PyCaret Using Google Drive

↳ 3 cells hidden

Generate Data

```
# Generating Dataframe for taxonomic level MANUAL
link = "https://drive.google.com/file/d/1U-Uir1CIiuqXudeng6Rq86CaWo9rSqUd/view?usp=s

# to get the id part of the file
id = link.split("/")[-2]

downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile("training.csv")

training_df = pd.read_csv('training.csv')
# training_df = training_df.drop(columns = 'Unnamed: 0')
print(training_df)
```

```
        Sublevel Name  pp_magtropy  pp_avg_magnitude   entropy
0     Picornavirales    54.822801         75.778785  1.382249
1     Picornavirales    54.752611         75.747092  1.383443
2     Picornavirales    57.295723         79.198669  1.382279
3     Picornavirales    57.290198         79.048836  1.379797
4     Picornavirales    38.714513         53.025828  1.369663
..               ...          ...               ...       ...
260    Sobelivirales    42.572952         58.821615  1.381666
261    Sobelivirales    41.679669         57.657512  1.383349
262    Sobelivirales    41.679669         57.657512  1.383349
263    Sobelivirales    43.562799         60.146212  1.380678
264    Sobelivirales    41.161828         57.029920  1.385505

[265 rows x 4 columns]
```

```
# Generating Dataframe for COVID-19 Sequences
testing_link = "https://drive.google.com/file/d/1_SxcTlA9dDIergs__seb-DbnifluBQF6/vi

sublevel = input("Sublevel of Testing Data: ")
# to get the id part of the file
id = testing_link.split("/")[-2]

downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile('testing.csv')

testing_df = pd.read_csv('testing.csv')
testing_df = testing_df.drop(columns = 'Unnamed: 0')
testing_df = testing_df[testing_df['Sublevel Name'] == sublevel]
```

```
testing_df = testing_df[testing_df['Sublevel Name'] == sublevel]
print(testing_df)
```

```
Sublevel of Testing Data: Embecovirus
    Sublevel Name  pp_magtropy  pp_avg_magnitude   entropy
112   Embecovirus   114.269624        153.103733  1.339846
113   Embecovirus   114.111031        155.141480  1.359566
114   Embecovirus   114.987320        153.815693  1.337675
115   Embecovirus   114.226726        153.062393  1.339988
116   Embecovirus   114.320187        153.136267  1.339538
..            ...          ...               ...       ...
207   Embecovirus   112.497193        153.807531  1.367212
208   Embecovirus   114.288491        153.117355  1.339744
209   Embecovirus   114.870606        153.996769  1.340611
210   Embecovirus   115.440977        150.518479  1.303857
211   Embecovirus   114.422743        153.317131  1.339918

[100 rows x 4 columns]
```

Magtropy

```
magtropy_df = training_df.drop(columns = ["pp_avg_magnitude", "entropy"])
print(magtropy_df)
```

```
       Sublevel Name  pp_magtropy
0    Picornavirales    54.822801
1    Picornavirales    54.752611
2    Picornavirales    57.295723
3    Picornavirales    57.290198
4    Picornavirales    38.714513
..              ...          ...
260   Sobelivirales    42.572952
261   Sobelivirales    41.679669
262   Sobelivirales    41.679669
263   Sobelivirales    43.562799
264   Sobelivirales    41.161828

[265 rows x 2 columns]
```

```
experiment = setup(data=magtropy_df, target='Sublevel Name')
# if the error states target is not defined, change from Sublevel_Name to Sublevel N
# label encodings alphabetical
```

	Description	Value
0	session_id	7443
1	Target	Sublevel Name
2	Target Type	Multiclass
3	Label Encoded	Nidovirales: 0, Picornavirales: 1, Sobeliviral...
4	Original Data	(265, 2)
5	Missing Values	False
6	Numeric Features	1
7	Categorical Features	0
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(185, 1)
12	Transformed Test Set	(80, 1)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	c0d7
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent
29	Normalize	False

