간단한 Python 코드만으로 높은 성능의 기계 학습 모델 만들기

김무현
시니어 데이터 사이언티스트
AWS
Common ML problems

- Tabular Prediction
- Image Classification
- Object Detection
- Text Classification

Regression  Classification
How to solve ML problems

1. Preparing dataset
2. Feature engineering
3. Choosing ML algorithm
4. Training Model
5. Evaluating Trained Model
6. Data science, machine learning, deep learning skill and experience required tasks
7. Using Model
Required skills in building ML models

Data science for feature engineering

Machine Learning and Deep Learning for modeling

Model tuning experience

ML and DL toolkits such as scikit-learn, TensorFlow, PyTorch, MXNet
Do I need to learn machine learning or deep learning and ML/DL framework as an application developer or data analyst?
AutoML

“Automated machine learning (AutoML) is the process of automating the process of applying machine learning to real-world problems. AutoML covers the complete pipeline from the raw dataset to the deployable machine learning model.” - Wikipedia

**Automated Machine Learning** provides methods and processes to make Machine Learning available for non-Machine Learning experts, to improve efficiency of Machine Learning and to accelerate research on Machine Learning. - AutoML.org

- Hyperparameter optimization
- Meta-learning (learning to learn)
- Neural architecture search
AutoGluon - AutoML Toolkit for Deep Learning
https://autogluon.mxnet.io

Example using AutoGluon to train and deploy high-performance model on a tabular dataset:

```python
from autogluon import TabularPrediction as task
def train():
    predictor = task.fit(train_data=test_dataset)
def predict():
    predictions = predictor.predict(test_dataset)
```

AutoGluon can be applied just as easily for prediction tasks with image or text data.
AutoGluon for “All”

**Developers/Analysts with no ML skill**
- Automating all ML pipeline
  - feature engineering
  - model selection
  - model training
  - hyperparameter optimization

**ML Experts**
- Quick model prototyping for baseline
- Hyperparameter optimization
- Optimizing custom models

**Researchers**
- Model optimization
- Searching for new architectures
What you can do with AutoGluon

• Quick prototyping achieving the state-of-the-art performance for
  • Tabular prediction
  • Image classification
  • Object detection
  • Text classification

• Customizing model searching

• Hyperparameter optimization on model training in Python or PyTorch

• Neural Architecture Searching
Basic usage of AutoGluon
Simple 3 steps to get the best ML model

Step 1. Prepare your dataset

Step 2. Load the dataset for training ML

Step 3. Call `fit()` to get the best ML model
What happens behind the scene

Loading the dataset for training ML
- ML problem defined (binary/multiple classification or regression)
- Feature engineering for each model being trained
- Missing value handling
- Splitting dataset into training and validation

Calling `fit()` to get the best ML model
- Training models
- Hyperparameter optimization
- Model selection
Common ML problems

Tabular Prediction

Image Classification

Object Detection

Text Classification

Regression  Classification
ML algorithms for Tabular prediction

- Random Forest
- XT (Extremely randomized trees)
- K-nearest neighbors
- CatBoost - gradient boosting on decision trees
  - [https://catboost.ai/](https://catboost.ai/)
- LightGBM
  - [https://lightgbm.readthedocs.io](https://lightgbm.readthedocs.io)
- Neural Network
Let’s prepare a tabular dataset

A structured data stored in CSV format where
- each row represents an example and
- each column presents the measurements of some variable or feature

Files stored either in an Amazon S3 bucket or the local file system

Some data found in “Adult data set (https://archive.ics.uci.edu/ml/datasets/adult)”

