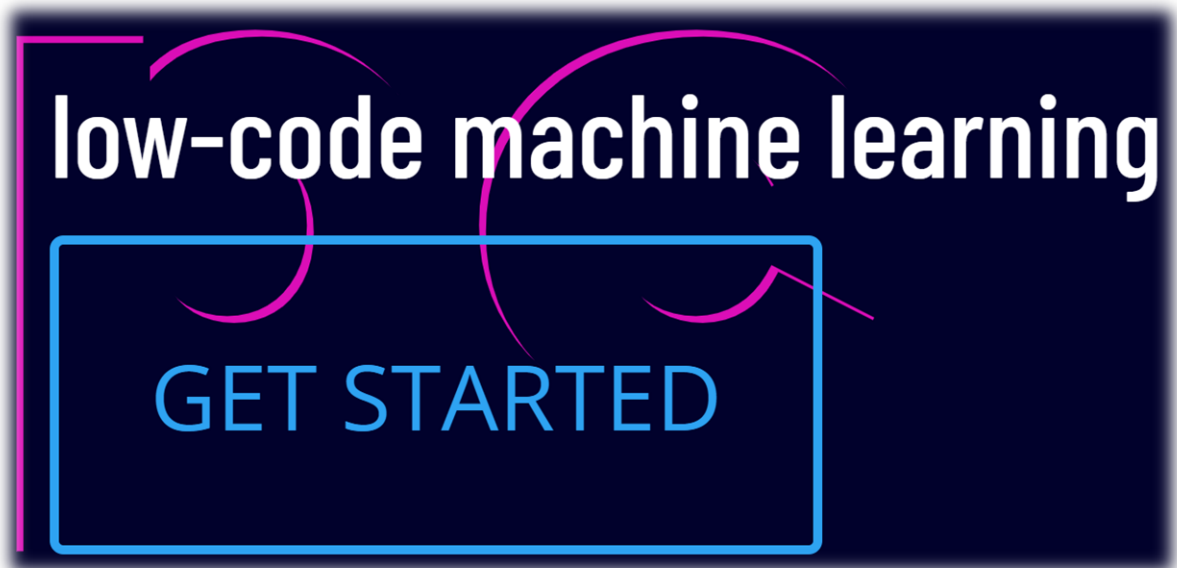
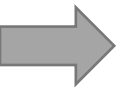




Introduction and installation





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



➔ 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.



Data
Preparation



Model
Training



Hyperparameter
Tuning



Analysis &
Interpretability



Model
Selection



Experiment
Logging



➔ Preprocessing (setup)

Data Preparation	Scale and Transform	Feature Engineering	Feature Selection
<ul style="list-style-type: none">• Missing values• Data Types• One-Hot Encoding• Ordinal Encoding• Cardinal Encoding• Handle Unknown Levels• Target Imbalance• Remove outliers	<ul style="list-style-type: none">• Normalize• Feature Transform• Target Transform	<ul style="list-style-type: none">• Feature interaction• Polynomial Features• Group Features• Bin Numeric Features• Combine Rare Levels• Create Clusters	<ul style="list-style-type: none">• Feature Selection• Remove Multicollinearity• Principal Component Analysis• Ignore Low Variance



