PyCaret Cheat sheet

pycaret.org

Tutorials

Clustering
Beginner

Anomaly
Beginner

Association Rules
Beginner

NLP
Beginner
Intermediate

Regression
Beginner
Intermediate

Classification
Binary (Beginner)
Binary (Intermediate)
Multiclass (Beginner)

Time Series
Beginner

Unsupervised Learning

Supervised Learning

Time Series Analysis

Clustering

```
setup(data=df)
create_model(*)
assign_model(*)
plot_model(**)
evaluate_model()
tune_model()
predict_model()
deploy_model()
save_model()
load_model()
pull()
models()
get_metrics()
add_metric()
remove_metric()
get_logs()
get_config()
set_config()
save_config()
load_config()
get_clusters()

* model:
‘kmeans’
‘ap’
‘meanshift’
‘sc’
‘hclust’
‘dbscan’
‘optics’
‘birch’
‘kmodes’

** plot =
‘cluster’
‘tsne’
‘elbow’
‘silhouette’
‘distance’
‘distribution’
```

Anomaly Detection

```
setup(data=df)
create_model(*)
assign_model(*)
plot_model(**)
evaluate_model()
tune_model()
predict_model()
deploy_model()
save_model()
load_model()
models()
get_logs()
get_config()
set_config()
save_config()
load_config()
get_outliers()


* model:
‘abod’
‘cluster’
‘histogram’
‘knn’
‘lof’
‘svm’
‘pca’
‘mcd’
‘sod’
‘sos’

** plot=
‘tsne’
‘umap’
```

Association Rule

```
setup(data=df,
     transaction_id=‘column’,
     item_id=‘column’)
create_model()
plot_model(**)
get_rules()

** plot=
‘2d’  ‘3d’
```

NLP

```
setup(data=df)
create_model(*)
assign_model()
plot_model()
tune_model()
evaluate_model()
save_model()
load_model()
models()
get_logs()
get_config()
set_config()
get_topics()

* model:
‘lda’
‘lsi’
‘hdp’
‘rp’
‘nmf’
‘frequency’
‘distribution’
‘bigram’
‘trigram’
‘sentiment’
‘pos’
‘tsne’
‘topic_model’
‘topic_distribution’
‘wordcloud’
‘umap’
```

Regression & Classification

```
setup(data=df,
      target=‘column’)
models()*
compare_models()
create_model()*
tune_model()
ensemble_model()
blend_models()
stack_models()
plot_model()**
evaluate_model()
interpret_model()
1 calibrate_model()
1 optimize_threshold()
assign_model()
calibrate_model()
optimize_threshold()
get_leaderboard()
dashboard()
deep_check()
eda()
check_fairness()
get_leaderboard()
predict_model()
finalize_model()
convert_model()
deploy_model()
save_model()
save_experiment()
save_config()
load_config()
load_model()
convert_model()
create_api()
create_docker()
create_app()
automl()
pull()
models()
get_logs()
get_config()
set_config()
get_system_logs()
```

```
* model:
(classification)

‘lr’
‘knn’
‘nb’
‘dt’
‘svm’
‘rbfsvm’
‘gpc’
‘mlp’
‘ridge’
‘rf’
‘qda’
‘ada’
‘gbc’
‘lda’
‘et’
‘xgboost’
‘lightgbm’
‘catboost’

** plot=
‘auc’
‘threshold’
‘pr’
‘error’
‘class_report’
‘boundary’
‘rfe’
‘learning’
‘manifold’
‘calibration’
‘vc’
‘dimension’
‘feature’
‘feature_all’
‘parameter’
‘lift’
‘gain’
‘tree’
‘ks’
‘confusion_matrix’
```

```
* model:
(regression)
‘lr’
‘lasso’
‘ridge’
‘en’
‘lar’
‘llar’
‘omp’
‘br’
‘ard’
‘par’
‘ransac’
‘tr’
‘huber’
‘kr’
‘svm’
‘knn’
‘dt’
‘rf’
‘et’
‘ada’
‘gbr’
‘mlp’
‘xgboost’
‘lightgbm’
‘catboost’

** plot=
‘residuals’
‘error’
‘cooks’
‘rfe’
‘learning’
‘vc’
‘manifold’
‘feature’
‘feature_all’
‘residuals_
interactive’
‘parameter’
‘tree’
```

