Pycaret is a low-code python library for ML

1️⃣ Imports

In [1]:
```
import pandas as pd
from pycaret.classification import *
```

2️⃣ Load your data (example)

In [3]:
```
data = pd.read_csv("Thyroid_Diff.csv")
```

3️⃣ Setup the PyCaret experiment

In [5]:
```
clf_setup = setup(
    data=data,
    target='Recurred',      # or your target column name
    session_id=123,
    fold=5
)
```

	Description	Value
0	Session id	123
1	Target	Recurred
2	Target type	Binary
3	Target mapping	No: 0, Yes: 1
4	Original data shape	(383, 17)
5	Transformed data shape	(383, 50)
6	Transformed train set shape	(268, 50)
7	Transformed test set shape	(115, 50)
8	Numeric features	1
9	Categorical features	15
10	Preprocess	True
11	Imputation type	simple
12	Numeric imputation	mean
13	Categorical imputation	mode
14	Maximum one-hot encoding	25
15	Encoding method	None
16	Fold Generator	StratifiedKFold
17	Fold Number	5
18	CPU Jobs	-1
19	Use GPU	False
20	Log Experiment	False
21	Experiment Name	clf-default-name
22	USI	bfa0

4️⃣ Find the best model and store it in best_model

In [21]:
```
from pycaret.classification import compare_models, tune_model, save_model

# 1) Find the best baseline model (already have this)
best_model = compare_models(sort='AUC')

# 2) Tune that best model, still optimizing for AUC
best_tuned_model = tune_model(best_model, optimize='AUC')
```

```
# 3) save the tuned best model as a .pkl file
save_model(best_tuned_model, 'best_tuned_thyroid_model')
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9626	0.9941	0.9626	0.9632	0.9622	0.9058	0.9074	0.3100
rf	Random Forest Classifier	0.9551	0.9907	0.9551	0.9552	0.9546	0.8868	0.8880	0.3360
et	Extra Trees Classifier	0.9551	0.9893	0.9551	0.9558	0.9546	0.8869	0.8889	0.3020
gbc	Gradient Boosting Classifier	0.9477	0.9889	0.9477	0.9481	0.9474	0.8698	0.8709	0.2560
lr	Logistic Regression	0.9476	0.9834	0.9476	0.9475	0.9465	0.8659	0.8681	2.3240
ridge	Ridge Classifier	0.9477	0.9751	0.9477	0.9484	0.9466	0.8660	0.8692	0.1880
svm	SVM - Linear Kernel	0.8769	0.9749	0.8769	0.9005	0.8743	0.6968	0.7252	0.1860
lda	Linear Discriminant Analysis	0.9477	0.9710	0.9477	0.9484	0.9466	0.8660	0.8692	0.1780
ada	Ada Boost Classifier	0.9440	0.9674	0.9440	0.9451	0.9436	0.8603	0.8626	0.2520
nb	Naive Bayes	0.9253	0.9402	0.9253	0.9279	0.9239	0.8105	0.8162	0.1840
qda	Quadratic Discriminant Analysis	0.8807	0.9307	0.8807	0.8862	0.8692	0.6645	0.6914	0.1800
dt	Decision Tree Classifier	0.9402	0.9260	0.9402	0.9425	0.9399	0.8511	0.8543	0.1900
knn	K Neighbors Classifier	0.8693	0.9095	0.8693	0.8769	0.8618	0.6502	0.6728	1.9740
dummy	Dummy Classifier	0.7164	0.5000	0.7164	0.5133	0.5981	0.0000	0.0000	0.1800

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.9444	0.9795	0.9444	0.9440	0.9438	0.8586	0.8596
2	0.9630	0.9967	0.9630	0.9630	0.9630	0.9112	0.9112
3	0.9434	0.9684	0.9434	0.9430	0.9428	0.8577	0.8586
4	0.9811	0.9956	0.9811	0.9816	0.9809	0.9526	0.9536
Mean	0.9664	0.9880	0.9664	0.9663	0.9661	0.9160	0.9166
Std	0.0218	0.0121	0.0218	0.0220	0.0220	0.0550	0.0547

