PYCARET

Introduction to PyCaret and installation

Pedram Jahangiry

PYCARET

Installation on Windows and Google Colab

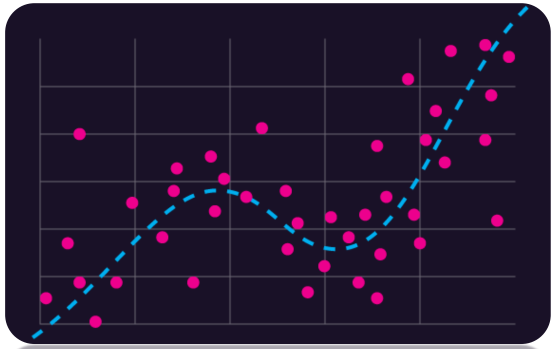
ANACONDA

Google colab

Pedram Jahangiry

PYCARET

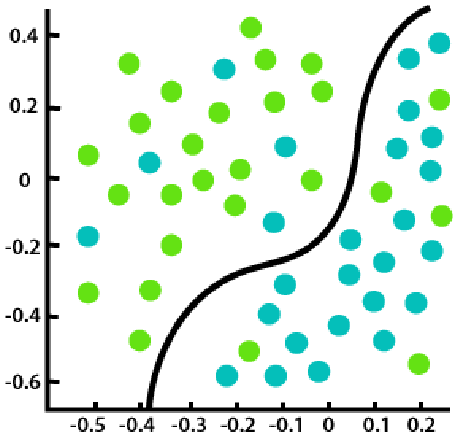
Tutorial notebook for **Machine Learning Regression**



Pedram Jahangiry

PYCARET

Tutorial notebook for **Machine Learning Classification**



Pedram Jahangiry

What is PyCaret?

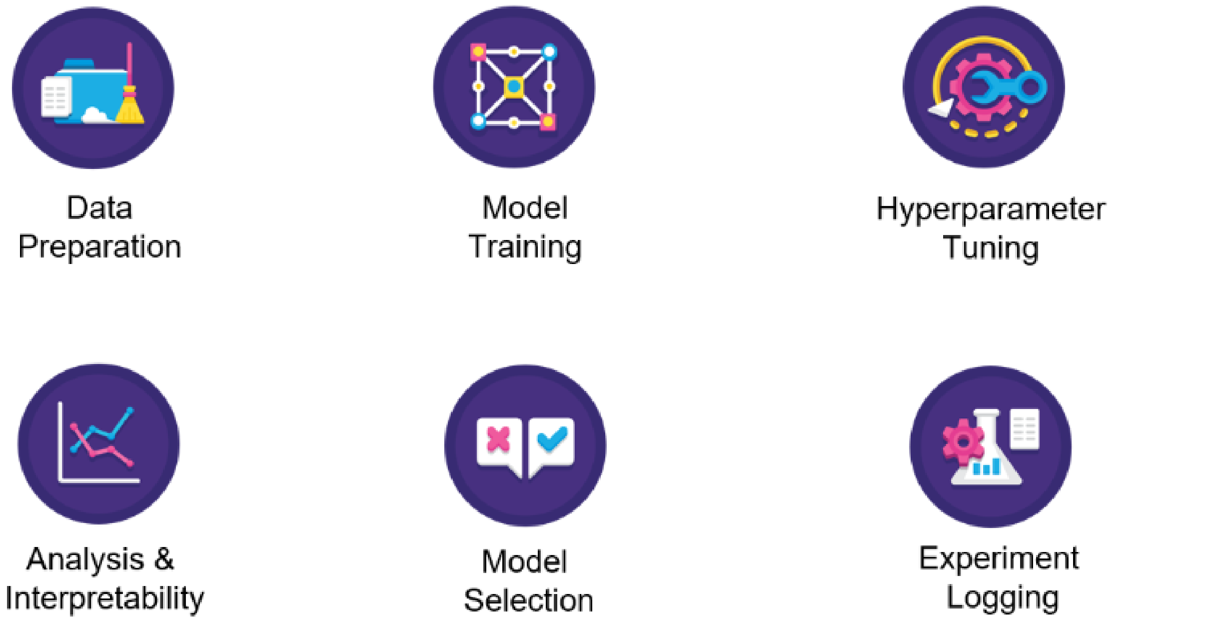
- PyCaret is an open-source, low-code machine learning library in Python that automates machine learning workflows.
- PyCaret can be used to replace hundreds of lines of code with few lines only. You spend less time coding and more time on analysis
- PyCaret is essentially a Python wrapper around several machine learning libraries and frameworks such as scikit-learn, XGBoost, LightGBM, CatBoost, and few more.