<table>
<thead>
<tr>
<th>age</th>
<th>workclass</th>
<th>fnlwgt</th>
<th>education</th>
<th>education-num</th>
<th>marital-status</th>
<th>occupation</th>
<th>relationship</th>
<th>race</th>
<th>sex</th>
<th>capital-gain</th>
<th>capital-loss</th>
<th>hours-per-week</th>
<th>native-country</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Private</td>
<td>178478</td>
<td>Bachelors</td>
<td>13</td>
<td>Never-married</td>
<td>Tech-support</td>
<td>Own-child</td>
<td>White</td>
<td>Female</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>United-States</td>
<td>&lt;=50K</td>
</tr>
<tr>
<td>2</td>
<td>State-gov</td>
<td>61743</td>
<td>9th-10th</td>
<td>3</td>
<td>Never-married</td>
<td>Transport-moving</td>
<td>Not-in-family</td>
<td>White</td>
<td>Male</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>United-States</td>
<td>&lt;=50K</td>
</tr>
<tr>
<td>3</td>
<td>Private</td>
<td>37589</td>
<td>HS-grad</td>
<td>9</td>
<td>Never-married</td>
<td>Other-service</td>
<td>Not-in-family</td>
<td>White</td>
<td>Male</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>United-States</td>
<td>&lt;=50K</td>
</tr>
<tr>
<td>4</td>
<td>?</td>
<td>200235</td>
<td>HS-grad</td>
<td>9</td>
<td>Married-civ-spouse</td>
<td>?</td>
<td>Husband</td>
<td>White</td>
<td>Male</td>
<td>0</td>
<td>0</td>
<td>50</td>
<td>United-States</td>
<td>&gt;50K</td>
</tr>
<tr>
<td>5</td>
<td>Private</td>
<td>224541</td>
<td>7th-8th</td>
<td>4</td>
<td>Married-civ-spouse</td>
<td>Handlers-cleaners</td>
<td>Husband</td>
<td>White</td>
<td>Male</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>El-Salvador</td>
<td>&lt;=50K</td>
</tr>
<tr>
<td>6</td>
<td>Private</td>
<td>178054</td>
<td>Some-college</td>
<td>10</td>
<td>Married-civ-spouse</td>
<td>Sales</td>
<td>Husband</td>
<td>White</td>
<td>Male</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>United-States</td>
<td>&gt;50K</td>
</tr>
<tr>
<td>7</td>
<td>Private</td>
<td>283561</td>
<td>Some-college</td>
<td>10</td>
<td>Married-civ-spouse</td>
<td>Craft-repair</td>
<td>Husband</td>
<td>White</td>
<td>Male</td>
<td>60</td>
<td>0</td>
<td>60</td>
<td>United-States</td>
<td>&gt;50K</td>
</tr>
<tr>
<td>8</td>
<td>Private</td>
<td>173613</td>
<td>HS-grad</td>
<td>9</td>
<td>Divorced</td>
<td>Adm-clerical</td>
<td>Not-in-family</td>
<td>White</td>
<td>Female</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>United-States</td>
<td>&lt;=50K</td>
</tr>
<tr>
<td>9</td>
<td>Private</td>
<td>214617</td>
<td>Some-college</td>
<td>10</td>
<td>Never-married</td>
<td>Handlers-cleaners</td>
<td>Own-child</td>
<td>White</td>
<td>Male</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>United-States</td>
<td>&lt;=50K</td>
</tr>
<tr>
<td>10</td>
<td>Private</td>
<td>84681</td>
<td>Assoc-voc</td>
<td>11</td>
<td>Married-civ-spouse</td>
<td>Sales</td>
<td>Husband</td>
<td>White</td>
<td>Male</td>
<td>0</td>
<td>0</td>
<td>45</td>
<td>United-States</td>
<td>&lt;=50K</td>
</tr>
<tr>
<td>11</td>
<td>Private</td>
<td>225892</td>
<td>Some-college</td>
<td>10</td>
<td>Married-civ-spouse</td>
<td>Craft-repair</td>
<td>Husband</td>
<td>White</td>
<td>Male</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>United-States</td>
<td>&gt;50K</td>
</tr>
</tbody>
</table>
Step 0: Install AutoGluon

# CUDA 10.0 and a GPU for object detection is recommended
# We install MXNet to utilize deep learning models

# For Linux with GPU installed
pip install --upgrade mxnet-cu100

# For Linux without GPU
pip install --upgrade mxnet

# Install AutoGluon package
pip install autogluon
Step 1: Loading dataset

```python
from autogluon import TabularPrediction as task

train_path = 'https://autogluon.s3.amazonaws.com/datasets/AdultIncomeBinaryClassification/train_data.csv'
train_data = task.Dataset(file_path=train_path)
```

- **file_path**: str (optional)
  Path to the data file (may be on local filesystem or URL to cloud S3 bucket).

