PYCARET

Introduction

Machine learning has become a fundamental component of various industries, enabling businesses to derive insights, automate tasks, and make data-driven decisions. However, building effective machine learning models often involves complex coding and numerous trial-and-error iterations. To simplify this process and accelerate model development, PyCaret emerges as a robust machine learning library that empowers users to build, evaluate, and deploy machine learning models with ease. In this article, we will explore the key features of PyCaret and provide an example code to showcase its capabilities.

What is PyCaret?

PyCaret is an open-source, low-code machine learning library designed to simplify the end-to-end machine learning workflow. It offers a diverse range of functionalities, including data preprocessing, feature selection, model training, hyperparameter tuning, and ensemble modeling. PyCaret supports both supervised and unsupervised learning tasks, making it a versatile tool for various machine learning projects.

Key Features of PyCaret

1. Streamlined Workflow: PyCaret provides a unified and intuitive interface to perform various machine learning tasks. With just a few lines of code, users can preprocess data, compare multiple models, fine-tune hyperparameters, and evaluate model performance.

2. Automated Preprocessing: Data preprocessing is a critical step in machine learning, involving tasks such as missing value imputation, feature scaling, and categorical variable encoding. PyCaret automates these preprocessing tasks, saving valuable time and effort for the user.

3. Model Selection and Training: PyCaret offers a comprehensive suite of pre-built machine learning models, ranging from traditional algorithms like linear regression and support vector machines to ensemble methods like random forest and gradient boosting. Users can easily compare multiple models and select the best-performing one based on various evaluation metrics.

4. Hyperparameter Tuning: Fine-tuning model hyperparameters is essential for achieving optimal performance. PyCaret integrates popular hyperparameter tuning techniques, such as grid search and random search, to automatically search for the best hyperparameter combinations.

5. Model Stacking and Blending: PyCaret facilitates model stacking and blending, enabling users to combine multiple models to create more robust and accurate predictions. This technique leverages the strengths of individual models and enhances overall predictive power.

1.Installing Pycaret

```
pip install pycaret
```

2.Import necessary libraries

```
import pandas as pd
from pycaret.regression import *
```

Example Code: Startup Profit prediction with PyCaret

3.Importing 50 Startup dataset

```
data = pd.read_csv('50_Startups.csv')
data.head()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

4.Initialize the setup

- The `setup()` function automatically preprocesses the data and splits it into training and testing sets.

```
regression = setup(data, target='Profit', session_id=123)
regression
```

	Description	Value
0	Session id	123
1	Target	Profit
2	Target type	Regression
3	Original data shape	(50, 5)
4	Transformed data shape	(50, 7)
5	Transformed train set shape	(35, 7)
6	Transformed test set shape	(15, 7)
7	Numeric features	3
8	Categorical features	1
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Fold Generator	KFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	reg-default-name
21	USI	a5fa

```
<pycaret.regression.oop.RegressionExperiment at 0x1dd3f03dc90>
```

5.Compare and evaluate the different models

- Next, we use the `compare_models()` function to compare and evaluate different regression models using default settings.
- PyCaret automatically evaluates models using various metrics, such as R-squared, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).

[5]:
```
best_model = compare_models()
best_model
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	8439.4910	151995494.0469	10096.8588	0.8889	0.1706	0.1561	0.0450
xgboost	Extreme Gradient Boosting	7917.4076	118977639.2000	10203.3711	0.8580	0.1829	0.1224	0.0990
br	Bayesian Ridge	7759.7853	117731002.5676	9473.9148	0.8562	0.1525	0.1384	0.0200
en	Elastic Net	7873.8324	118420853.3844	9537.6059	0.8465	0.1529	0.1389	0.0170
ridge	Ridge Regression	8100.0242	122473530.3254	9725.4221	0.8288	0.1535	0.1406	0.0180
lr	Linear Regression	8152.8553	123518717.5635	9771.9435	0.8245	0.1537	0.1410	0.7230
lasso	Lasso Regression	8151.8396	123507256.6629	9771.4025	0.8245	0.1537	0.1410	0.0170
lar	Least Angle Regression	8152.9283	123519790.9567	9771.9813	0.8245	0.1537	0.1410	0.0180
llar	Lasso Least Angle Regression	8151.8396	123507256.6023	9771.4025	0.8245	0.1537	0.1410	0.0170
rf	Random Forest Regressor	9202.3593	163149994.9410	10881.6519	0.8107	0.1824	0.1712	0.0560
gbr	Gradient Boosting Regressor	9240.8438	174505103.5238	11643.4834	0.7818	0.1914	0.1790	0.0350
ada	AdaBoost Regressor	10149.8483	210450730.0702	12763.0204	0.7789	0.2062	0.1919	0.0400
dt	Decision Tree Regressor	9558.0188	165921920.2526	12286.8804	0.7459	0.1809	0.1388	0.0170
knn	K Neighbors Regressor	14566.4761	391132018.4000	16422.3665	0.6241	0.2061	0.2142	0.0200
huber	Huber Regressor	14314.6314	263368335.3002	15665.3034	0.4497	0.2361	0.1933	0.0230
par	Passive Aggressive Regressor	18606.0220	533748877.8664	20619.1582	0.2323	0.2814	0.2415	0.0170
omp	Orthogonal Matching Pursuit	21475.8208	708725968.2359	23723.5900	-0.1109	0.2820	0.2926	0.0160
lightgbm	Light Gradient Boosting Machine	29255.4787	1506939961.7722	34890.3672	-0.3835	0.3813	0.4375	0.0380
dummy	Dummy Regressor	29255.4801	1506940000.0000	34890.3670	-0.3835	0.3813	0.4375	0.0200