30	Normalize Method	None
31	Transformation	False
32	Transformation Method	None
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	False
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	False
43	Multicollinearity Threshold	None
44	Clustering	False
45	Clustering Iteration	None
46	Polynomial Features	False
47	Polynomial Degree	None
48	Trignometry Features	False
49	Polynomial Threshold	None
50	Group Features	False
51	Feature Selection	False
52	Features Selection Threshold	None

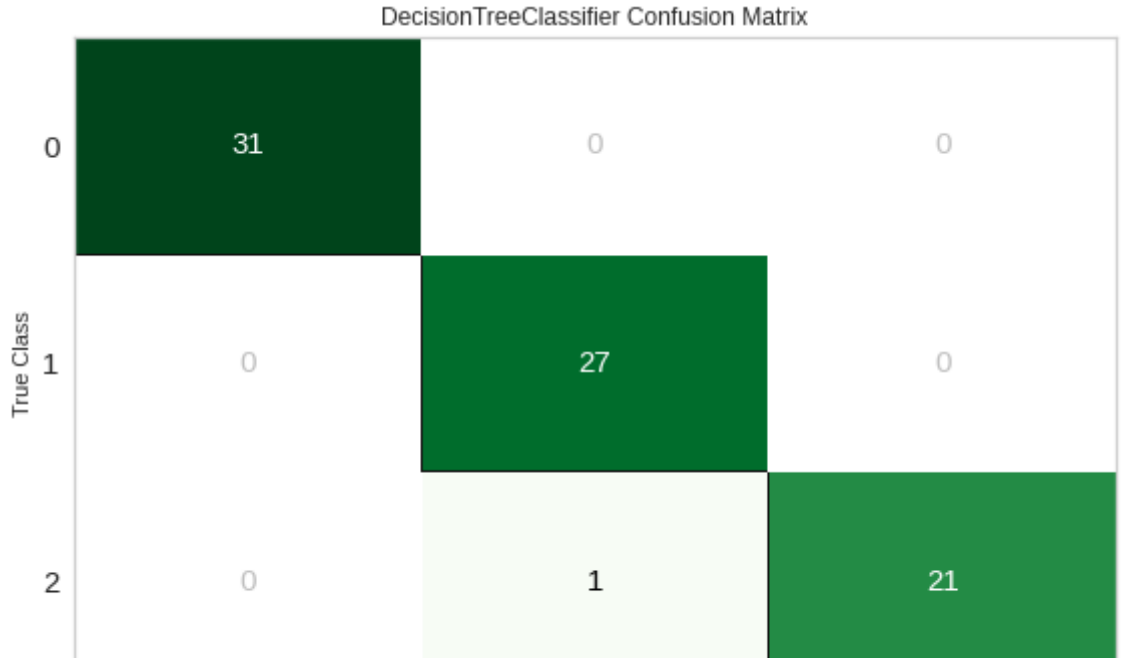
```
compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dt	Decision Tree Classifier	0.9892	0.9916	0.9869	0.9907	0.9890	0.9833	0.9842	0.021
rf	Random Forest Classifier	0.9892	0.9962	0.9869	0.9907	0.9890	0.9833	0.9842	0.470
gbc	Gradient Boosting Classifier	0.9892	0.9968	0.9869	0.9907	0.9890	0.9833	0.9842	0.186
et	Extra Trees Classifier	0.9892	1.0000	0.9869	0.9907	0.9890	0.9833	0.9842	0.465
catboost	CatBoost Classifier	0.9892	1.0000	0.9869	0.9907	0.9890	0.9833	0.9842	0.738
lightgbm	Light Gradient	0.9839	0.9981	0.9869	0.9870	0.9838	0.9752	0.9771	0.050

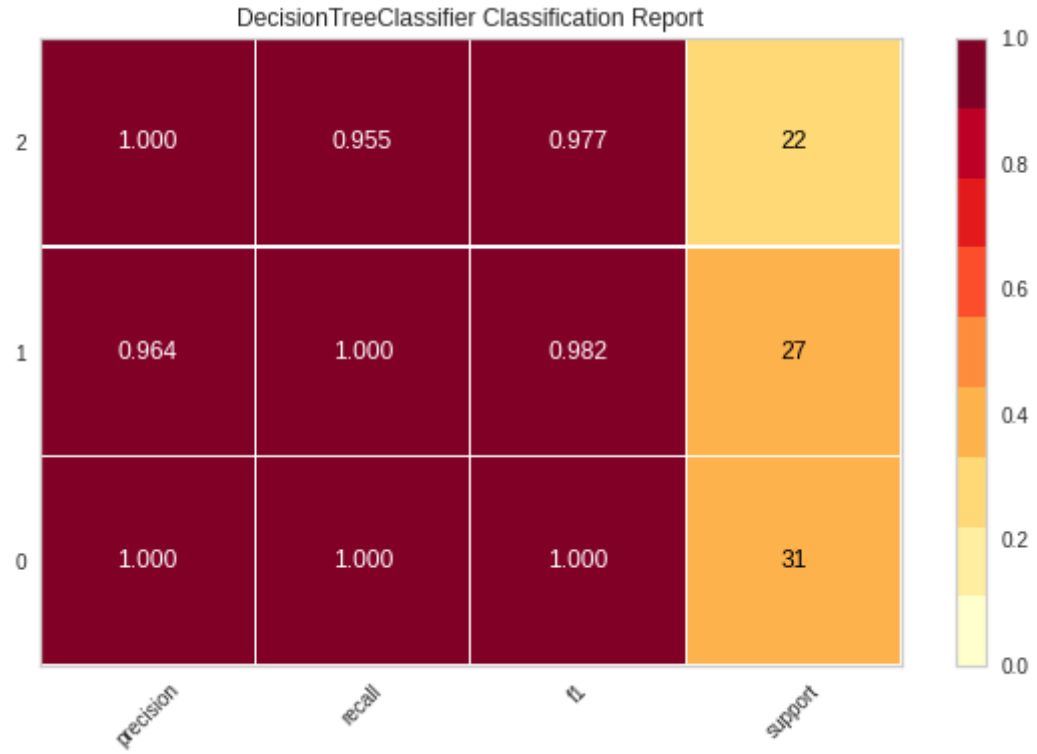
```
estimator = create_model('dt')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9474	0.9545	0.9167	0.9532	0.9452	0.9167	0.9209