Data
Preparation





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, ...)`



Model
Training



Hyperparameter
Tuning

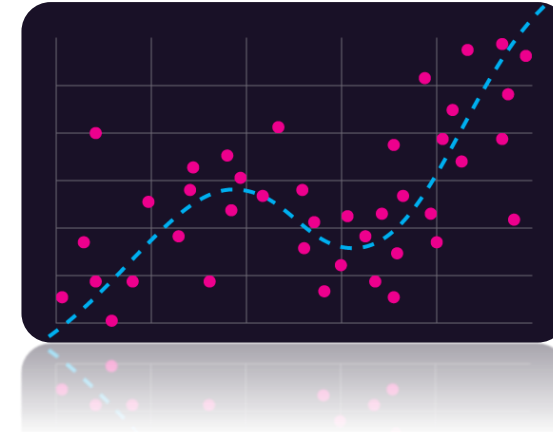


Model
Selection



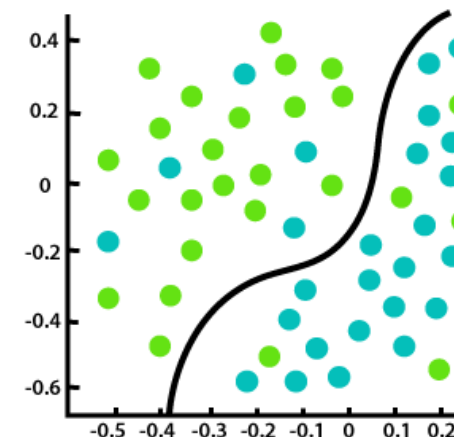
➔ List of models (Regression)

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)

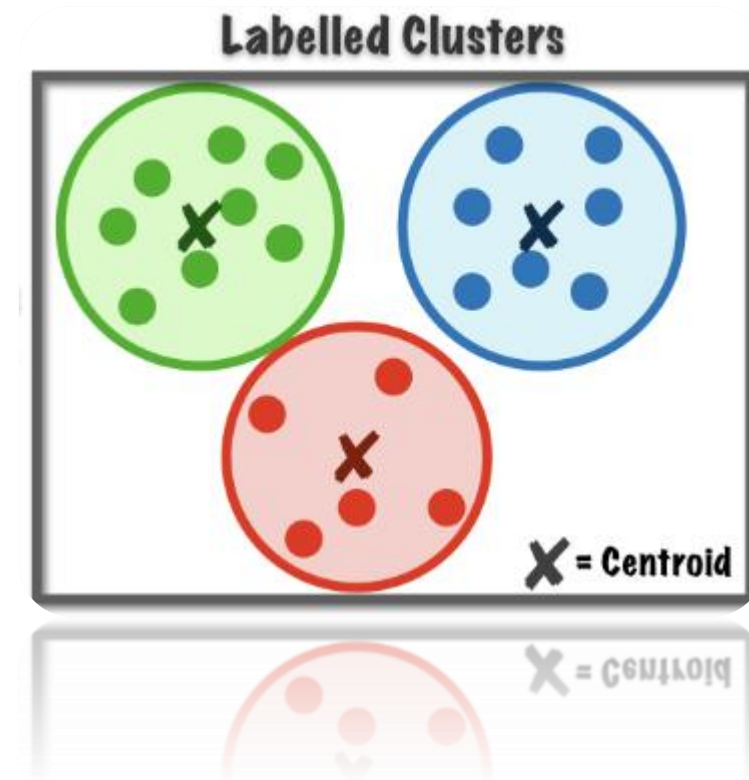
ID	Name
'lr'	Logistic Regression
'knn'	K Nearest Neighbour
'nb'	Naives Bayes
'dt'	Decision Tree Classifier
'svm'	SVM - Linear Kernel
'rbfsvm'	SVM - Radial Kernel
'gpc'	Gaussian Process Classifier
'mlp'	Multi Level Perceptron
'ridge'	Ridge Classifier
'rf'	Random Forest Classifier
'qda'	Quadratic Discriminant 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