```
Fitting 5 folds for each of 10 candidates, totalling 50 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).
Transformation Pipeline and Model Successfully Saved
```

Out[21]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('label_encoding',
                  TransformerWrapperWithInverse(exclude=None, include=None,
                                                transformer=LabelEncoder())),
                 ('numerical_imputer',
                  TransformerWrapper(exclude=None, include=['Age'],
                                     transformer=SimpleImputer(add_indicator=Fals
e,
                                                               copy=True,
                                                               fill_value=None,
                                                               keep_empty_features
=False,
                                                               missing_values=nan,
                                                               strategy='mean')...
                  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=123,
                                 reg_alpha=0.0, reg_lambda=0.0, subsample=1.0,
                                 subsample_for_bin=200000, subsample_freq=0))],
          verbose=False),
 'best_tuned_thyroid_model.pkl')
```

In [13]:
```
models()
```

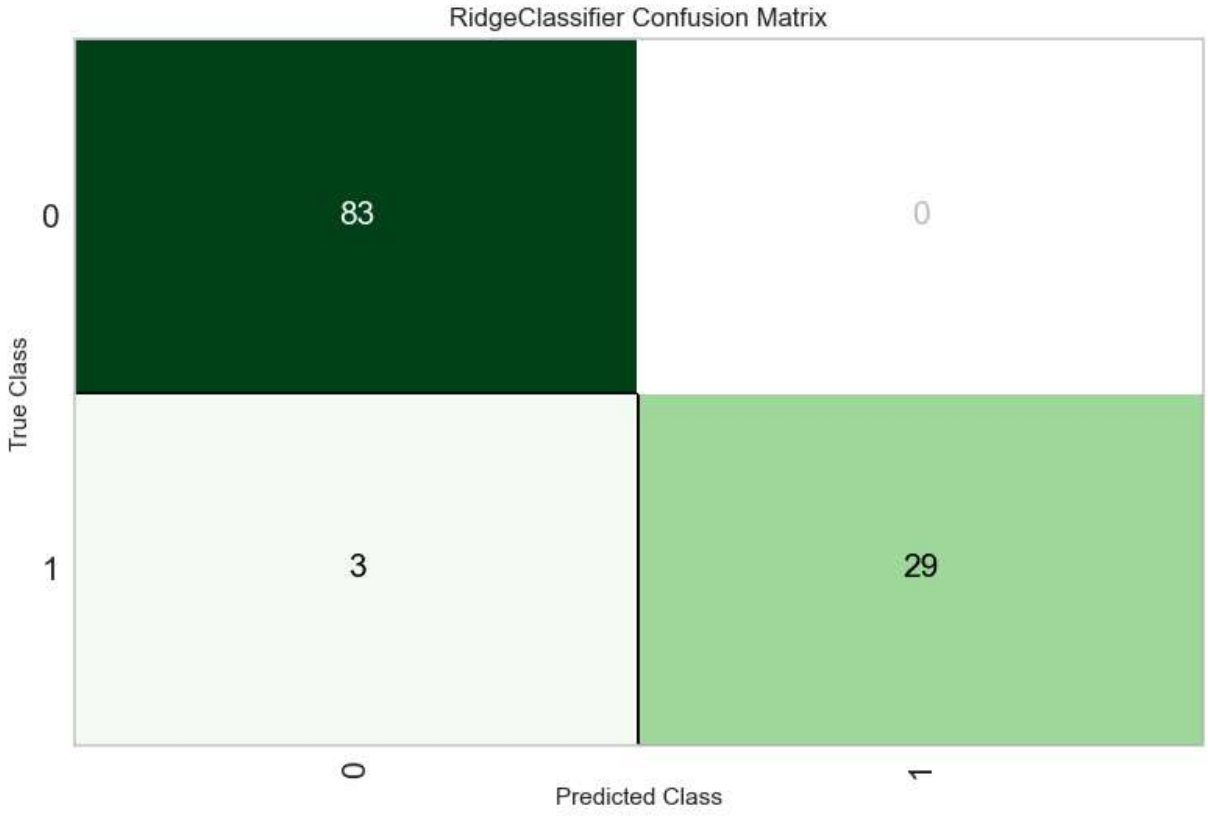
Out[13]:

ID	Name	Reference	Turbo
lr	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression	True
knn	K Neighbors Classifier	sklearn.neighbors._classification.KNeighborsCl...	True
nb	Naive Bayes	sklearn.naive_bayes.GaussianNB	True
dt	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier	True
svm	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDC...	True
rbfsvm	SVM - Radial Kernel	sklearn.svm._classes.SVC	False
gpc	Gaussian Process Classifier	sklearn.gaussian_process._gpc.GaussianProcessC...	False
mlp	MLP Classifier	sklearn.neural_network._multilayer_perceptron....	False
ridge	Ridge Classifier	sklearn.linear_model._ridge.RidgeClassifier	True
rf	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier	True
qda	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscrim...	True
ada	Ada Boost Classifier	sklearn.ensemble._weight_boosting.AdaBoostClas...	True
gbc	Gradient Boosting Classifier	sklearn.ensemble._gb.GradientBoostingClassifier	True
lda	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscrimina...	True
et	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier	True
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMClassifier	True
dummy	Dummy Classifier	sklearn.dummy.DummyClassifier	True

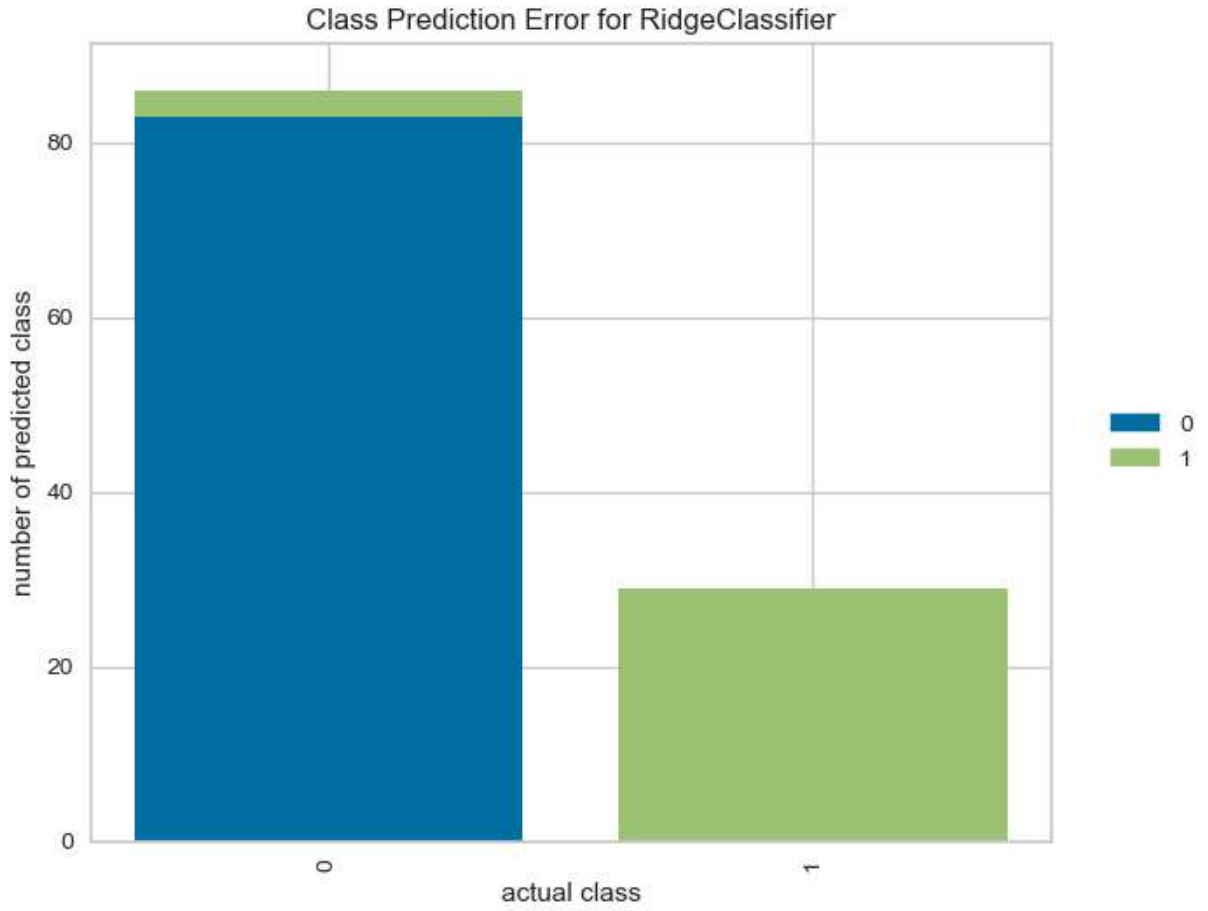
In [14]:
```
print(model_ridge_classifier)
```

```
RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True,
                max_iter=None, positive=False, random_state=123, solver='auto',
                tol=0.0001)
```