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	0.9444	0.9615	0.9524	0.9537	0.9448	0.9167	0.9209
6	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Mean	0.9892	0.9916	0.9869	0.9907	0.9890	0.9833	0.9842
SD	0.0216	0.0169	0.0274	0.0186	0.0220	0.0333	0.0316

```
plot_model(estimator, 'confusion_matrix')
```




```
plot_model(estimator, 'class_report')
```




```
magtropy_testing_df = testing_df.drop(columns = ["pp_avg_magnitude", "entropy"])
print(magtropy_testing_df)
```

```
     Sublevel Name  pp_magtropy
112    Embecovirus   114.269624
113    Embecovirus   114.111031
114    Embecovirus   114.987320
115    Embecovirus   114.226726
116    Embecovirus   114.320187
..             ...          ...
```

```
207   Embecovirus   112.497193
208   Embecovirus   114.288491
209   Embecovirus   114.870606
210   Embecovirus   115.440977
211   Embecovirus   114.422743

[100 rows x 2 columns]
```

```
X_test = magtropy_testing_df.drop(columns = ["Sublevel Name"])
predict = estimator.predict(X_test)
print(predict)
print(len(predict))
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
100
```

```
unique_elements, count_elements = np.unique(predict, return_counts = "True")
results = np.asarray((unique_elements, count_elements))
print(results)
```

```
[[  0]
 [100]]
```

