ruang lingkupnya. Fungsi *innerFunction* hanya memiliki ruang lingkup di dalam fungsi luar, tetapi dengan penutupan Python, kita dapat dengan mudah memperluas cakupannya untuk memanggil fungsi di luar cakupannya.

Dalam Python, *closure* adalah objek fungsi yang mengingat nilai dalam cakupan yang melingkupi terlepas dari fakta bahwa nilai tersebut tidak ada dalam memori. Ini adalah catatan yang menyimpan fungsi bersama dengan lingkungan: pemetaan yang mengaitkan setiap variabel bebas fungsi (variabel yang digunakan secara lokal tetapi didefinisikan dalam ruang lingkup penutup) dengan nilai atau referensi yang diikat oleh nama saat penutupan dibuat. *Closure*, berbeda dengan fungsi biasa, memungkinkan fungsi untuk mengakses variabel yang ditangkap melalui salinan nilai atau referensi penutup, bahkan ketika fungsi dipanggil di luar ruang lingkup. Karena *Closure in Python* digunakan sebagai fungsi pemanggilan kembali, penutupan menawarkan semacam penyembunyian data, yang membantu mengurangi penggunaan variabel global. *Closures* dalam *Python* terbukti efektif ketika kita memiliki sedikit fungsi dalam kode kita. Namun, jika kita perlu banyak fungsi, maka gunakanlah kelas (OOP).

2) Strategi Ensembling untuk Batch Data Processor

Ensemble learning adalah cara untuk mencapai konsensus dalam prediksi dengan menggabungkan karakteristik penting dari dua atau lebih model. Kerangka kerja pembelajaran kelompok akhir lebih kuat daripada model individu yang membentuk kelompok karena varians kesalahan prediksi yang lebih kecil. *Ensemble learning* bertujuan untuk mengumpulkan data yang saling melengkapi dari berbagai model kontribusi—yaitu, kerangka kerja ensemble berhasil ketika model kontribusi beragam secara statistik. (Dietterich, 2022)

Strategi ensembling untuk pemroses batch data adalah cara untuk meningkatkan kualitas prediksi atau analisis data pada skala besar dengan menggabungkan hasil dari beberapa model atau metode pembelajaran mesin (machine learning). Ini dicapai dengan mengumpulkan prediksi dari beberapa model dan menggabungkannya menjadi satu prediksi yang lebih akurat dan kuat daripada setiap model digunakan secara terpisah. Beberapa strategi *ensembling* yang umum digunakan untuk *batch data processor* adalah sebagai berikut:

a) *Voting* (Pemungutan Suara): Strategi ini menggunakan beberapa model berbeda pada data yang sama dan menghitung prediksi dari setiap model. Untuk menghasilkan prediksi akhir, hasil dari analisis ini dibandingkan dengan metode voting mayoritas atau berat.

b) *Baging (Bootstrap Aggregating)*:Teknik ini menggunakan penggantian (bootstrap) untuk membuat beberapa subset acak dari data pelatihan. Pada setiap subset, model yang sama atau berbeda dilatih. Untuk menghasilkan prediksi akhir, prediksi dari setiap model dikumpulkan dan digabungkan.

c) *Boosting*: Strategi ini menghasilkan model yang lebih kuat dengan memperkuat beberapa model yang lemah, seperti pohon keputusan sederhana, secara berurutan. Prediksi dari setiap model yang diperkuat kemudian digabungkan untuk mendapatkan hasil akhir.

d) *Stacking (Meta-Learning)*: Strategi ini melatih berbagai model pada data yang sama, dan prediksi dari model-model tersebut digunakan sebagai fitur untuk melatih model agregasi (meta-model), yang kemudian digunakan untuk membuat prediksi akhir.

e) *Blending*: Teknik ini lebih mudah digunakan daripada stacking. Beberapa data digunakan untuk melatih beberapa model, dan beberapa data lainnya digunakan untuk menguji model. Setelah itu, bobot yang ditetapkan sebelumnya digunakan untuk menggabungkan prediksi dari model-model ini.

Tergantung pada dataset dan masalah yang dihadapi, setiap strategi ensembling memiliki kelebihan dan kekurangan. Kombinasi strategi ensembling yang tepat dapat meningkatkan akurasi dan ketahanan model pada batch data processor.

3) Implementasi dan Pengujian Model Ensemble

a) Persiapan Data

- Menggunakan dataset iris dari skicit-learn
- Memisahkan data menjadi fitur (x) dan target (y)

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 3. Kode Persiapan Data

b) Pemilihan Model

- Menggunakan beberapa model yang berbeda, seperti Decision Tree, Random Forest, dan Gradient Boosting.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

model_dt = DecisionTreeClassifier(random_state=42)
model_rf = RandomForestClassifier(random_state=42)
model_gb = GradientBoostingClassifier(random_state=42)
```

Gambar 4. Kode Pemilihan Model

c) Pelatihan Model
- Melatih setiap model pada data pelatihan.