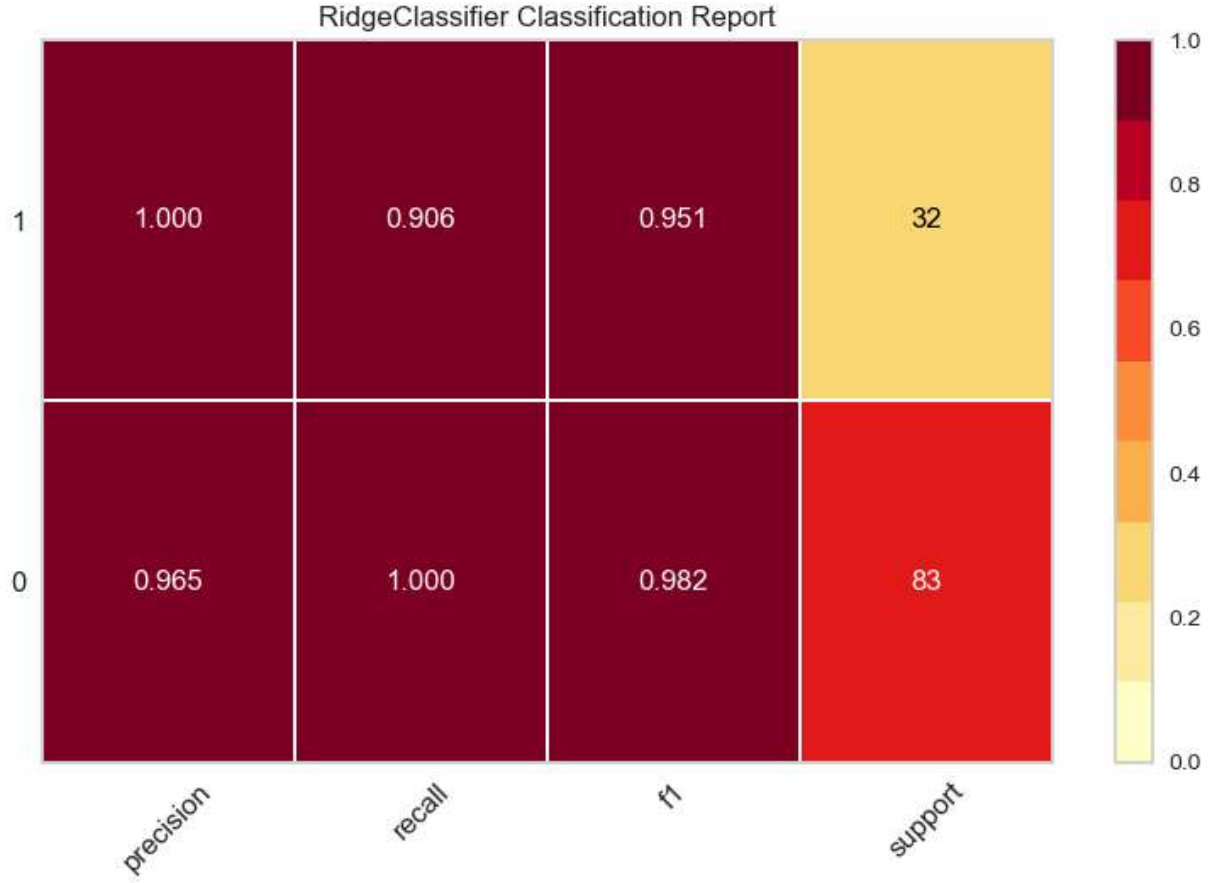
In [15]:
```
# Confusion matrix
plot_model(tune_ridge_classifier, plot='confusion_matrix')
```