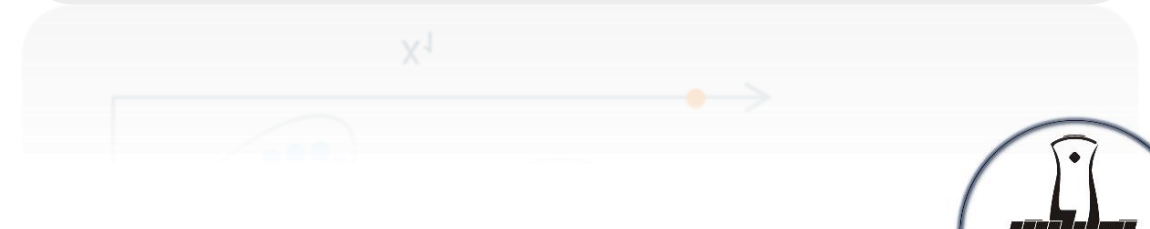
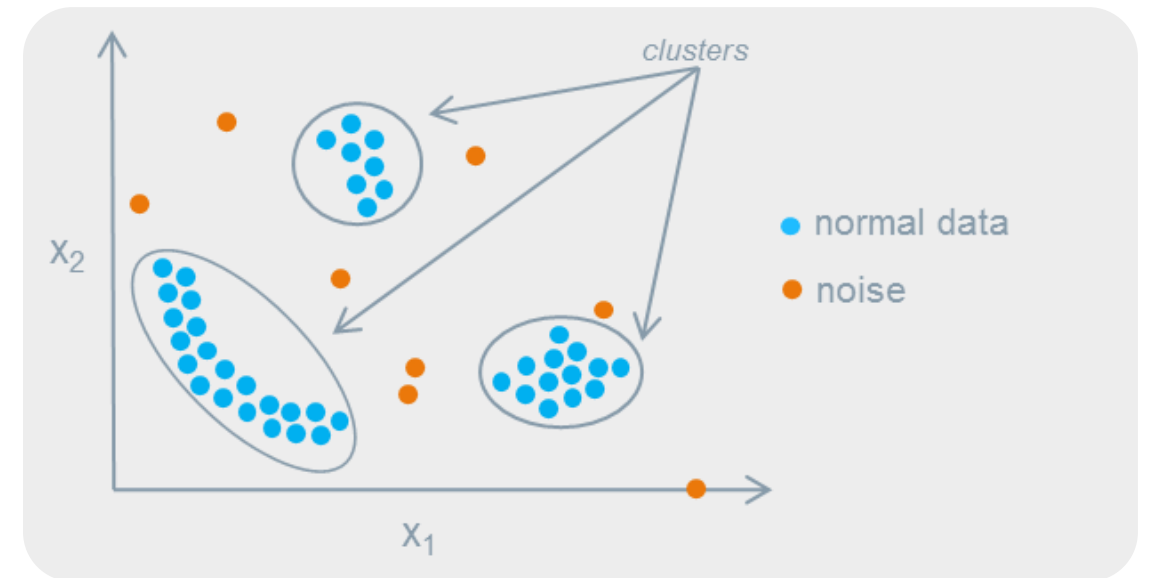
'kmodes'	K-Modes Clustering
'birch'	Birch Clustering



➔ 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

'sos'	stochastic outlier selection
'sod'	subspace outlier detection
'mcd'	minimum covariance determinant



→ Analysis and interpretability



Analysis &
Interpretability

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

- `plot_model(my_model)`
- `interpret_model(model)`

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'

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



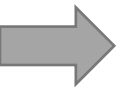
➔ Finalize, Predict, Save and Deploy model

```
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**.





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 --pre pycaret[full]`
5. Create notebook kernel:

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

On Google Colab, as of today (Dec, 2022):

```
!pip install --pre pycaret[full]
```





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')
```

```
2946~woq6j(p62f' ,wλ~p62f~b1bej1u6,)
# 2946 p62f b1bej1u6
```

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')
```

```
2946~woq6j(p62f' ,wλ~p62f~b1bej1u6,)
# 2946 p62f b1bej1u6
```





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')
```

```
2946~woq6J(kmeans, 'kmeans_pipeline')
# 2946 kmeans b1b6j1u6
```

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')

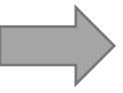
# assign anomaly labels on training data
iforest_results = assign_model(iforest)