```
model_dt.fit(X_train, y_train)
model_rf.fit(X_train, y_train)
model_gb.fit(X_train, y_train)
```

```
GradientBoostingClassifier
GradientBoostingClassifier(random_state=42)
```

Gambar 5. Kode dan Output Pelatihan Model

d) Pembuatan *Ensemble*
- Membuat *ensemble* dengan menggunakan metode voting yang menggabungkan ketiga model yang telah dilatih.

```
from sklearn.ensemble import VotingClassifier

ensemble = VotingClassifier(estimators=[
    ('dt', model_dt),
    ('rf', model_rf),
    ('gb', model_gb)
], voting='hard')
```

Gambar 6. Kode Pembuatan *Ensemble*

e) Pengujian Model Ensemble
- Menguji performa model ensemble pada data pengujian yang belum pernah dilihat sebelumnya.

```
ensemble.fit(X_train, y_train)
ensemble_score = ensemble.score(X_test, y_test)
print(f"Accuracy of Ensemble: {ensemble_score}")
```

```
Accuracy of Ensemble: 1.0
```

Gambar 7. Kode dan Output Pengujian Model Ensemble

4) Validasi dan Optimalisasi

Validasi *Cross-Validation*: Untuk memastikan generalisasi model kelompok, validasi silang dilakukan setelah pengujian awal. Ini membantu menentukan apakah model bekerja dengan baik secara keseluruhan atau hanya pada data khusus. Optimalisasi *Hyperparameter*: *Hyperparameter* model kelompok dipenjara dan dioptimalkan untuk meningkatkan kinerja. Ini mencakup penggunaan metode seperti pencarian grid atau pencarian acak untuk menemukan kombinasi *hyperparameter* yang ideal.

```
from sklearn.model_selection import GridSearchCV

params = {
    'rf__n_estimators': [50, 100, 200],
    'gb__n_estimators': [50, 100, 200],
    'gb__learning_rate': [0.05, 0.1, 0.2]
}

grid_search = GridSearchCV(estimator=ensemble, param_grid=params, cv=3)
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
best_score = grid_search.best_score_

print(f"Best Parameters: {best_params}")
print(f"Best Cross-Validation Score: {best_score}")
```