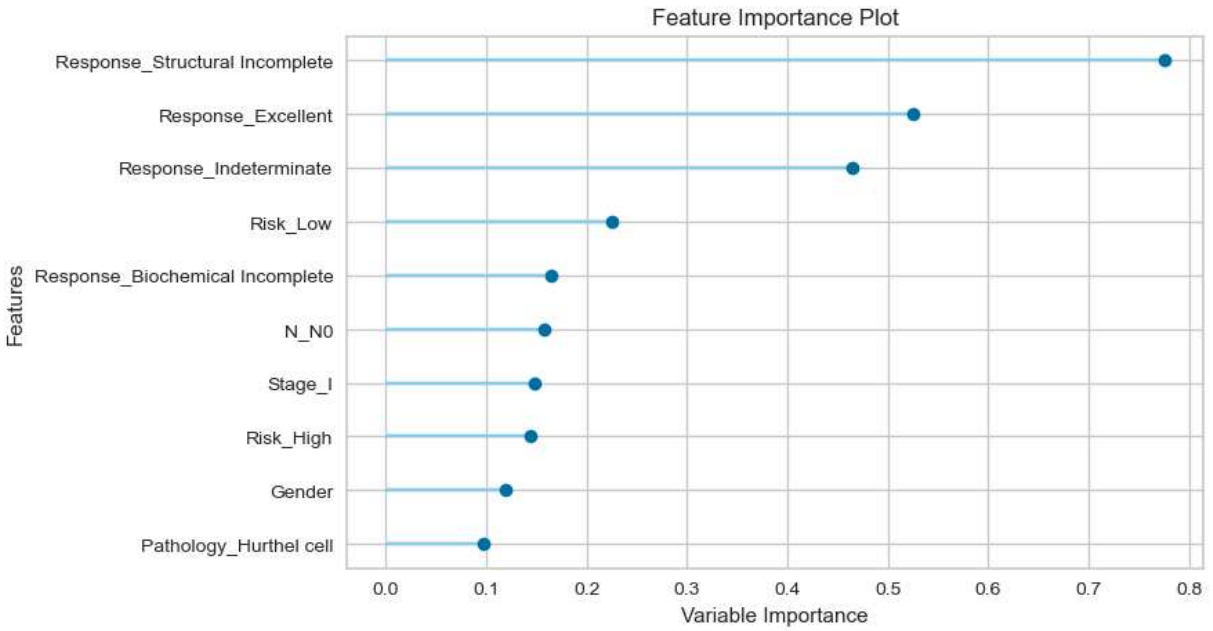
In [16]:
```
# Classification error
plot_model(tune_ridge_classifier, plot='error')
```