# 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')
```

```
2946~woq6J(iforest, 'iforest_pipeline')
# 2946 iforest b1b6j1u6
```





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.



→ 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](#)

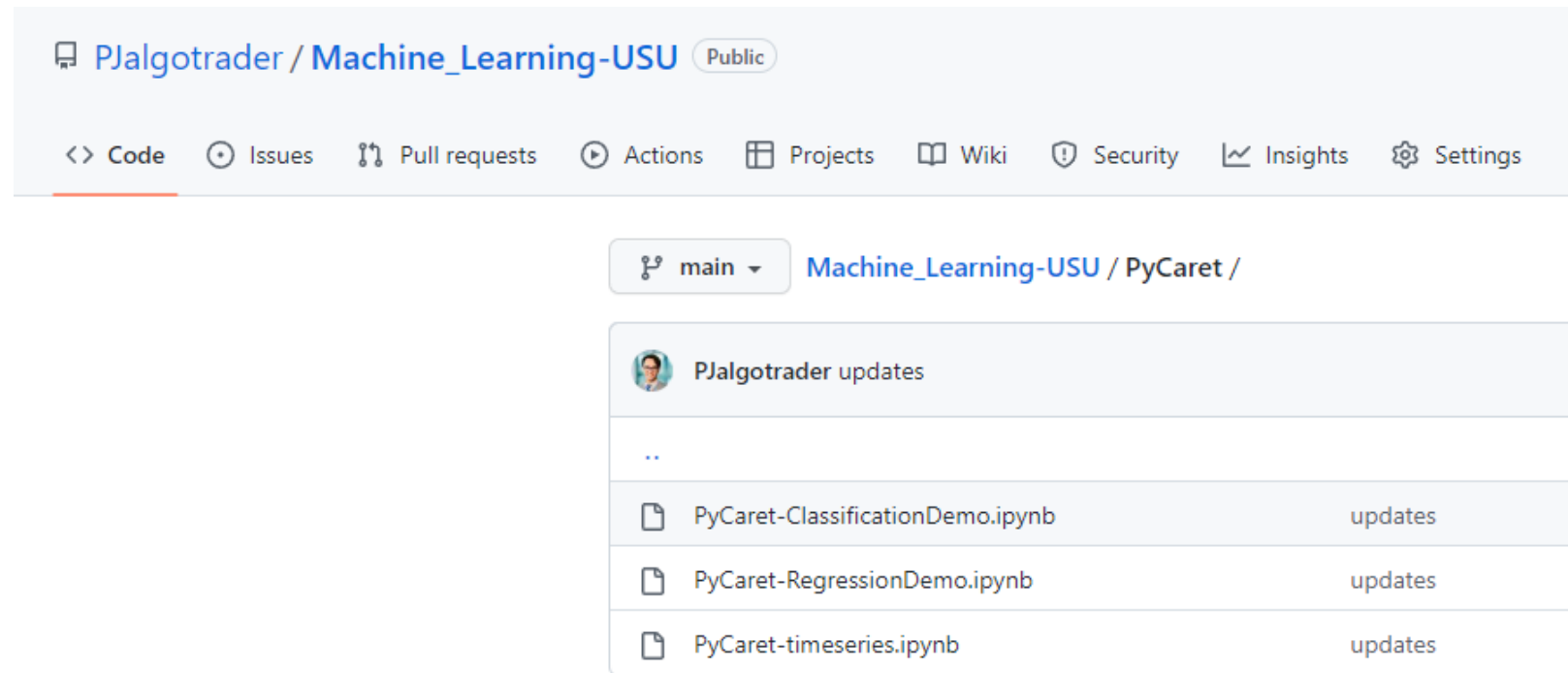
PyCaret's software and community development plan.



➔ Practical example in Python

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

https://github.com/PJalgotrader/Machine_Learning-USU/tree/main/PyCaret



The screenshot shows the GitHub repository page for PJalgotrader / Machine_Learning-USU. The repository is public. The navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The current view is the main branch, showing the directory structure. The PyCaret directory is highlighted, containing three files: PyCaret-ClassificationDemo.ipynb, PyCaret-RegressionDemo.ipynb, and PyCaret-timeseries.ipynb, all with 'updates' status.

PJalgotrader / Machine_Learning-USU Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main Machine_Learning-USU / PyCaret /

PJalgotrader updates

..

PyCaret-ClassificationDemo.ipynb	updates
PyCaret-RegressionDemo.ipynb	updates
PyCaret-timeseries.ipynb	updates