Magnitude avg

```
avg_magnitude_df = training_df.drop(columns = ["pp_magtropy", "entropy"])
print(avg_magnitude_df)
```

```
      Sublevel Name  pp_avg_magnitude
0    Picornavirales         75.778785
1    Picornavirales         75.747092
2    Picornavirales         79.198669
3    Picornavirales         79.048836
4    Picornavirales         53.025828
..              ...               ...
260   Sobelivirales         58.821615
261   Sobelivirales         57.657512
262   Sobelivirales         57.657512
263   Sobelivirales         60.146212
264   Sobelivirales         57.029920

[265 rows x 2 columns]
```

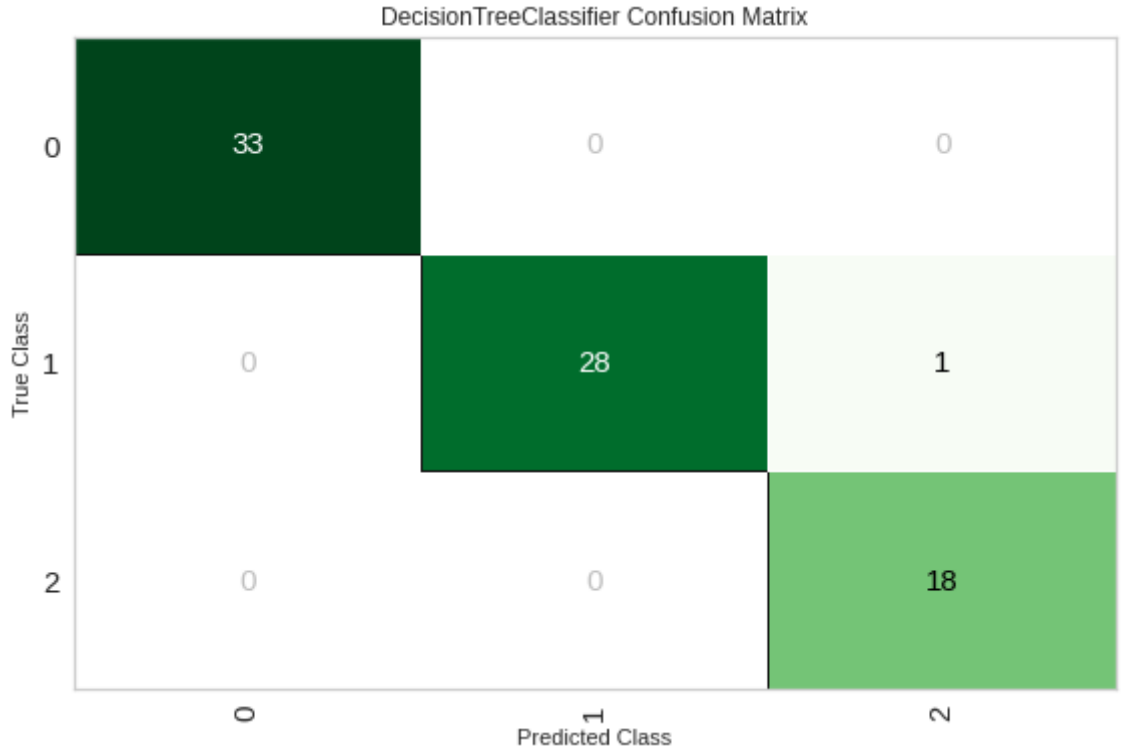
```
experiment = setup(data=avg_magnitude_df, target='Sublevel Name')
```

```
compare_models()
```

