In [5]:
```
#loading data from pycaret
from pycaret.datasets import get_data
```

In [8]:
```
# classificatin example
diabetes_df = get_data("diabetes")
```

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [9]:
```
from pycaret.classification import *
clf = setup(data = diabetes_df, target= 'Class variable')
```

	Description	Value
0	Session id	8224
1	Target	Class variable
2	Target type	Binary
3	Original data shape	(768, 9)
4	Transformed data shape	(768, 9)
5	Transformed train set shape	(537, 9)
6	Transformed test set shape	(231, 9)
7	Numeric features	8
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	constant
12	Low variance threshold	0
13	Fold Generator	StratifiedKFold
14	Fold Number	10
15	CPU Jobs	-1
16	Use GPU	False
17	Log Experiment	False
18	Experiment Name	clf-default-name
19	USI	2ea2

In [11]:
```
#compare models
compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.7672	0.8402	0.6526	0.6762	0.6621	0.4851	0.4869	0.0420
rf	Random Forest Classifier	0.7652	0.8413	0.5974	0.6907	0.6375	0.4656	0.4704	0.1370
ridge	Ridge Classifier	0.7635	0.0000	0.5509	0.7084	0.6163	0.4502	0.4597	0.0600
gbc	Gradient Boosting Classifier	0.7631	0.8391	0.6289	0.6837	0.6516	0.4730	0.4767	0.0740
lda	Linear Discriminant Analysis	0.7616	0.8376	0.5561	0.7038	0.6176	0.4486	0.4575	0.0270
lr	Logistic Regression	0.7597	0.8406	0.5614	0.6967	0.6191	0.4469	0.4543	0.0980
nb	Naive Bayes	0.7580	0.8175	0.6149	0.6702	0.6387	0.4577	0.4608	0.0500
et	Extra Trees Classifier	0.7523	0.8257	0.5769	0.6771	0.6204	0.4380	0.4433	0.1300
knn	K Neighbors Classifier	0.7412	0.7730	0.5827	0.6611	0.6105	0.4192	0.4273	0.0700
qda	Quadratic Discriminant Analysis	0.7412	0.8116	0.5670	0.6523	0.6031	0.4132	0.4179	0.0250
ada	Ada Boost Classifier	0.7410	0.7917	0.5553	0.6660	0.5983	0.4101	0.4191	0.1850
dt	Decision Tree Classifier	0.7245	0.6972	0.6088	0.6084	0.6054	0.3942	0.3968	0.0750
dummy	Dummy Classifier	0.6518	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0270
svm	SVM - Linear Kernel	0.5384	0.0000	0.5272	0.4271	0.3958	0.0703	0.0960	0.0420

```
Processing:   0%|          | 0/61 [00:00<?, ?it/s]
```

Out[11]:
```
*                    LGBMClassifier
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
               random_state=8224, reg_alpha=0.0, reg_lambda=0.0, silent='warn',
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

Regression problem

In [12]:
```
boston_df=get_data('boston')
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [13]:
```
from pycaret.regression import *
reg= setup(data= boston_df, target='medv')
```

	Description	Value
0	Session id	440
1	Target	medv
2	Target type	Regression
3	Data shape	(506, 14)
4	Train data shape	(354, 14)
5	Test data shape	(152, 14)
6	Numeric features	13
7	Preprocess	True
8	Imputation type	simple
9	Numeric imputation	mean
10	Categorical imputation	constant
11	Low variance threshold	0
12	Fold Generator	KFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	reg-default-name
18	USI	a017

In [15]:
```
#compare models
compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	2.0921	8.9537	2.9637	0.8571	0.1422	0.1090	0.1030
rf	Random Forest Regressor	2.2145	10.6581	3.1857	0.8345	0.1433	0.1127	0.2700
et	Extra Trees Regressor	2.1684	11.1270	3.2327	0.8286	0.1407	0.1064	0.1580
lightgbm	Light Gradient Boosting Machine	2.2631	12.1780	3.3714	0.8217	0.1569	0.1183	0.3300
ada	AdaBoost Regressor	2.5752	13.6054	3.6114	0.7941	0.1702	0.1355	0.1170
dt	Decision Tree Regressor	2.9056	17.3566	4.0777	0.7288	0.1913	0.1519	0.0390
lr	Linear Regression	3.1709	21.5196	4.5975	0.6545	0.2374	0.1613	1.7480
ridge	Ridge Regression	3.1631	21.7097	4.6116	0.6524	0.2479	0.1622	0.0510
br	Bayesian Ridge	3.2065	22.2917	4.6678	0.6481	0.2585	0.1646	0.0430
en	Elastic Net	3.4160	24.7668	4.8957	0.6345	0.2468	0.1673	0.0470
lasso	Lasso Regression	3.4705	25.7036	4.9796	0.6240	0.2476	0.1688	0.1240
lar	Least Angle Regression	3.7072	27.6928	5.1611	0.5338	0.2941	0.1880	0.0750
omp	Orthogonal Matching Pursuit	4.3270	35.6184	5.9650	0.4804	0.2899	0.2063	0.0760
huber	Huber Regressor	4.2284	38.1467	6.0104	0.4396	0.2809	0.2060	0.0780
knn	K Neighbors Regressor	4.3458	40.5181	6.2801	0.3806	0.2471	0.2048	0.0540
llar	Lasso Least Angle Regression	6.2399	73.8546	8.4672	-0.0507	0.3633	0.3310	0.0550
dummy	Dummy Regressor	6.2399	73.8546	8.4672	-0.0507	0.3633	0.3310	0.0440
par	Passive Aggressive Regressor	6.8750	79.1292	8.8285	-0.2819	0.3930	0.3782	0.0370

```
Processing:   0%|          | 0/77 [00:00<?, ?it/s]
```

