```
!pip install pycaret
```

```
import pandas as pd
import numpy as np
# Importing dataset
from pycaret.datasets import get_data
diabetes = 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 |

CLASSIFICATION MODEL COMPARISON

- AUC: Area Under the Curve
- Accuracy: (TP + TN)/Total
- Precision: TP/(TP + FP)
- Recall: TP/(TP + FN)
- F1: 2 x [(Precision x Recall)/(Precision + Recall)]
- Cohen's Kappa: The higher the kappa value, the stronger the degree of agreement. When: Kappa = 1, perfect agreement exists. Kappa < 0, agreement is weaker than expected by chance; this rarely happens. Kappa close to 0, the degree of agreement is the same as would be expected by chance.
- MCC: The Matthews correlation coefficient (MCC) is a more reliable statistical rate which produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements in the dataset.

```
# Importing module and initializing setup
from pycaret.classification import *
clf1 = setup(data = diabetes, target = 'Class variable')
# return best model
best = compare_models()
# return top 3 models based on 'Accuracy'
top3·=·compare_models(n_select·=·3)
#·return·best·model·based·on·AUC
best·=·compare_models(sort·=·'AUC')·#default·is·'Accuracy'
#·compare·specific·models
best_specific·=·compare_models(include·=·['dt','rf','lda'])
#·blacklist·certain·models
best_specific·=·compare_models(exclude·=·['svm'])
```

| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| **lr** | Logistic Regression | 0.7673 | 0.8258 | 0.5912 | 0.6942 | 0.6324 | 0.4652 | 0.4724 | 0.250 |
| **ridge** | Ridge Classifier | 0.7653 | 0.0000 | 0.5746 | 0.7028 | 0.6229 | 0.4570 | 0.4678 | 0.017 |
| **rf** | Random Forest Classifier | 0.7599 | 0.8218 | 0.5538 | 0.6940 | 0.6127 | 0.4427 | 0.4505 | 0.519 |
| **lda** | Linear Discriminant Analysis | 0.7598 | 0.8200 | 0.5640 | 0.6954 | 0.6140 | 0.4441 | 0.4549 | 0.018 |
| **knn** | K Neighbors Classifier | 0.7504 | 0.7742 | 0.5810 | 0.6687 | 0.6154 | 0.4331 | 0.4397 | 0.118 |
| **gbc** | Gradient Boosting Classifier | 0.7487 | 0.8345 | 0.5848 | 0.6589 | 0.6151 | 0.4301 | 0.4350 | 0.135 |
| **ada** | Ada Boost Classifier | 0.7392 | 0.7838 | 0.5640 | 0.6454 | 0.5953 | 0.4056 | 0.4119 | 0.112 |
| **lightgbm** | Light Gradient Boosting Machine | 0.7375 | 0.7973 | 0.5918 | 0.6329 | 0.6084 | 0.4121 | 0.4149 | 0.048 |
| **et** | Extra Trees Classifier | 0.7301 | 0.7895 | 0.5105 | 0.6508 | 0.5684 | 0.3761 | 0.3846 | 0.516 |
| **dt** | Decision Tree Classifier | 0.7093 | 0.6726 | 0.5535 | 0.5880 | 0.5672 | 0.3495 | 0.3518 | 0.019 |
| **nb** | Naive Bayes | 0.6760 | 0.7406 | 0.2102 | 0.6320 | 0.3056 | 0.1572 | 0.2036 | 0.018 |
| **dummy** | Dummy Classifier | 0.6537 | 0.5000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.015 |
| **qda** | Quadratic Discriminant Analysis | 0.6090 | 0.6080 | 0.4211 | 0.3904 | 0.3755 | 0.1221 | 0.1317 | 0.020 |

Double-click (or enter) to edit

REGRESSION MODEL COMPARISON

- MAE: Mean Absolute Error
- MSE: Mean Squared Error
- RMSE: Root Mean Squared Error
- RMSLE: Root Mean Squared Logarithmic Error
- MAPE: Mean Absolute Percentage Error

```
# Importing dataset
from pycaret.datasets import get_data
boston = get_data('boston')
# Importing module and initializing setup
from pycaret.regression import *
reg1 = setup(data = boston, target = 'medv')
# return best model
best = compare_models()
# return top 3 models based on 'R2'
top3 = compare_models(n_select = 3)
# return best model based on MAPE
best = compare_models(sort = 'MAPE') #default is 'R2'
# compare specific models
best_specific = compare_models(include = ['dt','rf','br'])
```

```
15 # blacklist certain models
```

| | Model | MAE | MSE | RMSE | R2 | RMSLE | MAPE | TT (Sec) |
|---|---|---|---|---|---|---|---|---|
| **gbr** | Gradient Boosting Regressor | 2.0899 | 9.3512 | 2.9591 | 0.8761 | 0.1325 | 0.1059 | 0.100 |
| **rf** | Random Forest Regressor | 2.2396 | 11.2100 | 3.2395 | 0.8547 | 0.1428 | 0.1127 | 0.510 |
| **lightgbm** | Light Gradient Boosting Machine | 2.3578 | 12.0260 | 3.3609 | 0.8508 | 0.1491 | 0.1184 | 0.042 |
| **et** | Extra Trees Regressor | 2.1286 | 10.9724 | 3.1659 | 0.8466 | 0.1423 | 0.1080 | 0.450 |
| **ada** | AdaBoost Regressor | 2.7942 | 14.6806 | 3.7168 | 0.8118 | 0.1746 | 0.1497 | 0.101 |
| **lr** | Linear Regression | 3.4004 | 23.6335 | 4.7592 | 0.7111 | 0.2151 | 0.1667 | 0.015 |
| **ridge** | Ridge Regression | 3.3753 | 23.8544 | 4.7807 | 0.7074 | 0.2189 | 0.1660 | 0.016 |
| **lar** | Least Angle Regression | 3.5243 | 24.9663 | 4.8628 | 0.6988 | 0.2223 | 0.1745 | 0.023 |
| **br** | Bayesian Ridge | 3.4019 | 24.5736 | 4.8526 | 0.6978 | 0.2249 | 0.1680 | 0.015 |
| **dt** | Decision Tree Regressor | 3.0904 | 24.2529 | 4.7051 | 0.6570 | 0.1943 | 0.1505 | 0.018 |
| **en** | Elastic Net | 3.7861 | 29.1991 | 5.2992 | 0.6447 | 0.2429 | 0.1807 | 0.016 |
| **lasso** | Lasso Regression | 3.7954 | 29.4488 | 5.3200 | 0.6424 | 0.2434 | 0.1809 | 0.017 |
| **huber** | Huber Regressor | 3.8376 | 32.6169 | 5.6180 | 0.6022 | 0.2529 | 0.1847 | 0.047 |
| **omp** | Orthogonal Matching Pursuit | 4.2474 | 34.0793 | 5.7519 | 0.5690 | 0.2966 | 0.2129 | 0.015 |
| **knn** | K Neighbors Regressor | 4.6165 | 42.6744 | 6.4303 | 0.4842 | 0.2546 | 0.2203 | 0.064 |
| **llar** | Lasso Least Angle Regression | 6.8222 | 87.3518 | 9.2249 | -0.0490 | 0.4004 | 0.3766 | 0.015 |
| **dummy** | Dummy Regressor | 6.8222 | 87.3518 | 9.2249 | -0.0490 | 0.4004 | 0.3766 | 0.011 |
| **par** | Passive Aggressive Regressor | 9.8828 | 148.7945 | 11.9817 | -1.1990 | 0.5874 | 0.5430 | 0.018 |

✓ 1m 34s completed at 11:32 AM