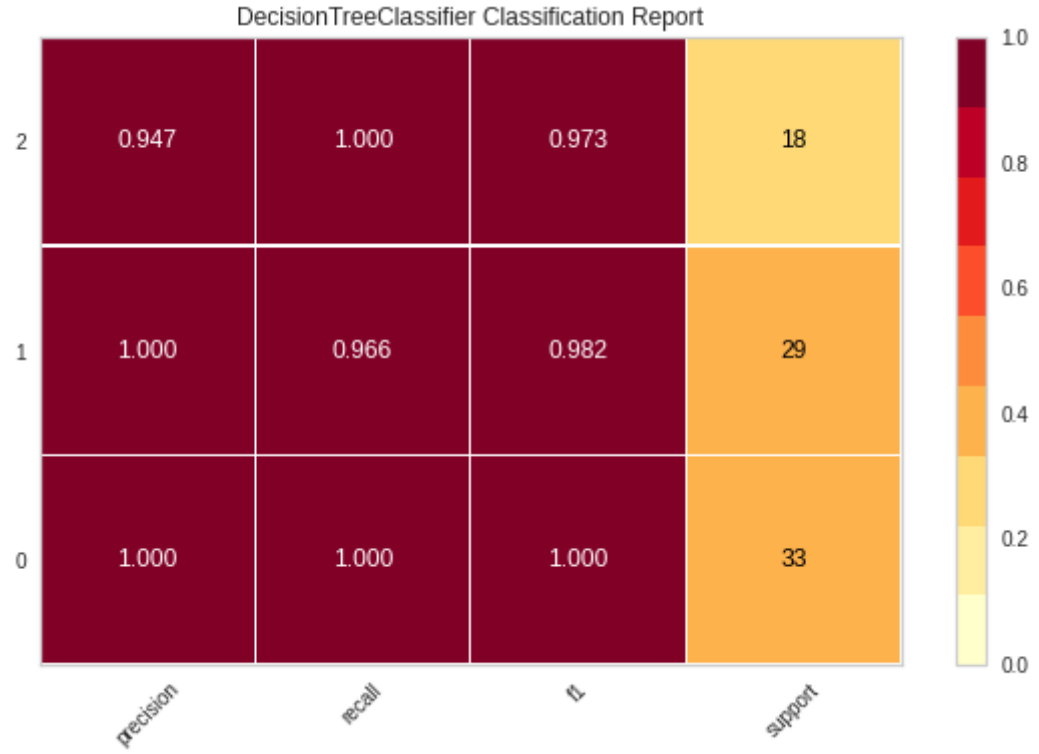
```
estimator = create_model('dt')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Mean	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SD	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
plot_model(estimator, 'confusion_matrix')
```




```
plot_model(estimator, 'class_report')
```




```
magnitude_avg_testing_df = testing_df.drop(columns = ["pp_magtropy", "entropy"])
print(magnitude_avg_testing_df )
```

```
     Sublevel Name  pp_avg_magnitude
112    Embecovirus        153.103733
113    Embecovirus        155.141480
114    Embecovirus        153.815693
115    Embecovirus        153.062393
116    Embecovirus        153.136267
..             ...               ...
207    Embecovirus        153.807531
208    Embecovirus        153.117355
209    Embecovirus        153.996769
210    Embecovirus        150.518479
211    Embecovirus        153.317131

[100 rows x 2 columns]
```

```
X_test = magnitude_avg_testing_df.drop(columns = ["Sublevel Name"])
predict = estimator.predict(X_test)
print(predict)
print(len(predict))
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
100
```

```
unique_elements, count_elements = np.unique(predict, return_counts = "True")
results = np.asarray((unique_elements, count_elements))
print(results)
```

```
print(results)
```

```
[[  0]
 [100]]
```

Entropy

```
entropy_df = training_df.drop(columns = ["pp_magtropy","pp_avg_magnitude"])
print(entropy_df)
```

```
       Sublevel Name   entropy
0    Picornavirales  1.382249
1    Picornavirales  1.383443
2    Picornavirales  1.382279
3    Picornavirales  1.379797
4    Picornavirales  1.369663
..              ...       ...
260   Sobelivirales  1.381666
261   Sobelivirales  1.383349
262   Sobelivirales  1.383349
263   Sobelivirales  1.380678
264   Sobelivirales  1.385505

[265 rows x 2 columns]
```

```
experiment = setup(data=entropy_df, target='Sublevel Name')
```

	Description	Value
0	session_id	5571
1	Target	Sublevel Name
2	Target Type	Multiclass
3	Label Encoded	Nidovirales: 0, Picornavirales: 1, Sobeliviral...
4	Original Data	(265, 2)
5	Missing Values	False
6	Numeric Features	1
7	Categorical Features	0
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(185, 1)
12	Transformed Test Set	(80, 1)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	7844
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent
29	Normalize	False