[5]:
```
ExtraTreesRegressor
ExtraTreesRegressor(n_jobs=-1, random_state=123)
```

6.Tune Hyperparameters the best model

- After identifying the best model, we fine-tune its hyperparameters using the `tune_model()` function.
- PyCaret performs an automatic hyperparameter search to find the optimal combination.

[6]:
```
tuned_model = tune_model(best_model)
tuned_model
```

2	14260.7373	481324328.6195	21939.1050	0.7453	0.3959	0.3281
3	23583.0006	785960623.7273	28034.9893	0.6410	0.1992	0.1737
4	6518.6413	63461869.6309	7966.2959	0.9057	0.0704	0.0592
5	5083.5500	32385171.6163	5690.7971	0.7305	0.0626	0.0560
6	5795.0448	45830140.0508	6769.7962	0.9256	0.0594	0.0508
7	13838.1259	212391019.3541	14573.6413	0.5548	0.1381	0.1341
8	702.1902	1231349.8169	1109.6620	0.9963	0.0135	0.0084
9	19946.0405	841606501.3523	29010.4550	0.6980	0.1900	0.1384
Mean	13380.9984	450972786.0000	16423.6858	0.7647	0.2034	0.2244
Std	11646.2768	607636496.4080	13462.3672	0.1531	0.2476	0.3595

7.Evaluate the model performance ¶

- We then evaluate the performance of the tuned model using the `evaluate_model()` function.
- PyCaret generates a comprehensive report with various evaluation metrics and visualizations.

[7]:
```
evaluate_model(tuned_model)
```

Click to show javascript error.

8.Make predictions on new data

- To make predictions on new data, we load the new dataset, `new_data.csv`, and use the `predict_model()` function with the tuned model.

[8]:
```
new_data = pd.read_csv('1000_Companies.csv')
predictions = predict_model(tuned_model, data=new_data)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Extra Trees Regressor	5305.4327	243313798.0882	15598.5191	0.8676	0.0839	0.0452

9.Save the model for future use

- Finally, we save the tuned model using the `save_model()` function for future use.

[9]:
```
save_model(tuned_model, 'Profit_prediction_model')
```

```
Transformation Pipeline and Model Successfully Saved
```

[9]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(include=['R&D Spend', 'Administration',
                                              'Marketing Spend'],
                                     transformer=SimpleImputer())),
                 ('categorical_imputer',
                  TransformerWrapper(include=['State'],
                                     transformer=SimpleImputer(strategy='most_frequent'))),
                 ('onehot_encoding',
                  TransformerWrapper(include=['State'],
                                     transformer=OneHotEncoder(cols=['State'],
                                                               handle_missing='return_nan',
                                                               use_cat_names=True))),
                 ('clean_column_names',
                  TransformerWrapper(transformer=CleanColumnNames())),
                 ('trained_model',
                  ExtraTreesRegressor(n_jobs=-1, random_state=123))]),
 'Profit_prediction_model.pkl')
```

Conclusion

PyCaret offers an accessible and efficient approach to utilizing machine learning for a wide range of tasks. With its extensive feature set and streamlined workflow, PyCaret enables users to quickly prototype and deploy machine learning models. Whether you are new to the field or an experienced data scientist, PyCaret can greatly simplify the machine learning process, allowing you to concentrate on extracting valuable insights from your data.