AutoGluon v1.4 Cheat Sheet

Installation

AutoGluon (GitHub) supports Python 3.9 to 3.12. Installation is available for Linux, MacOS, and Windows. More installation options.

```
pip install autogluon
```

Preparing Data

AutoGluon accepts pandas DataFrames as inputs, where each row stores an example, with columns as features. We use the Kaggle Titanic dataset to demonstrate how to use AutoGluon.

```
import pandas as pd
train_data = pd.read_csv('titanic/train.csv')
```

AutoGluon works with raw data. Little or no data preprocessing, such as removing obvious non-predictive columns, is needed.

```
train_data = train_data.drop(columns=['PassengerId'])
```

Training

Train models to predict the values in the column 'Survived'. The training log will tell you how AutoGluon extracts features, selects, trains, and ensembles models.

```
from autogluon.tabular import TabularPredictor

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

More options to construct a **TabularPredictor** instance (docs):

```
verbosity=3 # Logging (1=warnings, 2=default, 3=verbose)
# The metric used to tune models. All available metrics.
eval_metric='roc_auc'
```

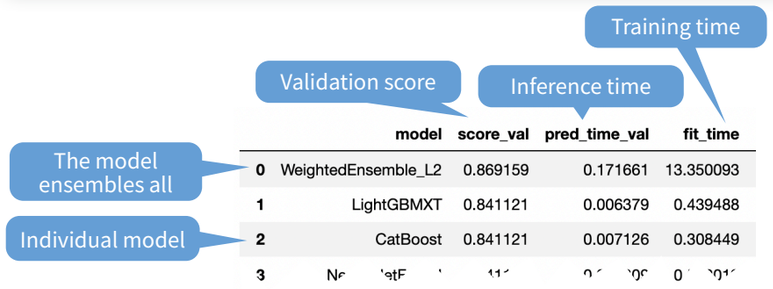
More options for the **fit** method (docs, presets):

```
# Limit the training time, in seconds
time_limit=600
# Better model ensemble for a better accuracy, but
longer training time. All available options.
presets='best_quality'
# Use a separate dataset to tune models.
tuning_data=val_data
# Explore less models. You can fully control the model
search space. All available options.
hyperparameters='very_light'
# Ignore some models.
excluded_model_types=['KNN', 'NN_TORCH']
# Speed up inference.
infer_limit=0.00005 # At most 0.05 ms per row
```

Monitoring

Understand the contribution of each model (docs).

```
predictor.leaderboard(display=True)
```

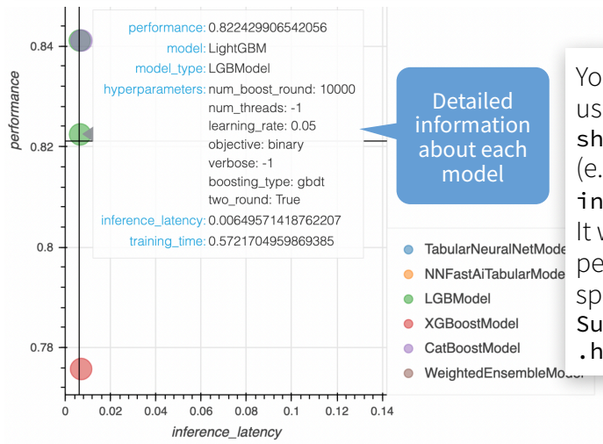


More options for `leaderboard`:

```
# Report metrics on a separate test dataset.
data=test_data
# Evaluate more metrics.
extra_metrics=['accuracy', 'log_loss']
```

Understand more about the trained models (docs).

```
predictor.fit_summary(show_plot=True)
```



You need bokeh to use `show_plot=True` (e.g. `pip install bokeh`). It will generate performance vs speed results in `SummaryOfModels.html`

Understand the importance of each feature (docs).

```
predictor.feature_importance(test_data)
```

Importance score

	importance	stddev	p_value
Name	0.204744	0.020024	0.001587
Pclass	0.041199	0.011695	0.012912
Sex	0.039950	0.010314	0.010751
Age	0.024969	0.004325	0.004926
SibSp	0.016854	0.001873	0.002045

A feature

Predicting

```
test_data = pd.read_csv('titanic/test.csv')
# Predict for each row
predictor.predict(test_data)
# Return the class probabilities for classification
predictor.predict_proba(test_data)
# Evaluate various metrics, it needs test_data to have
the label column
predictor.evaluate(test_data)
```

AutoGluon predicts with the final ensemble model. You can also predict using an individual model.

```
# Get a list of string names
models = predictor.model_names()
# Predict with the 2nd model. Both predict_proba and
evaluate also accept the model argument
predictor.predict(test_data, model=models[1])
```

Deploying

AutoGluon models are saved to disk automatically. You can check logs to find where it is saved, or by checking `predictor.path`.

```
# Load saved model from disk.
predictor = TabularPredictor.load('AutogluonModels/
ag-20250129_004130/')
```

If the inference speed matters, there are multiple ways to accelerate the speed. First, you can force all models in memory.

```
predictor.persist()
```

During training, you can use presets for the `fit` method optimized for fast inference (though may hurt model performance).

```
presets=['good_quality', 'optimize_for_deployment']
```

Finally, clone a minimal artifact for inference.

```
# Will create a minimal artifact for deployment.
predictor.clone_for_deployment(path='prod_model_dir')
# Load the optimized predictor and persist models in mem.
predictor_opt = TabularPredictor.load('prod_model_dir')
predictor_opt.persist()
predictions = predictor_opt.predict(test_data)
```

- Click here for detailed Tabular tutorials.
- For data involving text and images, try out MultiModalPredictor.
- For data involving forecasting, try out TimeSeriesPredictor.
- For effortless production deployments, use AutoGluon Cloud.
- Check the latest version of this cheat sheet at https://auto.gluon.ai/stable/cheatsheet.html
- Any questions? Ask here
- Like what you see? Consider starring AutoGluon on GitHub and following us on twitter to get notified of the latest updates!