30	Normalize Method	None
31	Transformation	False
32	Transformation Method	None
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	False
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	False
43	Multicollinearity Threshold	None

```
compare_models()
```

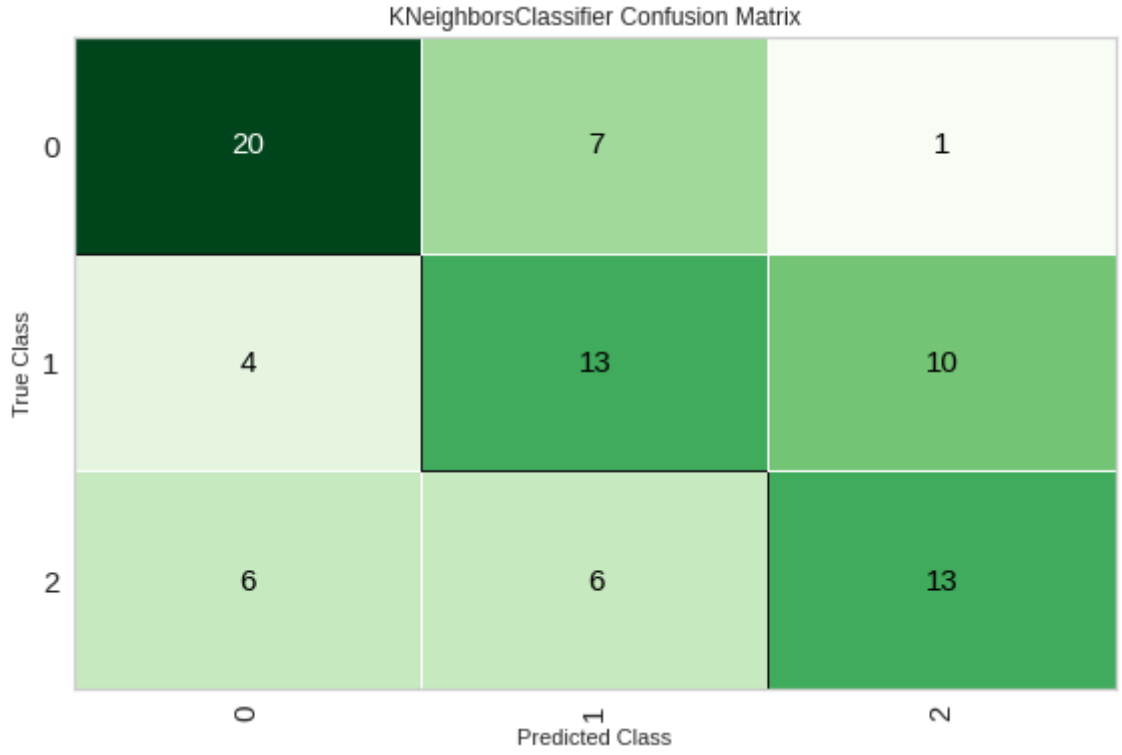
Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)

```
estimator = create_model('knn')
```