Out[15]:
```
*                    GradientBoostingRegressor
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='squared_error',
                          max_depth=3, max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_samples_leaf=1,
                          min_samples_split=2, min_weight_fraction_leaf=0.0,
                          n_estimators=100, n_iter_no_change=None,
                          random_state=440, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

In [21]:
```
reg_model = create_model('gbr')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	2.2653	11.0031	3.3171	0.8915	0.1600	0.1249
1	1.8407	7.3078	2.7033	0.9077	0.1111	0.0794
2	2.2816	10.9389	3.3074	0.8169	0.1728	0.1348
3	1.8224	5.7186	2.3914	0.8912	0.1205	0.0952
4	2.0376	6.6243	2.5738	0.9170	0.1470	0.1201
5	2.4883	13.0829	3.6170	0.7858	0.1601	0.1232
6	2.1320	7.0456	2.6544	0.9291	0.1423	0.1186
7	2.2063	10.0294	3.1669	0.8937	0.1349	0.0881
8	1.8464	11.3553	3.3698	0.7965	0.1368	0.0947
9	2.0000	6.4312	2.5360	0.7419	0.1370	0.1113
Mean	2.0921	8.9537	2.9637	0.8571	0.1422	0.1090
Std	0.2110	2.4642	0.4126	0.0622	0.0177	0.0175

```
Processing:   0%|          | 0/4 [00:00<?, ?it/s]
```

In [22]:
```
reg_model
```

Out[22]:
```
*                    GradientBoostingRegressor
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='squared_error',
                          max_depth=3, max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_samples_leaf=1,
                          min_samples_split=2, min_weight_fraction_leaf=0.0,
                          n_estimators=100, n_iter_no_change=None,
                          random_state=440, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

In [24]:
```
reg_model.get_params
```

Out[24]:
```
<bound method BaseEstimator.get_params of GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='squared_error',
                          max_depth=3, max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_samples_leaf=1,
                          min_samples_split=2, min_weight_fraction_leaf=0.0,
                          n_estimators=100, n_iter_no_change=None,
                          random_state=440, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)>
```

In [30]:
```
# hyper tuning of models
#reg_model_hypertune = tune_model('gbr', n_iter=50, optimize='mae')
tuned_reg = tune_model(reg_model)
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	2.6335	16.1611	4.0201	0.8406	0.1757	0.1366
1	1.9413	8.5036	2.9161	0.8926	0.1224	0.0827
2	2.5453	13.0340	3.6103	0.7819	0.1814	0.1460
3	1.7058	4.2981	2.0703	0.9184	0.1037	0.0906
4	2.1579	9.4586	3.0755	0.8815	0.1371	0.1146
5	2.5404	16.2696	4.0336	0.7336	0.1679	0.1260
6	1.7672	5.1152	2.2617	0.9485	0.1239	0.1005
7	2.0810	8.6031	2.9331	0.9088	0.1140	0.0781
8	1.7881	8.1714	2.8586	0.8536	0.1419	0.1004
9	2.1762	7.9896	2.8266	0.6794	0.1628	0.1217
Mean	2.1337	9.7592	3.0606	0.8429	0.1431	0.1097
Std	0.3281	3.9277	0.6262	0.0823	0.0260	0.0218

```
Processing:   0%|          | 0/7 [00:00<?, ?it/s]
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Original model was better than the tuned model, hence it will be returned. NOTE: The display metrics are for the tuned model (not the original one).
```

clustering

In [31]:
```
jewellery = get_data('jewellery')
```

	Age	Income	SpendingScore	Savings
0	58	77769	0.791329	6559.829923
1	59	81799	0.791082	5417.661426
2	62	74751	0.702657	9258.992965
3	59	74373	0.765680	7346.334504
4	87	17760	0.348778	16869.507130

In [33]:
```
from pycaret.clustering import *
clust_algo = setup(jewellery)
```

	Description	Value
0	Session id	1544
1	Original data shape	(505, 4)
2	Transformed data shape	(505, 4)
3	Numeric features	4
4	Preprocess	True
5	Imputation type	simple
6	Numeric imputation	mean
7	Categorical imputation	constant
8	Low variance threshold	0
9	CPU Jobs	-1
10	Use GPU	False
11	Log Experiment	False
12	Experiment Name	cluster-default-name
13	USI	4f29

In [ ]:
```

```