- Exploratory Data Analysis
- Data Preprocessing
- Model Training
- Model Explainability
- MLOps

Pedram Jahangiry

PyCaret is ideal for:

- Experienced Data Scientists who want to increase productivity.
- Citizen Data Scientists who prefer a low code machine learning solution.
- Data Science Professionals who want to build rapid prototypes.
- Data Science and Machine Learning students and enthusiasts.



Preprocessing (setup)

Data Preparation	Scale and Transform	Feature Engineering	Feature Selection
• Missing values	• Normalize	• Feature interaction	• Feature Selection
• Data Types	• Feature Transform	• Polynomial Features	• Remove Multicollinearity
• One-Hot Encoding	• Target Transform	• Group Features	• Principal Component Analysis
• Ordinal Encoding		• Bin Numeric Features	• Ignore Low Variance
• Cardinal Encoding		• Combine Rare Levels	
• Handle Unknown Levels		• Create Clusters	
• Target Imbalance			
• Remove outliers			

Data Preparation

Pedram Jahangiry

Model training

PyCaret trains multiple models simultaneously and outputs a table comparing the performance of each model by considering a few performance metrics.

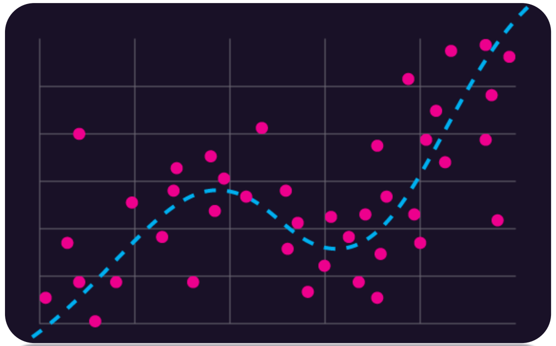
- Creating models: `create_model(‘dt’, fold=n, …)`

- Comparing models: `compare_models(n_select = n, sort=‘Accuracy’, …)`

- Tuning hyperparameters: `tune_model(dt, custom_grid: Optional, …)`

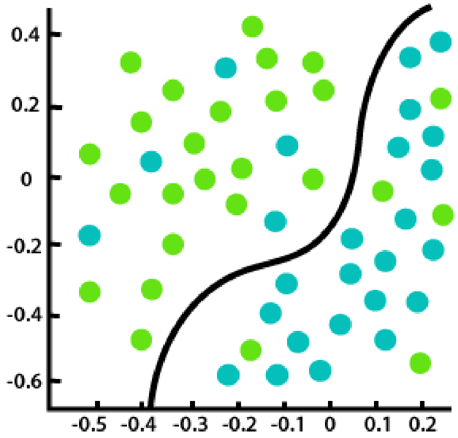
List of models (Regression)