(1) *classification only*

Time Series

```
setup(data=df)
models(*)
compare_models()
create_model(*)
tune_model()
blend_models()
plot_model()
finalize_model()
deploy_model()
save_model()
load_model()
pull()
models()
predict_model()
get_metrics()
add_metric()
remove_metric()
get_logs()
get_config()
set_config()
save_config()
load_config()

** plot=
‘ts’
‘cv’
‘acf’
‘pacf’
‘decomp_stl’
‘diagnostics’
‘forecast’
‘insample’
‘residuals’
‘train_test_split’
‘decomp_classical’
```

```
* model:
‘naïve’
‘grand_means’
‘snaive’
‘polytrend’
‘arima’
‘auto_arima’
‘exp_smooth’
‘ets’
‘theta’
‘tbats’
‘bats’
‘prophet’
‘lr_cds_dt’
‘en_cds_dt’
‘ridge_cds_dt’
‘lasso_cds_dt’
‘lar_cds_dt’
‘llar_cds_dt’
‘br_cds_dt’
‘huber_cds_dt’
‘par_cds_dt’
‘omp_cds_dt’
‘knn_cds_dt’
‘dt_cds_dt’
‘rf_cds_dt’
‘et_cds_dt’
‘gbr_cds_dt’
‘ada_cds_dt’
‘lightgbm_cds_dt’
```

PYCARET

pycaret.org

Parameters of `setup()` and its *default values*

Clustering

```
data,
preprocess = True,
imputation_type = 'simple',
iterative_imputation_iters = 5,
categorical_features = None,
categorical_imputation = 'mode',
categorical_iterative_imputer = 'lightgbm',
ordinal_features = None,
high_cardinality_features = None,
high_cardinality_method = 'frequency',
numeric_features = None,
numeric_imputation = 'mean',
numeric_iterative_imputer = 'lightgbm',
date_features = None,
ignore_features = None,
normalize = False,
normalize_method = 'zscore',
transformation = False,
transformation_method = 'yeo-johnson',
handle_unknown_categorical = True,
unknown_categorical_method = 'least_frequent',
pca = False,
pca_method = 'linear',
pca_components = None,
ignore_low_variance = False,
combine_rare_levels = False,
rare_level_threshold = 0.1,
bin_numeric_features = None,
remove_multicollinearity = False,
multicollinearity_threshold = 0.9,
remove_perfect_collinearity = False,
group_features = None,
group_names = None,
n_jobs = -1,
use_gpu = False,
custom_pipeline = None,
html = True,
session_id = None,
system_log = True,
log_experiment = False,
experiment_name = None,
log_plots = False,
log_profile = False,
log_data = False,
silent = False,
verbose = True,
profile = False,
profile_kwargs = None
```

Color code

required

optional

Anomaly Detection

```
data,
Preprocess = True,
imputation_type = 'simple',
iterative_imputation_iters = 5,
categorical_features = None,
categorical_imputation = 'mode',
categorical_iterative_imputer = 'lightgbm',
ordinal_features = None,
high_cardinality_features = None,
high_cardinality_method = 'frequency',
numeric_features = None,
numeric_imputation = 'mean',
numeric_iterative_imputer = 'lightgbm',
date_features = None,
ignore_features = None,
normalize = False,
normalize_method = 'zscore',
transformation = False,
transformation_method = 'yeo-johnson',
handle_unknown_categorical = True,
unknown_categorical_method = 'least_frequent',
pca = False,
pca_method = 'linear',
pca_components = None,
ignore_low_variance = False,
combine_rare_levels = False,
rare_level_threshold = 0.1,
bin_numeric_features = None,
remove_multicollinearity = False,
multicollinearity_threshold = 0.9,
remove_perfect_collinearity = False,
group_features = None,
group_names = None,
n_jobs = - 1,
use_gpu = False,
custom_pipeline = None,
html = True,
session_id = None,
system_log = True,
log_experiment = False,
experiment_name = None,
log_plots = False,
log_profile = False,
log_data = False,
silent = False,
verbose = True,
profile = False,
profile_kwargs = None
```

