

✓ Modul 7 - AutoML

✓ Instalasi Autogluon

Modul ini bersumber dari dokumentasi situs Autogluon dengan beberapa modifikasi.

Pada bagian ini, Anda akan menginstall AutoGluon di local komputer Anda, dalam kasus ini kita akan menggunakan Google Colab sebagai mesin komputasinya. Jalankan kode berikut untuk menginstall Autogluon menggunakan package manager PIP.

```
!python -m pip install --upgrade pip
!pip install autogluon
```

```
↵ ^C
↵ ^C
```

✓ AutoML dengan kasus tabular dataset

Dalam parktikum ini, kita akan menggunakan `TabularPredictor` di library AutoGluon untuk memprediksi nilai kolom target berdasarkan kolom lain dalam kumpulan data tabular. Selain data tabular AutoGluon dapat digunakan pada tipe data yang lain seperti citra, teks, audio, dan yang lainnya. Anda diharapkan bisa bereksplorasi dengan tipe data yang lain.

Pastikan AutoGluon telah diinstal, lalu import `TabularDataset` dan `TabularPredictor` dari AutoGluon. Proses yang akan dilalui diawali dengan proses untuk memuat data lalu melatih model dan membuat prediksi.

```
from autogluon.tabular import TabularDataset, TabularPredictor
```

✓ Dataset

kita akan menggunakan dataset dari [Nature issue 7887: AI-guided intuition for math theorems](#). Tujuannya adalah untuk memprediksi *signature* berdasarkan atribut-atribut penjelasnya. Kita mengambil 10 ribu sampel sebagai data latih dan 5 ribu sampel sebagai data uji [data asli](#). Dengan sampel yang lebih kecil dari data asli akan mempercepat praktikum ini, namun AutoGluon dapat menangani kumpulan data lengkap jika diinginkan.

Dataset ini dapat langsung dimuat dari URL yang diberikan. `TabularDataset` AutoGluon adalah subkelas pandas [DataFrame](#), sehingga metode `DataFrame` apa pun dapat digunakan pada `TabularDataset`.

Silakan jalankan dan scroll ke kanan untuk melihat seluruh kolom dan label signature dari dataset ini.

```
data_url = 'https://raw.githubusercontent.com/mli/ag-docs/main/knot_theory/'
train_data = TabularDataset(f'{data_url}train.csv')
train_data.head()
```

```
↵
```

	Unnamed: 0	chern_simons	cuspid_volume	hyperbolic_adjoint_torsion_degree	hyperbolic_torsion_degree	injectivity_radius	longitudinal_t
0	70746	0.090530	12.226322	0	10	0.507756	
1	240827	0.232453	13.800773	0	14	0.413645	
2	155659	-0.144099	14.761030	0	14	0.436928	
3	239963	-0.171668	13.738019	0	22	0.249481	
4	90504	0.235188	15.896359	0	10	0.389329	

Target kita adalah kolom "signature", yang memiliki 18 bilangan bulat unik. Meskipun pandas tidak mengenali tipe data ini dengan benar sebagai kategorikal, AutoGluon akan memperbaiki masalah ini secara otomatis.

```
# nama kolom target
label = 'signature'
train_data[label].describe()
```



signature

count	10000.000000
mean	-0.022000
std	3.025166
min	-12.000000
25%	-2.000000
50%	0.000000
75%	2.000000
max	12.000000

dtype: float64

▼ Pelatihan

Kita buat obyek `TabularPredictor` dengan menentukan nama kolom label, lalu melatih dataset dengan `TabularPredictor.fit()`. Kita tidak perlu menentukan parameter lain. AutoGluon akan mengenali ini sebagai tugas klasifikasi multi-kelas, melakukan rekayasa fitur otomatis, melatih beberapa model, dan kemudian menyatukan model untuk membuat predictor akhir.

```
predictor = TabularPredictor(label=label).fit(train_data)
```



```

0.965    = Validation score (accuracy)
0.14s    = Training runtime
0.0s     = Validation runtime
AutoGluon training complete, total runtime = 229.8s ... Best model: WeightedEnsemble_L2 | Estimated inference throughput: 1456.4 rows/
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("/content/AutoGluonModels/ag-20250321_033510")

```

Pelatihan model memerlukan waktu beberapa menit atau kurang tergantung pada CPU Anda. Anda dapat membuat pelatihan lebih cepat dengan menentukan argumen `time_limit`. Misalnya, `fit(..., time_limit=60)` akan menghentikan latihan setelah 60 detik. Batas waktu yang lebih tinggi umumnya akan menghasilkan performa prediksi yang lebih baik, dan batas waktu yang terlalu rendah akan mencegah AutoGluon untuk melatih dan menggabungkan serangkaian model yang baik.