ID	Name
‘lr’	Linear Regression
‘lasso’	Lasso Regression
‘ridge’	Ridge Regression
‘en’	Elastic Net
‘lar’	Least Angle Regression
‘llar’	Lasso Least Angle Regression
‘omp’	Orthogonal Matching Pursuit
‘br’	Bayesian Ridge
‘ard’	Automatic Relevance Determination
‘par’	Passive Aggressive Regressor
‘ransac’	Random Sample Consensus
‘tr’	TheilSen Regressor
‘huber’	Huber Regressor
‘kr’	Kernel Ridge
‘svm’	Support Vector Machine
‘knn’	K Neighbors Regressor
‘dt’	Decision Tree
‘rf’	Random Forest
‘et’	Extra Trees Regressor
‘ada’	AdaBoost Regressor
‘gbr’	Gradient Boosting Regressor
‘mlp’	Multi Level Perceptron
‘xgboost’	Extreme Gradient Boosting
‘lightgbm’	Light Gradient Boosting
‘catboost’	CatBoost Regressor



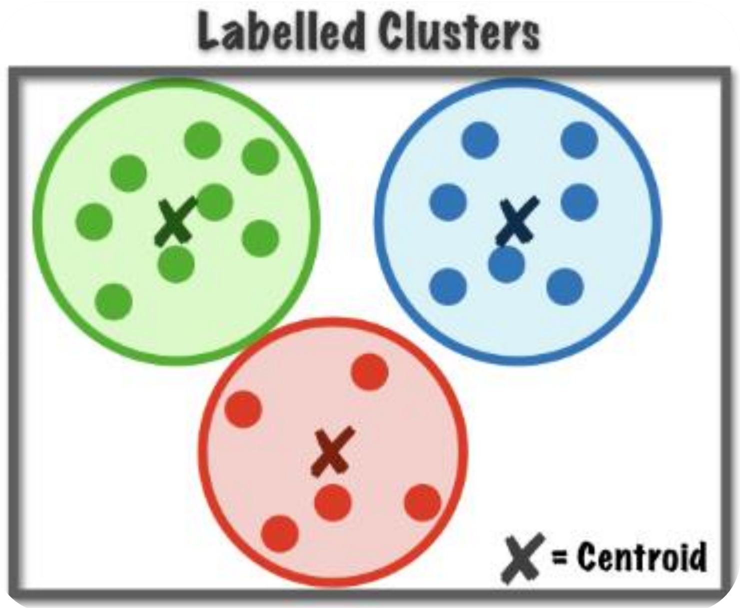
List of models (Classification)

ID	Name
‘lr’	Logistic Regression
‘knn’	K Nearest Neighbour
‘nb’	Naives Bayes
‘dt’	Decision Tree Classifier
‘svm’	SVM – Linear Kernel
‘rbfsvm’	SVM – Radial Kernel
‘gpc’	Gaussian Process Classifier
‘mlp’	Multi Level Perceptron
‘ridge’	Ridge Classifier
‘rf’	Random Forest Classifier
‘qda’	Quadratic Discriminant Analysis
‘ada’	Ada Boost Classifier
‘gbc’	Gradient Boosting Classifier
‘lda’	Linear Discriminant Analysis
‘et’	Extra Trees Classifier
‘xgboost’	Extreme Gradient Boosting
‘lightgbm’	Light Gradient Boosting
‘catboost’	CatBoost Classifier