- **df**: `pandas.DataFrame` (optional)
  If you already have your data in a pandas DataFrame, you can directly provide it by specifying `df`.

- **feature_types**: dict (optional)
  Mapping from column_names to string describing data type of each column. If not specified, AutoGluon's `fit()` will automatically infer what type of data each feature contains.

- **subsample**: int (optional)
  If specified = k, we only keep first k rows of the provided dataset.

- **name**: str (optional)
  Optional name to assign to dataset.
Step 2: Training ML models

predictor = task.fit(train_data, label='class', output_directory='ag-example-out/')

dataset to be used for ML training

column name to be predicted

directory where the trained models are saved

ag-example-out/
  SummaryOfModels.html
  learner.pkl
  models
    CatboostClassifier
      model.pkl
    ExtraTreesClassifierEntr
    ExtraTreesClassifierGini
    KNeighborsClassifierDist
    KNeighborsClassifierUnif
    LightGBMClassifier
    LightGBMClassifierCustom
    NeuralNetClassifier
      net.params
        tabularNN.pkl
        temp_net.params
    RandomForestClassifierEntr
    RandomForestClassifierGini
    trainer.pkl
    weighted_ensemble_l1
parameters of \textit{fit()}

\url{https://autogluon.mxnet.io/api/autogluon.task.html#autogluon.task.TabularPrediction.fit}

```python
def fit(train_data, label, tuning_data=None, output_directory=None, problem_type=None, eval_metric=None,
        hyperparameter_tune=False, feature_prune=False, auto_stack=False, holdout_frac=None,
        num_bagging_folds=0, num_bagging_sets=None, stack_ensemble_levels=0,
        hyperparameters = {
            'NN': {'num_epochs': 500},
            'GBM': {'num_boost_round': 10000},
            'CAT': {'iterations': 10000},
            'RF': {'n_estimators': 300},
            'XT': {'n_estimators': 300},
            'KNN': {},
            'custom': ['GBM'],
        },
        enable_fit_continuation=False,
        time_limits=None, num_trials=None, search_strategy='random', search_options={},
        nthreads_per_trial=None, ngpus_per_trial=None, dist_ip_addr=[], visualizer='none',
        verbosity=2, **kwargs):
```

- ‘random’ (random search)
- ‘skopt’ (SKopt Bayesian optimization)
- ‘grid’ (grid search)
- ‘hyperband’ (Hyperband)
- ‘rl’ (reinforcement learner)

- ‘mxboard’, ‘tensorboard’, ‘none’
### Step 3: Evaluate the model

```python
test_path = 'https://autogluon.s3.amazonaws.com/datasets/AdultIncomeBinaryClassification/test_data.csv'
test_data = task.Dataset(file_path=test_path)
leaderboard = predictor.leaderboard(test_data)
```

<table>
<thead>
<tr>
<th>model</th>
<th>score_test</th>
<th>score_val</th>
<th>fit_time</th>
<th>pred_time</th>
<th>stack_level</th>
</tr>
</thead>
<tbody>
<tr>
<td>weighted_ensemble_l1</td>
<td>0.876958</td>
<td>0.8748</td>
<td>1.799241</td>
<td>0.001847</td>
<td>1</td>
</tr>
<tr>
<td>CatboostClassifier</td>
<td>0.876548</td>
<td>0.8740</td>
<td>39.542085</td>
<td>0.039164</td>
<td>0</td>
</tr>
<tr>
<td>LightGBMClassifier</td>
<td>0.875729</td>
<td>0.8672</td>
<td>7.899164</td>
<td>0.033295</td>
<td>0</td>
</tr>
<tr>
<td>LightGBMClassifierCustom</td>
<td>0.874092</td>
<td>0.8676</td>
<td>34.620379</td>
<td>0.046849</td>
<td>0</td>
</tr>
<tr>
<td>RandomForestClassifierEntr</td>
<td>0.860375</td>
<td>0.8504</td>
<td>10.123323</td>
<td>0.226658</td>
<td>0</td>
</tr>
<tr>
<td>RandomForestClassifierGini</td>
<td>0.859044</td>
<td>0.8504</td>
<td>10.425733</td>
<td>0.236528</td>
<td>0</td>
</tr>
<tr>
<td>NeuralNetClassifier</td>
<td>0.858327</td>
<td>0.8544</td>
<td>265.441416</td>
<td>0.479654</td>
<td>0</td>
</tr>
<tr>
<td>ExtraTreesClassifierGini</td>
<td>0.846863</td>
<td>0.8388</td>
<td>7.431412</td>
<td>0.232303</td>
<td>0</td>
</tr>
<tr>
<td>ExtraTreesClassifierEntr</td>
<td>0.845122</td>
<td>0.8408</td>
<td>7.236258</td>
<td>0.222955</td>
<td>0</td>
</tr>
<tr>
<td>KNeighborsClassifierUnif</td>
<td>0.774695</td>
<td>0.7736</td>
<td>0.324976</td>
<td>0.120106</td>
<td>0</td>
</tr>
<tr>
<td>KNeighborsClassifierDist</td>
<td>0.762105</td>
<td>0.7644</td>
<td>0.319923</td>
<td>0.111614</td>
<td>0</td>
</tr>
</tbody>
</table>
```

*evaluation metric specified in fit(*)*
Step 4: Use the model in your app

predictor = task.load('ag-example-out/')  # loading models saved in the specified directory

test_path = 'https://autogluon.s3.amazonaws.com/datasets/AdultIncomeBinaryClassification/test_data.csv'
test_data = task.Dataset(file_path=test_path)
y_test = test_data['class']
test_data_nolabel = test_data.drop(labels=['class'], axis=1)

y_pred = predictor.predict(test_data_nolabel)  # predicting on the given dataset using the best model

predicted values in numpy array by default
parameters of *predict()*

https://autogluon.mxnet.io/api/autogluon.task.html#autogluon.task.tabular_prediction.TabularPredictor.predict

**dataset :** `TabularDataset` or `pandas.DataFrame`
The dataset to make predictions for. Should contain same column names as training Dataset and follow same format.

**model :** str (optional)
The name of the model to get predictions from. Defaults to None, which uses the highest scoring model on the validation set.

**as_pandas :** bool (optional)
Whether to return the output as a pandas Series (True) or numpy array (False)

**use_pred_cache :** bool (optional)
Whether to used previously-cached predictions for table rows we have already predicted on before

**add_to_pred_cache :** bool (optional)
Whether these predictions should be cached for reuse in future `predict()` calls on the same table rows
Demo
from autogluon import ImageClassification as task

# Loading dataset
dataset = task.Dataset('./data/shopeeiet/train')

# Train image classification models
time_limits = 10 * 60 # 10mins
classifier = task.fit(dataset, 
    time_limits=time_limits, 
    ngpus_per_trial=1)

# Test the trained model
test_dataset = task.Dataset('./data/shopeeiet/test')
inds, probs, probs_all = classifier.predict(test_dataset)
For advanced features of AutoGluon

Other AutoML tasks
- Image classification
- Object detection
- Text classification

Hyperparameter optimization
- Hyperparameter search space and search algorithm customization
- Distributed search

Neural architecture search
- ENAS/ProxylessNAS
Efficient NAS on Target Hardware: ProxylessNAS

https://autogluon.mxnet.io/tutorials/nas/enas_mnist.html

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
<th>Mobile Latency</th>
<th>Hardware-aware</th>
<th>No Proxy</th>
<th>No Repeat</th>
<th>Search cost (GPU hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV1 [16]</td>
<td>70.6</td>
<td>89.5</td>
<td>113ms</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>Manual</td>
</tr>
<tr>
<td>MobileNetV2 [30]</td>
<td>72.0</td>
<td>91.0</td>
<td>75ms</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>Manual</td>
</tr>
<tr>
<td>NASNet-A [38]</td>
<td>74.0</td>
<td>91.3</td>
<td>183ms</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>48,000</td>
</tr>
<tr>
<td>AmoebaNet-A [29]</td>
<td>74.5</td>
<td>92.0</td>
<td>190ms</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>75,000</td>
</tr>
<tr>
<td>MnasNet [31]</td>
<td>74.0</td>
<td>91.8</td>
<td>76ms</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>40,000</td>
</tr>
<tr>
<td>MnasNet (our impl.)</td>
<td>74.0</td>
<td>91.8</td>
<td>79ms</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>40,000</td>
</tr>
<tr>
<td>Proxyless-G (mobile)</td>
<td>71.8</td>
<td>90.3</td>
<td>83ms</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>200</td>
</tr>
<tr>
<td>Proxyless-G + LL</td>
<td>74.2</td>
<td>91.7</td>
<td>79ms</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>200</td>
</tr>
<tr>
<td>Proxyless-R (mobile)</td>
<td>74.6</td>
<td>92.2</td>
<td>78ms</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 2: ProxylessNAS achieves state-of-the-art accuracy (%) on ImageNet (under mobile latency constraint ≤ 80ms) with 200× less search cost in GPU hours. “LL” indicates latency regularization loss. Details of MnasNet’s search cost are provided in appendix C.

(Source) PROXYLESSNAS: DIRECTNEURALARCHITECTURESEARCH ONTARGETTASK ANDHARDWARE
More toolkits for developers

GluonCV

Deep Java Library
Open source library to build and deploy Deep Learning in Java

- Framework Agnostic
- Built for Java developers
- Ease of deployment