```
Best Parameters: {'gb__learning_rate': 0.05, 'gb__n_estimators': 50, 'rf__n_estimators': 50}
Best Cross-Validation Score: 0.9333333333333332
```

Gambar 8. Kode dan Output Validasi dan Optimalisasi

HASIL DAN PEMBAHASAN

1) Analisis Perfoma Model Ensemble

Model ensemble adalah salah satu teknik yang sering digunakan dalam machine learning untuk meningkatkan performa prediksi. Ensemble model menggabungkan beberapa model pembelajaran mesin (misalnya, decision tree, random forest, gradient boosting, dll.) untuk membuat prediksi yang lebih akurat daripada model tunggal. Analisis performa model ensemble adalah proses evaluasi untuk mengukur seberapa baik ensemble model bekerja dalam memprediksi data baru. Berikut adalah beberapa langkah yang biasanya dilakukan dalam analisis performa model ensemble:

a) Pembentukan ensemble model: langkah pertama adalah memilih model-model individu yang akan disatukan dalam ensemble. model-model ini bisa berasal dari algoritma yang berbeda atau variasi dari algoritma yang sama dengan parameter yang berbeda.
b) Pengukuran kinerja: ensemble model dievaluasi menggunakan metrik kinerja yang sesuai dengan tugas yang dihadapi. misalnya, untuk masalah klasifikasi, metrik seperti akurasi, presisi, recall, atau f1-score bisa digunakan. sedangkan untuk masalah regresi, metrik seperti mean squared error (mse) atau mean absolute error (mae) bisa digunakan.
c) Analisis kinerja individu: setelah ensemble model terbentuk, penting untuk menganalisis kinerja model-model individu yang menyusun ensemble. ini membantu memahami apakah ada model yang mendominasi atau model-model yang kurang berkontribusi pada kinerja keseluruhan.
d) Analisis keragaman: salah satu alasan utama di balik keberhasilan ensemble adalah keberagaman model-model yang digunakan. jika model-model individu dalam ensemble terlalu mirip satu sama lain, ensemble mungkin tidak memberikan peningkatan kinerja yang signifikan. oleh karena itu, analisis keragaman digunakan untuk memastikan bahwa model-model individu memiliki kecenderungan untuk membuat kesalahan yang berbeda.
e) Penyesuaian hyperparameter: ensemble model seringkali memiliki hyperparameter yang perlu disesuaikan. analisis performa membantu dalam penyesuaian hyperparameter ini untuk meningkatkan kinerja ensemble.
f) Validasi silang: untuk memastikan bahwa kinerja ensemble tidak terlalu bergantung pada pembagian tertentu dari data, teknik validasi silang seperti validasi silang lipatan (cross-validation) digunakan.
g) Analisis ketergantungan fitur: ensemble model juga dapat digunakan untuk menganalisis ketergantungan fitur (feature dependency) dalam data. ini membantu dalam pemahaman lebih lanjut tentang bagaimana fitur-fitur berinteraksi satu sama lain dalam membuat prediksi.
h) Analisis overfitting: meskipun ensemble model cenderung lebih tahan terhadap overfitting daripada model tunggal, masih mungkin untuk terjadi overfitting terutama

jika model-model individu di-overfit terhadap data pelatihan. analisis performa membantu dalam mendeteksi dan mengatasi overfitting ini.

i) Visualisasi dan interpretasi: terakhir, hasil analisis performa dapat divisualisasikan dan diinterpretasikan untuk memahami secara lebih dalam bagaimana ensemble model bekerja dan faktor-faktor apa yang mempengaruhi kinerjanya.

Dengan melakukan analisis performa model ensemble dengan seksama, pengguna dapat mendapatkan wawasan yang berharga tentang kinerja model dan meningkatkan kemampuan prediktifnya.

2) Penerapan Closure dalam Proses Batch Data Processor

Closure dapat meningkatkan fleksibilitas dan efisiensi dalam proses pemrosesan data batch. Closure dapat membantu mengelola state dan parameter dalam situasi ini, serta meningkatkan skalabilitas dan generalisasi pengolahan data batch. Misalnya, Anda dapat menggunakan Closure untuk membuat fungsi yang mempertahankan state tertentu antara pemrosesan batch data. Ini memungkinkan Anda untuk dengan mudah mengakses dan memanipulasi data di setiap iterasi. Ini juga meningkatkan modularitas dan keterbacaan kode dengan membagi logika pemrosesan data ke dalam unit yang dapat digunakan kembali

3) Keuntungan dan Tantangan

Penerapan teknik Python dalam proses pemrosesan data batch menawarkan manfaat besar dalam fleksibilitas pengelolaan state model, optimisasi kode, skalabilitas yang lebih baik, dan penanganan yang lebih mudah. Namun, ada beberapa masalah yang mungkin muncul. Ini termasuk kesulitan untuk memahami konsep closure bagi pengembang yang kurang berpengalaman, kemungkinan bahwa kinerja akan menurun jika tidak diimplementasikan dengan benar, kesulitan untuk memecahkan masalah terkait, dan kekurangan dukungan dan dokumentasi yang memadai. Pengguna closure dapat mengoptimalkan kinerja ensembling machine learning dalam proses batch data processor dengan lebih baik dengan memahami manfaat dan masalah ini.

PENUTUP

Simpulan

Kesimpulan yang dapat diambil bahwa penggunaan teknik Python closure dalam proses ensembling pada prosessor data batch dapat memberikan peningkatan yang signifikan dalam fleksibilitas, optimisasi kode, skalabilitas, dan penanganan yang efisien terhadap state dan parameter model. Namun, ada masalah seperti pemahaman yang sulit, kemungkinan kinerja yang lebih buruk jika tidak dilaksanakan dengan benar, dan kebutuhan akan dukungan dan dokumentasi yang cukup.

Saran

Saran untuk penelitian lanjutan adalah sebagai berikut:

a. Untuk meningkatkan pemahaman tentang manfaat dan masalah penggunaan teknik pembukaan Python dalam konteks ensembling pada batch data processor, lakukan penelitian lebih lanjut.
b. mengembangkan teknik atau alat bantu yang dapat membantu pengembang yang kurang berpengalaman memahami dan menerapkan teknik closure.
c. Evaluasi lebih lanjut tentang kinerja dan efektivitas teknik closure dalam skenario-skenario nyata pada pemroses data batch dengan volume data yang besar.
d. Meningkatkan dokumentasi dan dukungan terkait dengan penggunaan closure dalam konteks pembelajaran mesin pada pemroses data batch untuk membantu pengembang dalam implementasi yang efektif dan efisien.
e. Mempelajari strategi atau metode lain yang dapat digunakan bersama dengan closure unlocking.

Dengan melakukan langkah-langkah ini, penelitian selanjutnya diharapkan dapat menghasilkan pemahaman yang lebih mendalam serta solusi yang lebih optimal untuk penerapan teknik closure Python dalam meningkatkan kinerja pembelajaran mesin ensembling pada batch data processor.

REFERENSI

Dietterich, T. G. (2022). *Ensemble Learning.* USA, Amerika Serikat: Oregon State University.

Liu, Y. Z. (2022). Causal ML: Python Package for causal inference machine learning. *original software publication.*