Regression & Classification

```
data = DataFrame, target = 'column_name',
train_size = 0.7,
test_data = None,
preprocess = True,
imputation_type = 'simple',
iterative_imputation_iters = 5,
categorical_features = None,
categorical_imputation = 'constant',
categorical_iterative_imputer = 'lightgbm',
ordinal_features = None,
high_cardinality_features = None,
high_cardinality_method = 'frequency',
numeric_features = None,
numeric_imputation = 'mean',
numeric_iterative_imputer = 'lightgbm',
date_features = None,
ignore_features = None,
normalize = False,
normalize_method = 'zscore',
transformation = False,
transformation_method = 'yeo-johnson',
handle_unknown_categorical = True,
unknown_categorical_method = 'least_frequent',
pca = False,
pca_method = 'linear',
pca_components = None,
ignore_low_variance = False,
combine_rare_levels = False,
rare_level_threshold = 0.1,
bin_numeric_features = None,
remove_outliers = False,
outliers_threshold = 0.05,
remove_multicollinearity = False,
multicollinearity_threshold = 0.9,
remove_perfect_collinearity = True,
create_clusters = False,
cluster_iter = 20,
polynomial_features = False,
polynomial_degree = 2,
trigonometry_features = False,
polynomial_threshold = 0.1,
group_features = None,
group_names = None,
feature_selection = False,
feature_selection_threshold = 0.8,
feature_selection_method = 'classic',
feature_interaction = False,
feature_ratio = False,
interaction_threshold = 0.01,
transform_target = False,
transform_target_method = 'box-cox',
data_split_shuffle = True,
data_split_stratify = False,
fold_strategy = 'kfold',
fold = 10,
fold_shuffle = False,
fold_groups = None,
n_jobs = - 1,
use_gpu = False,
custom_pipeline = None,
html = True,
session_id = None,
log_experiment = False,
experiment_name = None,
log_plots = False,
log_profile = False,
log_data = False,
silent = False,
verbose = True,
profile = False,
profile_kwargs = None
```

Time Series

```
data = [.Series, .DataFrame],
preprocess = True,
imputation_type = 'simple',
fold_strategy = 'expanding',
fold = 3,
fh = 1,
seasonal_period = None,
enforce_pi = False,
n_jobs = -1,
use_gpu = False,
custom_pipeline = None,
html = True,
session_id = None,
system_log = True,
log_experiment = False,
experiment_name = None,
log_plots = False,
log_profile = False,
log_data = False,
verbose = True,
profile = False,
profile_kwargs = None
```

Association Rule

```
data,
transaction_id ='column_name',
item_id = 'column_name',
ignore_items = None,
session_id = None
```

NLP

```
data,
Target = 'column_name',
custom_stopwords = None,
Html = True,
session_id = None,
log_experiment = False,
experiment_name = None,
log_plots = False,
log_data = False,
Verbose = True
```

Cheat sheet

PyCaret is an open source, low-code machine learning library in Python that allows you to go preparing your data to deploying your model within minutes in your choice of notebook environment

Installing PyCaret

```
# install pycaret
pip install pycaret
# install full version of pycaret
pip install pycaret[full]
# install pycaret time series module
pip install pycaret-ts-alpha
```

PyCaret on GPU

```
# uninstall lightgbm CPU
pip uninsstall lightgbm –y
# install lightgbm GPU
  pip install lightgbm --install-option=--gpu
--install-option="--opencl-include-dir=/usr/
/local/include/" --install-option="--opencl-
library=usr/local/cuda/lib64/libOpenCL.so"
```

Run PyCaret on a Docker Container

```
FROM python:3.7-slim
WORKDIR /app
ADD . /app
RUN apt-get update && apt-get install libgomp1
RUN pip install --trusted-host pypi.python.org
-r requirements.txt

CMD pytest # replace it with your entry point
```

PyCaret Tutorials

Classification
Binary classification (Beginner)
Binary classification (Intermediate)
Multiclass classification (Beginner)

Regression
Regression (Beginner)
Regression (Intermediate)

Clustering
Clustering (Beginner)

Anomaly detection
Anomaly detection (Beginner)

Natural Language Processing
NLP (Beginner)
NLP (Intermediate)

Association Rule Mining
Association Rule Mining (Beginner)

Time Series
Time series and forecasting (Beginner)

Loading data from PyCaret's repository

```
# loading data from pycaret
from pycaret.datasets import get_data
data = get_data('dataset_name')
```

Loading data using Pandas

```
# importing pandas
import pandas as pd
df = pd.read_csv(r'dir/file_name.csv')
```

Supervised Learning

Regression and Classification

```
# set up environment
from pycaret.regression import *
from pycaret.classification import *
clf1 = setup(data = df, target='column')
# create and evaluate model
compare_models()
model = create_model(*)
model_tuned = tune_model(model)
ens_model = ensemble_model(model, method=***)
blender = blend_models(top3)
stacker = stack_models(top3)
plot_model(model, plot=**)
evaluate_model(model)
interpret_model(model)
(1) calibrate_model()
(1) optimize_threshold()
# make predictions
df1 = predict_model(model=model, data=df)
# model deployment

final_model = finalize_model(model)
save_model(model, 'saved_model')
model_loaded = load_model('saved_model')
deploy_model(model=model,
model_name=model_final,
platform = 'aws', authentication = {'bucket :
'S3-bucket-name'})
# utils
pull()
models()
get_metrics()
add_metric()
remove_metric()
get_logs()
get_config()

set_config()

save_config()

load_config()

get_leaderboard()
*model:
(regression)        (classification)
'lr'                'lr'
'lasso'             'knn'
'ridge'             'nb'
'en'                'dt'
'lar'               'svm'
'llar'              'rbfsvm'
(regression)        (classification)
'omp'               'gpc'
'br'                'mlp'
'ard'               'ridge'
'par'               'rf'
'ransac'            'qda'
'tr'                'ada'
'huber'             'gbc'
'kr'                'lda'
'svm'               'et'
'knn'               'xgboost'
'dt'                'lightgbm'
'rf'                'catboost'
'et'
'gbr'
'mlp'
'xgboost'
'lightgbm'
'catboost'
**plot=
'residuals_interactive'  'residuals'  'error'
'cooks'  'rfe'  'learning'  'vc'  'manifold'
'feature'  'feature_all'  'parameter'  'tree'
*** method=
'bagging'  'boosting'
(1) classification only
```

Time Series Analysis

```
# set up environment
from pycaret.time_series import *
exp = setup(data = df, fh = 12)
# create and evaluate model

compare_models()

model = create_model(*)
model_tuned = tune_model(model)

blender = blend_models(top3)

plot_model(model, plot=**)
final_model = finalize_model(model)
# make predictions
pred_holdout = predict_model(arima)
pred_unseen = predict_model(finalize_model(
arima), fh=24)
# model deployment
final_model = finalize_model(model)
save_model(model, 'saved_model')
model_loaded = load_model('saved_model')
deploy_model(model=model,
model_name=model_final,
platform = 'aws', authentication = {'bucket :

'S3-bucket-name'})
# utils
pull()
models()
get_metrics()
add_metric()
remove_metric()
get_logs()
get_config()
set_config()
save_config()
```

* model:
'naïve'
'grand_means'
'snaive'
'polytrend'
'arima'
'exp_smooth'
'ets'
'theta'
'tbats'
'bats'
'prophet'
'lr_cds_dt'
'en_cds'
'ridge_cds_dt'
'lasso_cds_dt'
'lar_cds_dt'
'llar_cds_dt'
'br_cds_dt'
'huber_cds_dt'
'par_cds_dt'
'omp_cds_dt'
'knn_cds_dt'
'dt_cds_dt'
'rf_cds_dt'
'et_cds_dt'
'gbr_cds_dt'
'ada_cds_dt'
'lightgbm_cds_dt'

**plot=
'ts'
'cv'
'acf'
'acf'
'pacf'
'decomp_stl'
'diagnostics'
'forecast'
'insample'
'residuals'
'train_test_split'
'decomp_classical'

Unsupervised Learning

Clustering

```
# set up environment
from pycaret.clustering import *

clf1 = setup(data = df)
# create and evaluate model
model = create_model(*)
model_df = assign_model(*)
plot_model(model, plot=**)
evaluate_model(model)
model_tuned = tune_model(model=model,
        supervised_target = 'column_name')
# make predictions
df1 = predict_model(model=model, data = df)
# model deployment

save_model(model, 'saved_model')
model_loaded = load_model('saved_model')
deploy_model(model=model,
model_name=model_final,
platform = 'aws', authentication = {'bucket :
'S3-bucket-name'})
# utils
pull()
models()
get_metrics()
add_metric()
remove_metric()
get_logs()
get_config()
set_config()
save_config()
load_config()
get_clusters()
```

* model:
'kmeans'
'ap'
'meanshift'
'sc'
'hclust'
'dbscan'
'optics'
'birch'
'kmodes'

**plot=
'cluster'
'tsne'
'elbow'
'silhouette'
'distance'
'distribution'

Anomaly Detection

```
# set up environment
from pycaret.anomaly import *
clf1 = setup(data = df)
# create and evaluate model
model = create_model(*)
model_df = assign_model(*)
plot_model(model, plot=**)
evaluate_model(model)
model_tuned = tune_model(model=model,
        supervised_target='column')
# make predictions
df1 = predict_model(model=model, data=df)
# model deployment
save_model(model, 'saved_model')
model_loaded = load_model('saved_model')
deploy_model(model=model,
model_name=model_final,
platform = 'aws', authentication = {'bucket :
'S3-bucket-name'})

# utils
pull()
models()
get_metrics()
add_metric()
remove_metric()
get_logs()
get_config()
set_config()
save_config()
load_config()

get_clusters()
```

* model:
'abod'
'cluster'
'histogram'
'knn'
'lof'
'svm'
'pca'
'mcd'
'sod'
'sos'

**plot=
'tsne'
'umap'

Natural Language Processing

```
# set up environment
from pycaret.nlp import *
clf1 = setup(data=df, target='colunm')
# create and evaluate model
model = create_model(*)
model_df = assign_model(*)
plot_model(model, plot=**)
evaluate_model(model)
model_tuned = tune_model(model=model,
        supervised_target='column')
# model deployment
save_model(model, 'saved_model')
model_loaded = load_model('saved_model')
# utils
pull()
models()
get_logs()
get_config()
set_config()
get_topics()
* model:
'lda'  'lsi'  'hdp'  'rp'  'nmf'
'frequency'  'distribution'  'bigram'
'trigram'  'sentiment'  'pos'  'tsne'
'topic_model'  'topic_distribution'
'wordcloud'  'umap'
** plot=
'tsne'  'umap'
```

Association Rule

```
# set up environment

from pycaret.arules import *
clf1 = setup(data=df, transaction_id='colunm',
        item_id='column')

# create and evaluate model
model = create_model()
plot_model(model, plot='2d')
```

Other Resources

PyCaret Github
PyCaert Slack
Example Notebooks made by contributors
Blog tutorials
Documentation 'The detailed API docs of PyCaret'

Video Tutorials
Discussions 'Have questions?'

Changelog 'Changes and version history'
Roadmap of PyCaret