```java
Translator<BufferedImage, Classifications> translator = new MyTranslator();
try (Predictor<BufferedImage, Classifications> predictor = model.newPredictor(translator)) {
    return predictor.predict(img);
}
```

```java
private static final class MyTranslator implements Translator<BufferedImage, Classifications> {
    private List<String> classes;
    public MyTranslator() {
        classes = IntStream.range(0, 10).mapToObj(String::valueOf).collect(Collectors.toList());
    }
```
Resources

AutoGluon (https://autogluon.mxnet.io)

GluonCV (https://gluon-cv.mxnet.io)

AWS Computer Vision: Getting started with GluonCV (https://www.coursera.org/learn/aws-computer-vision-gluoncv)

Deep Java Library (https://djl.ai)


Machine Learning on AWS (https://ml.aws)

Amazon Machine Learning Solutions Lab (https://aws.amazon.com/ml-solutions-lab)
AWS 머신러닝(ML) 교육 및 자격증

Amazon의 개발자와 데이터 과학자를 교육하는 데 직접 활용 되었던 커리큘럼을 기반으로 학습하세요!

전체 팀을 위한 머신러닝 교육
비즈니스 의사 결정자, 데이터 과학자, 개발자, 데이터 플랫폼 엔지니어 등 역할에 따라 제공되는 맞춤형 학습 경로를 확인하세요.

원하는 방법으로! 교육 유연성 제공
약 65개 이상의 온라인 과정 및 AWS 전문 강사를 통해 실습과 실적용의 기회가 제공되는 강의실 교육이 준비되어 있습니다.

전문성에 대한 검증
업계에서 인정받는 'AWS 공인 머신러닝 – 전문분야’ 자격증을 통해 머신러닝 모델을 구축, 학습, 튜닝 및 배포하는 데 필요한 전문 지식이 있음을 입증할 수 있습니다.

여러분의 소중한 피드백을 기다립니다!
강연 평가 및 설문 조사에 참여해 주세요.
감사합니다