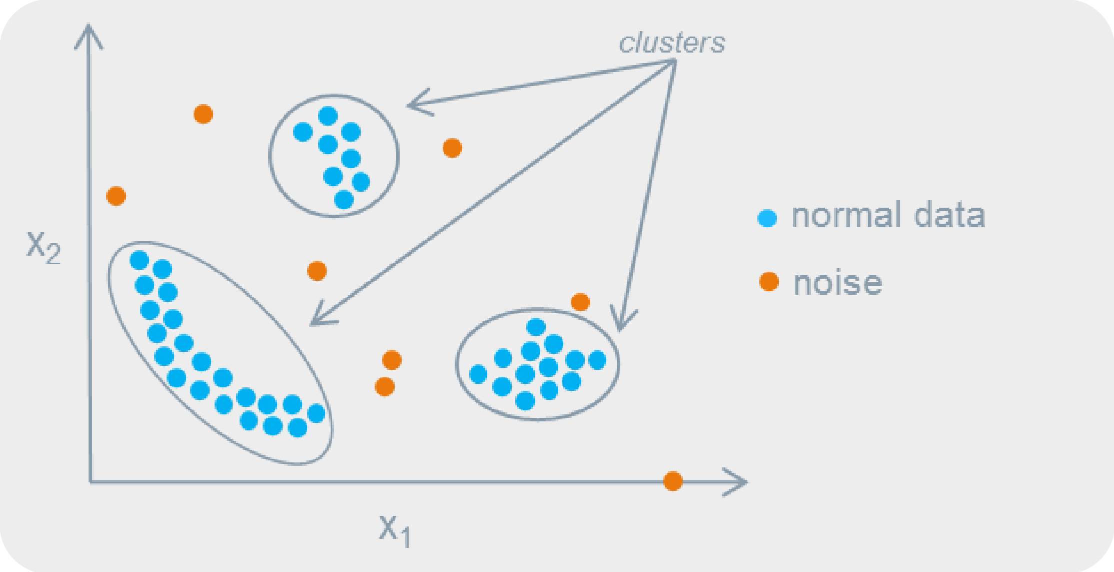
List of models (Clustering)

ID	Name
‘kmeans’	K-Means Clustering
‘ap’	Affinity Propagation
‘meanshift’	Mean shift Clustering
‘sc’	Spectral Clustering
‘hclust’	Agglomerative Clustering
‘dbscan’	Density-Based Spatial Clustering
‘optics’	OPTICS Clustering
‘birch’	Birch Clustering
‘kmodes’	K-Modes Clustering



List of models (Anomaly Detection)

ID	Name
‘abod’	Angle-base Outlier Detection
‘iforest’	Isolation Forest
‘cluster’	Clustering-Based Local Outlier
‘cof’	Connectivity-Based Outlier Factor
‘histogram’	Histogram-based Outlier Detection
‘knn’	k-Nearest Neighbors Detector
‘lof’	Local Outlier Factor
‘svm’	One-class SVM detector
‘pca’	Principal Component Analysis
‘mcd’	Minimum Covariance Determinant
‘sod’	Subspace Outlier Detection
‘sos	Stochastic Outlier Selection



Analysis and interpretability

```
My_model = create_model('Model_name')
```

- plot_model(my_model)
- interpret_model(model)

Analysis & Interpretability

Name	Plot
Area Under the Curve	'auc'
Discrimination Threshold	'threshold'
Precision Recall Curve	'pr'
Confusion Matrix	'confusion_matrix'
Class Prediction Error	'error'
Classification Report	'class_report'
Decision Boundary	'boundary'
Recursive Feature Selection	'rfe'
Learning Curve	'learning'
Manifold Learning	'manifold'
Calibration Curve	'calibration'
Validation Curve	'vc'
Dimension Learning	'dimension'
Feature Importance	'feature'
Model Hyperparameter	'parameter'

Name	Plot
Residuals Plot	'residuals'
Prediction Error Plot	'error'
Cooks Distance Plot	'cooks'
Recursive Feature Selection	'rfe'
Learning Curve	'learning'
Validation Curve	'vc'
Manifold Learning	'manifold'
Feature Importance	'feature'
Model Hyperparameter	'parameter'

Cluster PCA Plot (2d)	'cluster'
Cluster TSnE (3d)	'tsne'
Elbow Plot	'elbow'
Silhouette Plot	'silhouette'
Distance Plot	'distance'
Distribution Plot	'distribution'

Pedram Jahangiry

Finalize, Predict, Save and Deploy model

```
My_model = create_model('Model_name')
• finalize_model(my_model)
• predict_model(my_model)
• save_model(my_model)
• deploy_model(model)
```

- Finalize: This function trains a given estimator on the entire dataset including the holdout set
- predict: This function makes predictions on the test data set.
- Save: This function saves the transformation pipeline and trained model object into the current working directory as a pickle file for later use (load_model)
- Deploy: This function deploys the transformation pipeline and trained model on cloud.