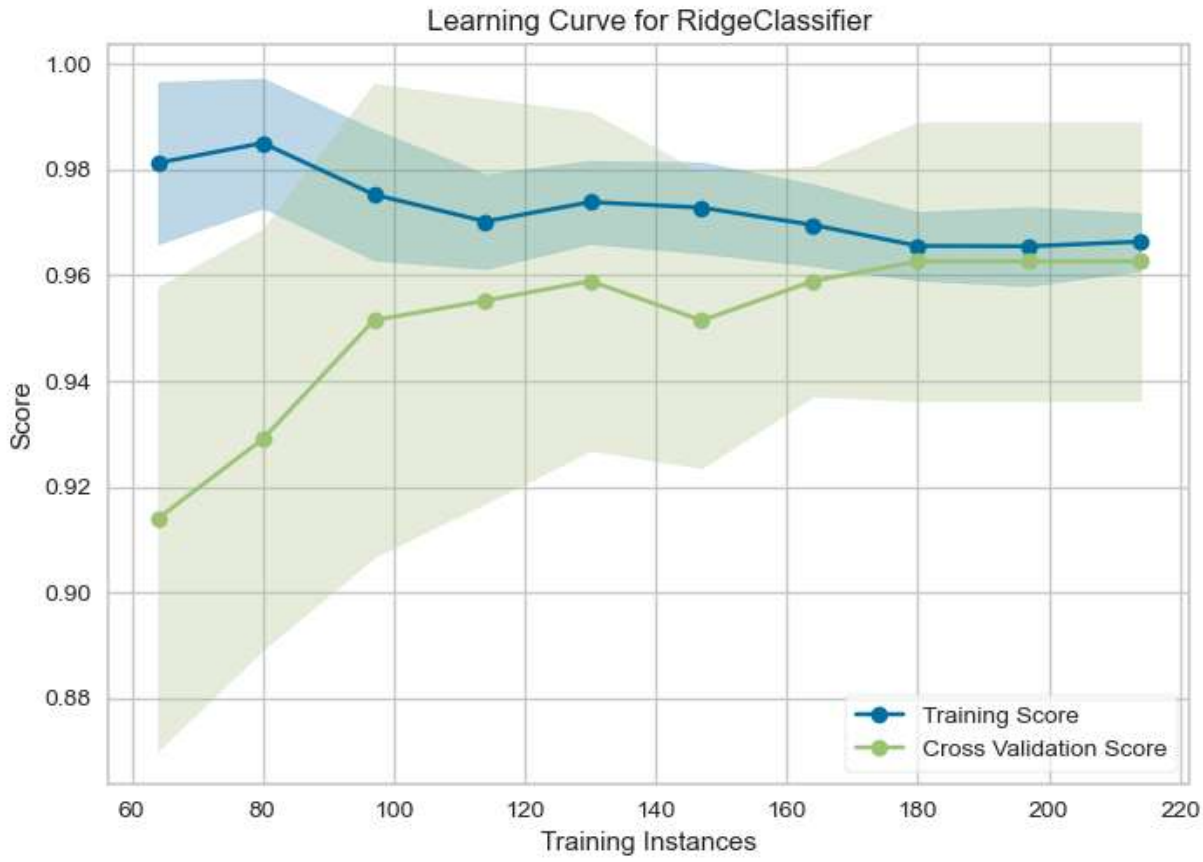
In [17]:
```
# Classification report (precision, recall, F1, etc.)
plot_model(tune_ridge_classifier, plot='class_report')
```



In [18]:
```
# Feature importance
plot_model(tune_ridge_classifier, plot='feature')
```



In [19]:
```
# Learning curve
plot_model(tune_ridge_classifier, plot='learning')
```



In [20]:
```
evaluate_model(tune_ridge_classifier)
```

```
interactive(children=(ToggleButtons(description='Plot Type:', icons=('',), options=
(('Pipeline Plot', 'pipelin…
```

In [ ]: