

Engenharia de Machine Learning

2025-03-17

Automated Machine Learning (AutoML)

O que é AutoML?

Ferramentas que facilita o processo de desenvolvimento com padrões pré-definidos para as principais tarefas:

- Preprocessamento e limpeza de dados
- Construção de Features
- Seleção de Modelos
- Ajuste de Hiperparâmetros
- Críticas e Interpretação de Modelos

PyCaret

PyCaret ajuda nas seguintes tarefas:

- Preparação dos dados
- Treino de Modelos
- Ajuste de Hiperparâmetros
- Análise e Interpretabilidade

Exemplo de uso para classificador:

```
1 # Load dataset
2 iris = load_iris()
3 df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
4 df['target'] = iris.target
5
6 # Initialize setup
7 clf_setup = setup(data=df, target='target', session_id=123,
8                     normalize=True, polynomial_features=True, silent=True)
9
10 # Compare all models
11 best_models = compare_models()
12
13 # Get the best model
14 best_model = models()['lr']
```

Instalação PyCaret

```
1 pip install "pycaret[models,mlops]"
```

O que pode ir dentro dos `[]` (extras):

models

- xgboost - XGBoost
- catboost - Gradient Boosting com árvores de decisão
- kmodes - K-modas e K-protótipos para clusterização
- mlxtend - Múltiplas técnicas de ensemble
- statsforecast - Previsão de séries temporais
- scikit-learn-intelex - Otimização do sklearn para Intel

mlops

- mlflow - Registro e rastreamento de experimentos
- gradio - Interface de usuário (competidor do Streamlit)
- evidently - Monitoramento de modelos

tuners

- tune-sklearn
- ray[tune]
- hyperopt
- optuna
- scikit-optimize

parallel

- dask
- distributed
- fugue
- flask
- Werkzeug

analysis

- shap
- interpret
- umap-learn
- ydata-profiling
- explainerdashboard
- fairlearn

Benefícios e Limitações

Benefícios

- Diminui a quantidade de código desenvolvida
- Explora um conjunto grande de modelos, mais rápido
- Customizável

Limitações

- Desempenho pode não ser a melhor em domínios específicos
- Cuidado com análise “Caixa Preta”!
- Tempo de computação pode ser alto
- Não substitui conhecimento especialista

PyCaret - Comparação de Modelos

Este exemplo usa a abordagem “OOP” do PyCaret.

1. Criar uma instância do objeto `ClassificationExperiment`
2. Chamar o método `setup` para configurar o experimento

- `data` - DataFrame com os dados
- `session_id` - Seed para geradores de números aleatórios

3. Usar `compare_models` para comparar os modelos

```
1 from pycaret.classification import *
2
3 exp = ClassificationExperiment()
4
5 exp.setup(data=train, target='Survived', session_id=session_id)
6
7 best_model = exp.compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.8237	0.8747	0.6993	0.8469	0.7614	0.6242	0.6355	0.0160
catboost	CatBoost Classifier	0.8157	0.8749	0.6990	0.8277	0.7537	0.6084	0.6179	0.2270
lightgbm	Light Gradient Boosting Machine	0.8098	0.8718	0.7238	0.7969	0.7548	0.6003	0.6063	61.3150
xgboost	Extreme Gradient Boosting	0.7957	0.8758	0.7133	0.7730	0.7380	0.5714	0.5767	0.1110
rf	Random Forest Classifier	0.7936	0.8621	0.7138	0.7757	0.7368	0.5683	0.5765	0.0270
lr	Logistic Regression	0.7897	0.8533	0.7005	0.7683	0.7278	0.5579	0.5641	0.2100
ridge	Ridge Classifier	0.7897	0.8528	0.7002	0.7700	0.7282	0.5581	0.5648	0.0060
lda	Linear Discriminant Analysis	0.7897	0.8528	0.7002	0.7700	0.7282	0.5581	0.5648	0.0050
ada	Ada Boost Classifier	0.7816	0.8431	0.7040	0.7498	0.7209	0.5425	0.5481	0.0150
nb	Naive Bayes	0.7796	0.8210	0.7293	0.7342	0.7284	0.5435	0.5468	0.1350
et	Extra Trees Classifier	0.7736	0.8277	0.6893	0.7507	0.7120	0.5268	0.5349	0.0230
dt	Decision Tree Classifier	0.7676	0.7541	0.6838	0.7438	0.7039	0.5142	0.5234	0.0050
knn	K Neighbors Classifier	0.6974	0.7258	0.5764	0.6473	0.6057	0.3625	0.3672	0.1680
svm	SVM - Linear Kernel	0.6814	0.7228	0.5814	0.5952	0.5588	0.3327	0.3446	0.0060
dummy	Dummy Classifier	0.5932	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1600
qda	Quadratic Discriminant Analysis	0.5088	0.6695	0.8862	0.5052	0.6011	0.1378	0.1427	0.0060

PyCaret - Otimização de Hiperparâmetros

Use o método `tune_model` para otimização de hiperparâmetros.

Por padrão, o PyCaret usa o algoritmo `RandomSearchCV` para otimização. Todo modelo possui uma grade de hiperparâmetros pré-definidos pelo PyCaret.

Parâmetros de interesse:

- `n_iter` - Número de iterações no `RandomSearchCV`
- `optimize` - Função a ser otimizada
- `params` - Dicionário com uma grade de hiperparâmetros customizada
- `search_library` - Biblioteca de otimização a ser usada

```
1 from pycaret.classification import *
2
3 exp = ClassificationExperiment()
4
5 exp.setup(data=train, target='Survived', session_id=session_id)
6
7 model = exp.create_model('lr')
8
9 tuned_model = exp.tune_model(model, n_iter=100, optimize='F1')
```

PyCaret - Exemplo com scikit-optimize

Otimização através de busca bayesiana para maximização de métricas de classificação.

- Exemplo com modelo `GradientBoostingClassifier`

Modelo	F1 médio
<code>model</code>	0.7626
<code>randcv_model</code>	0.7783
<code>bscv_model</code>	0.7702

```
1 from pycaret.classification import *
2
3 exp = ClassificationExperiment()
4
5 exp.setup(data=train, target='Survived', session_id=12345)
6
7 # Criar modelo e treina com hiperparâmetros padrão
8 model = exp.create_model('gbc')
9
10 # Otimização padrão do PyCaret com RandomCV
11 randcv_model = exp.tune_model(model, n_iter=100, optimize='F1')
12
13 # Otimização Bayesiana (isso pode demorar)
14 bscv_model = exp.tune_model(
15     model,
16     optimize='F1',
17     n_iter=100,
18     search_library='scikit-optimize',
19 )
```

Exemplo com Titanic