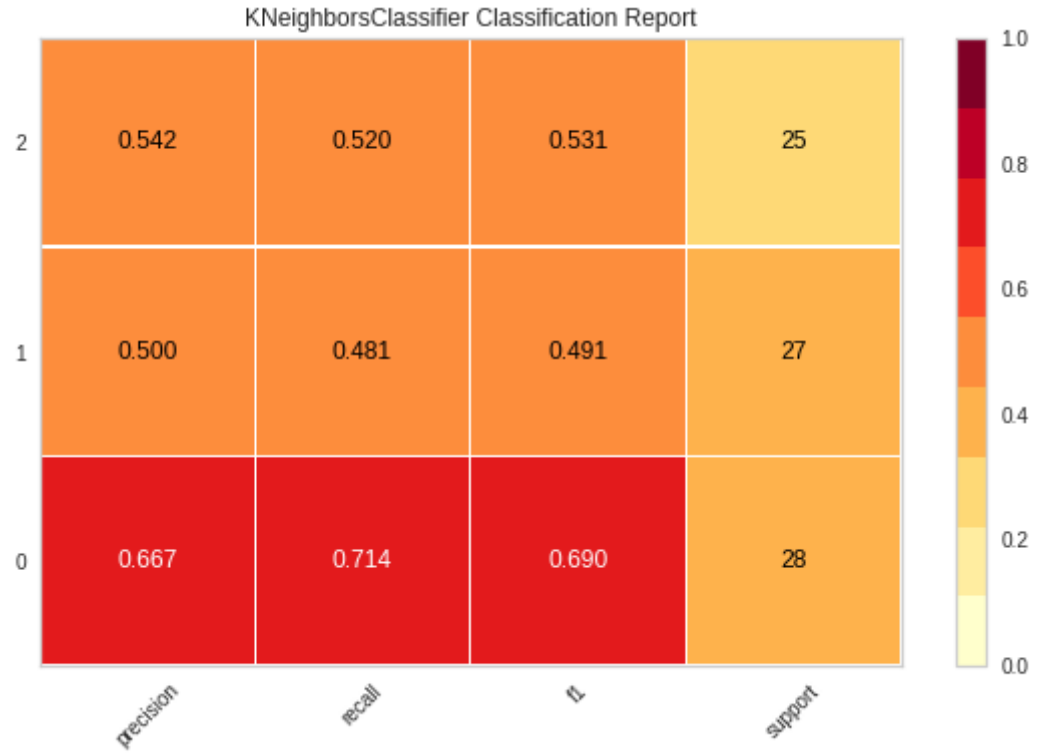
	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6316	0.8336	0.6548	0.6288	0.6281	0.4292	0.4310
1	0.6842	0.7884	0.6548	0.7059	0.6874	0.5043	0.5111
2	0.5789	0.7315	0.5238	0.6474	0.5614	0.3122	0.3236
3	0.5789	0.7621	0.5893	0.5405	0.5353	0.3667	0.3878
4	0.6842	0.8411	0.6905	0.7018	0.6832	0.5128	0.5195
5	0.6667	0.7859	0.6429	0.6883	0.6692	0.4783	0.4855
6	0.5556	0.7307	0.5119	0.5278	0.5346	0.3043	0.3090
7	0.6111	0.7063	0.5595	0.6944	0.5954	0.3731	0.3924
8	0.5556	0.7493	0.5833	0.5889	0.5593	0.3333	0.3396
9	0.6667	0.8185	0.6429	0.6782	0.6695	0.4857	0.4880
Mean	0.6213	0.7747	0.6054	0.6402	0.6123	0.4100	0.4187
SD	0.0495	0.0440	0.0576	0.0634	0.0592	0.0775	0.0756

Machine

```
plot_model(estimator, 'confusion_matrix')
```



```
plot_model(estimator, 'class_report')
```




```
entropy_testing_df = testing_df.drop(columns = ["pp_avg_magnitude", "pp_magtropy"])
print(entropy_testing_df)
```

```
     Sublevel Name   entropy
112   Embecovirus  1.339846
113   Embecovirus  1.359566
114   Embecovirus  1.337675
115   Embecovirus  1.339988
116   Embecovirus  1.339538
..            ...       ...
207   Embecovirus  1.367212
208   Embecovirus  1.339744
209   Embecovirus  1.340611
210   Embecovirus  1.303857
211   Embecovirus  1.339918

[100 rows x 2 columns]
```

```
X_test =entropy_testing_df.drop(columns = ["Sublevel Name"])
predict = estimator.predict(X_test)
print(predict)
print(len(predict))
```

```
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
100
```

```
unique_elements, count_elements = np.unique(predict, return_counts = "True")
results = np.asarray((unique_elements, count_elements))
print(results)
```

```
   [[ 0  1]
    [97  3]]
```