Dari output hasil pelatihan di atas, terdapat beberapa model yang secara otomatis dicoba oleh AutoGluon. AutoGluon mendapatkan model yang paling baik seperti yang tertampil pada output console. Semua model (tidak hanya best model) tersimpan pada folder AutoGluon. Anda dapat melihatnya dengan mengklik icon folder di sebelah kiri, Anda akan menemukan model-model yang telah ditrain oleh AutoGluon di folder AutoGluonModels.

Anda dapat melihat sekali lagi urutan model dari hasil pelatihan dengan memanggil method `leaderboard`.

▼ Pertanyaan 1:

1. Sebutkan 3 Model yang paling akurat dari beberapa model yang telah dicoba di atas oleh AutoGluon?
2. Model apa yang memiliki proses training yang paling cepat?
3. Model apa yang memiliki proses inferensi yang paling cepat?

```
predictor.leaderboard()
```

	model	score_val	eval_metric	pred_time_val	fit_time	pred_time_val_marginal	fit_time_marginal	stack_level	can_i
0	WeightedEnsemble_L2	0.964965	accuracy	0.685921	109.954163	0.001038	0.139485	2	
1	XGBoost	0.956957	accuracy	0.431871	14.069431	0.431871	14.069431	1	
2	CatBoost	0.955956	accuracy	0.013549	68.422242	0.013549	68.422242	1	
3	LightGBM	0.955956	accuracy	0.139202	8.642485	0.139202	8.642485	1	
4	RandomForestEntr	0.949950	accuracy	0.095747	7.811282	0.095747	7.811282	1	
5	LightGBMLarge	0.949950	accuracy	0.401065	15.354229	0.401065	15.354229	1	
6	ExtraTreesGini	0.946947	accuracy	0.105546	2.472330	0.105546	2.472330	1	
7	LightGBMXt	0.945946	accuracy	0.221250	12.042691	0.221250	12.042691	1	
8	RandomForestGini	0.944945	accuracy	0.096350	7.222713	0.096350	7.222713	1	
9	ExtraTreesEntr	0.942943	accuracy	0.105981	2.603799	0.105981	2.603799	1	
10	NeuralNetTorch	0.941942	accuracy	0.011221	69.205441	0.011221	69.205441	1	
11	NeuralNetFastAI	0.940941	accuracy	0.023802	12.722519	0.023802	12.722519	1	
12	KNeighborsUnif	0.223223	accuracy	0.016697	3.533675	0.016697	3.533675	1	
13	KNeighborsDist	0.213213	accuracy	0.015041	0.024559	0.015041	0.024559	1	

Anda dapat memuat model yang telah disimpan menggunakan perintah berikut:

```
predictor = TabularPredictor.load("AutoGluonModels/ag-xxx-xxx/")
```

Path silakan disesuaikan dengan hasil masing-masing

▼ Prediksi /inferensi

Setelah kita memiliki model yang sesuai dengan dataset pelatihan, kita dapat memuat dataset terpisah untuk digunakan sebagai prediksi dan evaluasi.

```
test_data = TabularDataset(f'{data_url}test.csv')
```

```
# Buang kolom label atau target pada dataset sebagai berikut
y_pred = predictor.predict(test_data.drop(columns=[label]))
```

```
# menampilkan hasil prediksi
y_pred.head()
```

```
➦ Loaded data from: https://raw.githubusercontent.com/mli/ag-docs/main/knot\_theory/test.csv | Columns = 19 / 19 | Rows = 5000 -> 5000
/usr/local/lib/python3.11/dist-packages/fastai/learner.py:455: UserWarning: load_learner` uses Python's insecure pickle module, which ca
If you only need to load model weights and optimizer state, use the safe `Learner.load` instead.
  warn("load_learner` uses Python's insecure pickle module, which can execute malicious arbitrary code when loading. Only load files you

signature
-----
0          -4
1          -2
2           0
3           4
4           2

dtype: int64
```

✓ Evaluasi

Kita dapat mengevaluasi predictor pada data uji menggunakan fungsi `evaluate()`, yang mengukur seberapa baik kinerja predictor kita pada data yang tidak digunakan untuk menyesuaikan model.

```
predictor.evaluate(test_data, silent=True)
```

```
➦ /usr/local/lib/python3.11/dist-packages/fastai/learner.py:455: UserWarning: load_learner` uses Python's insecure pickle module, which ca
If you only need to load model weights and optimizer state, use the safe `Learner.load` instead.
  warn("load_learner` uses Python's insecure pickle module, which can execute malicious arbitrary code when loading. Only load files you
{'accuracy': 0.9478,
 'balanced_accuracy': 0.754478262473782,
 'mcc': 0.9360368834449522}
```

✓ Menggunakan model yang lain

Proses prediksi dan evaluasi secara default akan menggunakan model dengan performa terbaik yang didapatkan dari data validasi saat proses training. Namun, jika kita ingin menggunakan model yang lain, kita bisa memberikan argumen `model=model_name` pada saat proses prediksi dan evaluasi.

`model_name` yang dapat digunakan hanya model yang telah diikutsertakan dalam proses training. Untuk melihat model yang dapat digunakan, kita bisa melihat leaderboard seperti yang telah dilakukan sebelumnya atau memanggil method `get_model_name` pada obyek predictor.

```
predictor.model_names()
```

```
➦ ['KNeighborsUnif',
 'KNeighborsDist',
 'NeuralNetFastAI',
 'LightGBMXT',
 'LightGBM',
 'RandomForestGini',
 'RandomForestEntr',
 'CatBoost',
 'ExtraTreesGini',
 'ExtraTreesEntr',
 'XGBoost',
 'NeuralNetTorch',
 'LightGBMLarge',
 'WeightedEnsemble_L2']
```

Prediksi dengan menggunakan XGBoost model.

```
# Buang kolom label atau target pada dataset sebagai berikut
y_pred = predictor.predict(test_data.drop(columns=[label]),
                           model='XGBoost')

# menampilkan hasil prediksi
y_pred.head()
```

	signature
0	-4
1	-2
2	0
3	4
4	2

dtype: int64

Prediksi dengan menggunakan XGBoost model.

```
predictor.evaluate(test_data, model='XGBoost', silent=True)
```

```
{'accuracy': 0.9448,
 'balanced_accuracy': 0.7445352845015228,
 'mcc': 0.9323703476874563}
```

✓ Tugas 1

1. Buatlah model menggunakan AutoML untuk Dataset auto-mpg dataset.

- Anda bisa mendapatkan dataset pada link berikut: <https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv>
- Model ini merupakan tugas regresi, dengan kolom target mpg.
- Secara garis besar langkah-langkah yang akan digunakan sama dengan proses klassifikasi di atas. AutoML akan menentukan secara otomatis tasknya berdasarkan tipe data pada kolom target.
- Langkah-langkah pengerjaan di kerjakan pada sel kode di bawah ini.

✓ Deskripsi data:

1. Title: Auto-Mpg Data

2. Sources: (a) Origin: This dataset was taken from the StatLib library which is

maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition.

(c) Date: July 7, 1993

3. Past Usage:

- See 2b (above)
- Quinlan, R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.

4. Relevant Information:

This dataset is a slightly modified version of the dataset provided in the StatLib library. In line with the use by Ross Quinlan (1993) in predicting the attribute "mpg", 8 of the original instances were removed because they had unknown values for the "mpg" attribute. The original dataset is available in the file "auto-mpg.data-original".

"The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes." (Quinlan, 1993)

5. Number of Instances: 398

6. Number of Attributes: 9 including the class attribute

7. Attribute Information:

1. mpg: continuous
2. cylinders: multi-valued discrete
3. displacement: continuous
4. horsepower: continuous

5. weight: continuous
6. acceleration: continuous
7. model year: multi-valued discrete
8. origin: multi-valued discrete
9. car name: string (unique for each instance)

8. Missing Attribute Values: horsepower has 6 missing values

1. Membuat training data berdasarkan URL yang diberikan

```
data_url = 'https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv'
train_data = TabularDataset(data_url)
train_data.head()
```

```
# handle missing value
print("Before: ")
print(train_data.isnull().sum())
train_data["horsepower"] = train_data["horsepower"].fillna(train_data["horsepower"].mean())
print("After: ")
print(train_data.isnull().sum())
```

Loaded data from: <https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv> | Columns = 7 / 7 | Rows = 398 -> 398

```
Before:
mpg          0
cylinders    0
displacement 0
horsepower   2
weight       0
acceleration 0
model-year   0
dtype: int64
After:
mpg          0
cylinders    0
displacement 0
horsepower   0
weight       0
acceleration 0
model-year   0
dtype: int64
```

2. Menentukan kolom target

```
label = 'mpg'
train_data[label].describe()
```

```
mpg
count 398.000000
mean  23.514573
std    7.815984
min    9.000000
25%   17.500000
50%   23.000000
75%   29.000000
max   46.600000

dtype: float64
```

3. Fitting model dengan AutoML

```
# Train the model
predictor = TabularPredictor(label="mpg").fit(train_data)
```



```

    0.01s = Validation runtime
Fitting model: KNeighborsDist ...
-4.3253 = Validation score (-root_mean_squared_error)
0.01s = Training runtime
0.01s = Validation runtime
Fitting model: LightGBMXt ...
-2.7173 = Validation score (-root_mean_squared_error)
0.36s = Training runtime
0.0s = Validation runtime
Fitting model: LightGBM ...
-2.374 = Validation score (-root_mean_squared_error)
0.54s = Training runtime
0.01s = Validation runtime
Fitting model: RandomForestMSE ...
[1000] valid_set's rmse: 2.38129
-2.4911 = Validation score (-root_mean_squared_error)
0.63s = Training runtime
0.08s = Validation runtime
Fitting model: CatBoost ...
-2.2861 = Validation score (-root_mean_squared_error)
1.22s = Training runtime
0.0s = Validation runtime
Fitting model: ExtraTreesMSE ...
-2.4399 = Validation score (-root_mean_squared_error)
0.61s = Training runtime
0.08s = Validation runtime
Fitting model: NeuralNetFastAI ...
-3.0898 = Validation score (-root_mean_squared_error)
0.82s = Training runtime
0.01s = Validation runtime
Fitting model: XGBoost ...
-2.6913 = Validation score (-root_mean_squared_error)
0.75s = Training runtime
0.01s = Validation runtime
Fitting model: NeuralNetTorch ...
-2.6855 = Validation score (-root_mean_squared_error)
1.66s = Training runtime
0.01s = Validation runtime
Fitting model: LightGBMLarge ...
-2.6984 = Validation score (-root_mean_squared_error)
0.6s = Training runtime
0.0s = Validation runtime
Fitting model: WeightedEnsemble_L2 ...
Ensemble Weights: {'CatBoost': 0.667, 'LightGBM': 0.333}
-2.2575 = Validation score (-root_mean_squared_error)
0.01s = Training runtime
0.0s = Validation runtime
AutoGluon training complete, total runtime = 7.8s ... Best model: WeightedEnsemble_L2 | Estimated inference throughput: 7089.3 rows/s
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("/content/AutoGluonModels/ag-20250321_034118")

```

```

# 4. Evaluate minimal 3 model terbaik
#   Gunakan data yang sama untuk evaluasi (karena kita tidak punya data testing)

```

```


leaderboard = predictor.leaderboard(train_data, silent=True)
print(leaderboard.head(3)) # Show top 3 models

```

```

# Evaluate top models
for model in leaderboard["model"].head(3):
    print(f"\nEvaluating Model: {model}")
    print(predictor.evaluate_predictions(y_true=train_data["mpg"], y_pred=predictor.predict(train_data), auxiliary_metrics=True))

```

 /usr/local/lib/python3.11/dist-packages/fastai/learner.py:455: UserWarning: load_learner` uses Python's insecure pickle module, which ca
If you only need to load model weights and optimizer state, use the safe `Learner.load` instead.

warn("load_learner` uses Python's insecure pickle module, which can execute malicious arbitrary code when loading. Only load files you

	model	score_test	score_val	eval_metric \
0	WeightedEnsemble_L2	-1.174316	-2.257539	root_mean_squared_error
1	XGBoost	-1.206658	-2.691255	root_mean_squared_error
2	CatBoost	-1.213043	-2.286121	root_mean_squared_error

	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal \
0	0.048530	0.011285	1.779016	0.002818
1	0.030750	0.007754	0.747223	0.030750
2	0.005915	0.001217	1.223994	0.005915

	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer \
0	0.000444	0.012758	2	True
1	0.007754	0.747223	1	True
2	0.001217	1.223994	1	True

	fit_order
0	12
1	9

```
Evaluating Model: WeightedEnsemble_L2
{'root_mean_squared_error': -1.1743156062027995, 'mean_squared_error': -1.3790171429714486, 'mean_absolute_error': -0.751392803958912, '
Evaluating Model: XGBoost
{'root_mean_squared_error': -1.1743156062027995, 'mean_squared_error': -1.3790171429714486, 'mean_absolute_error': -0.751392803958912, '
Evaluating Model: CatBoost
{'root_mean_squared_error': -1.1743156062027995, 'mean_squared_error': -1.3790171429714486, 'mean_absolute_error': -0.751392803958912, '

```

Pertanyaan 2

1. Berdasarkan hasil pelatihan dan evaluasi pada data auto-mpg, metric apa yang menggambarkan performa model tersebut? Jelaskan tentang metric tersebut!
2. Sebutkan 3 model terbaik dan score performanya!
3. Pilihlah model yang kiranya paling baik dari sisi performa ataupun waktu inferensinya! Jelaskan mengapa Anda memilih model tersebut!