Pedram Jahangiry

Workflow

- PyCaret offers both supervised and unsupervised workflow

Classification

```
# load dataset
import pandas as pd
train = pd.read_csv('train.csv')
test  = pd.read_csv('test.csv')

# init setup
from pycaret.classification import *
s = setup(train, target= 'target')

# model training and selection
best = compare_models()

# analyze best model
evaluate_model(best)


# predict on new data
predictions = predict_model(best, data =test )

# save best pipeline
save_model(best, 'my_best_pipeline')
```

Regression

```
# load dataset
import pandas as pd
train = pd.read_csv('train.csv')
test  = pd.read_csv('test.csv')

# init setup
from pycaret.regression import *
s = setup(train, target= 'target')

# model training and selection
best = compare_models()

# analyze best model
evaluate_model(best)


# predict on new data
predictions = predict_model(best, data =test )

# save best pipeline
save_model(best, 'my_best_pipeline')
```

Pedram Jahangiry

Workflow

- PyCaret offers both supervised and unsupervised workflow

Clustering

```
# load dataset
import pandas as pd
data = pd.read_csv('data.csv')

# init setup
from pycaret.clustering import *
s = setup(data, normalize= True)

# train k-means model
kmeans = create_model('kmeans')

# assign cluster labels on training data
kmeans_results = assign_model(kmeans)

# assign cluster labels on new data
new_data = pd.read_csv('new_data.csv')
predictions = predict_model(kmeans, data= new_data)

# save kmeans pipeline
save_model(kmeans, 'kmeans_pipeline')
```

Anomaly detection

```
# load dataset
import pandas as pd
data = pd.read_csv('data.csv')

# init setup
from pycaret.anomaly import *
s = setup(data, normalize= True)

# train isolation forest model
iforest = create_model('iforest')

# assign anomaly labels on training data
iforest_results = assign_model(iforest)

# assign anomaly labels on new data
new_data = pd.read_csv('new_data.csv')
predictions = predict_model(iforest, data= new_data)

# save iforest pipeline
save_model(iforest, 'iforest_pipeline')
```

Installation

- The most efficient way of installing PyCaret is through a virtual environment! Here are the steps:

1. Install anaconda https://www.anaconda.com/products/distribution
2. Create a conda environment: `conda create --name yourenvname python=3.8`
3. Activate conda environment: `conda activate yourenvname`
4. Install pycaret 3.0: `pip install pycaret[full]`
5. Create notebook kernel:

```
python -m ipykernel install --user --name yourenvname --display-name "display-name“
```

Pedram Jahangiry

Important Links

Tutorials	New to PyCaret? Checkout our official notebooks!
Example Notebooks	Example notebooks created by community.
Official Blog	Tutorials and articles by contributors.
Documentation	The detailed API docs of PyCaret
Video Tutorials	Our video tutorial from various events.
Cheat sheet	Cheat sheet for all functions across modules.
Discussions	Have questions? Engage with community and contributors.
Changelog	Changes and version history.
Roadmap	PyCaret's software and community development plan.

Pedram Jahangiry

PyCaret Time Series Module

PyCaret new time series module is now available with the main pycaret installation. Staying true to simplicity of PyCaret, it is consistent with the existing API and fully loaded with functionalities

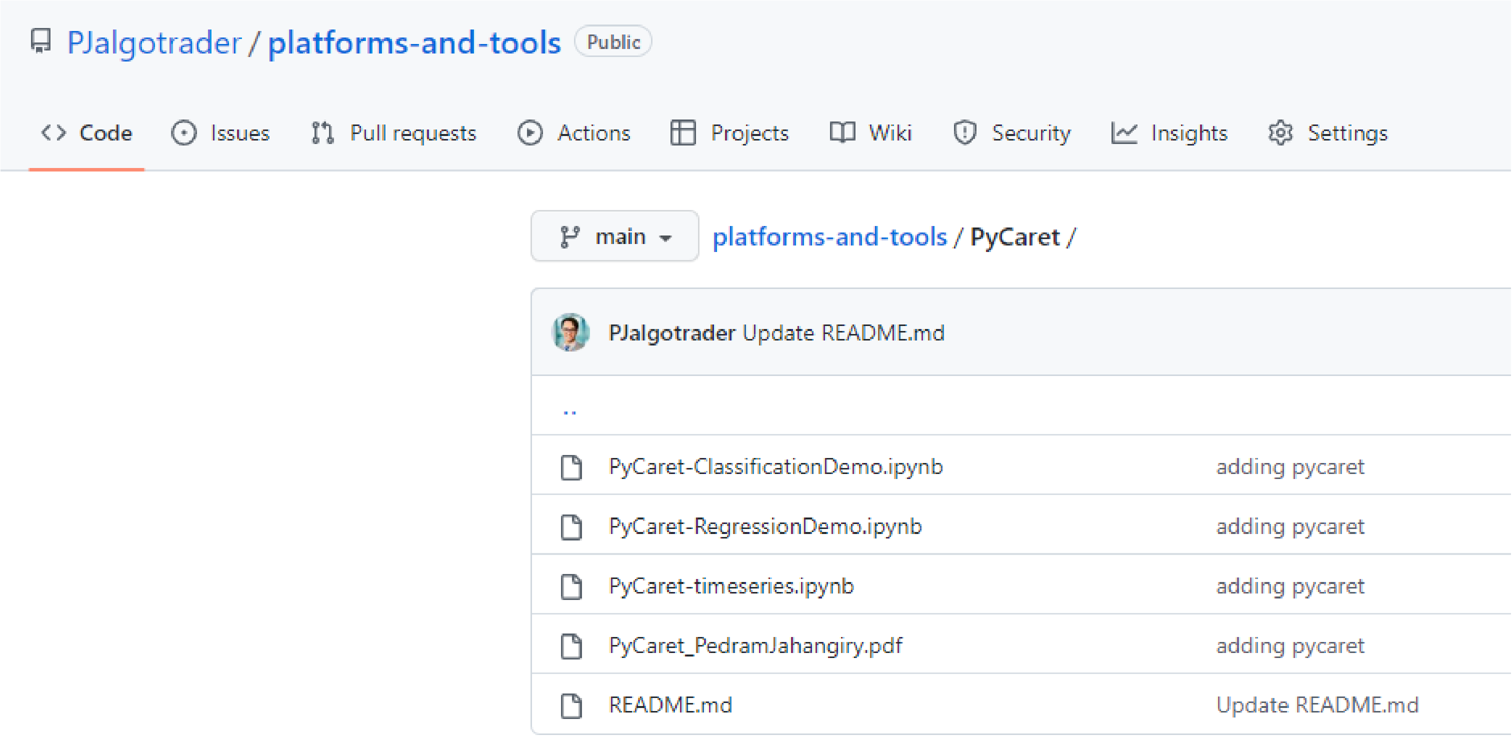
Time Series Quickstart	Get started with Time Series Analysis
Time Series Notebooks	New to Time Series? Checkout our official (detailed) notebooks!
Time Series Video Tutorials	Our video tutorial from various events.
Time Series FAQs	Have questions? Queck out the FAQ's
Time Series API Interface	The detailed API interface for the Time Series Module
Time Series Features and Roadmap	PyCaret's software and community development plan.

Pedram Jahangiry

Practical example in Python

Now let’s look at some practical examples in Python!

https://github.com/PJalgotrader/platforms-and-tools/tree/main/PyCaret



Pedram Jahangiry