DT PRACTICAL - 2

Installing pycaret

```
!pip install pycaret &> /dev/null
print ("Pycaret installed sucessfully!!")

    Pycaret installed sucessfully!!
```

Get pycaret version

```
from pycaret.utils import version
version()

    '2.3.10'
```

Classification: Basics

Loading Dataset - Loading dataset from pycaret

```
from pycaret.datasets import get_data
```

Get the list of datasets available in pycaret (55)

```
dataSets = get_data('index')

# instances = number of rows
# attributes/features = number of columns
```

	Dataset	Data Types	Default Task	Target Variable 1	T Var
0	anomaly	Multivariate	Anomaly Detection	None	

1	france	Multivariate	Association Rule Mining	InvoiceNo	Desc
2	germany	Multivariate	Association Rule Mining	InvoiceNo	Desc
3	bank	Multivariate	Classification (Binary)	deposit	
4	blood	Multivariate	Classification (Binary)	Class	
5	cancer	Multivariate	Classification (Binary)	Class	
6	credit	Multivariate	Classification (Binary)	default	
7	diabetes	Multivariate	Classification (Binary)	Class variable	
8	electrical_grid	Multivariate	Classification (Binary)	stabf	
9	employee	Multivariate	Classification (Binary)	left	
10	heart	Multivariate	Classification (Binary)	DEATH	
11	heart_disease	Multivariate	Classification (Binary)	Disease	
12	hepatitis	Multivariate	Classification (Binary)	Class	
13	income	Multivariate	Classification (Binary)	income >50K	
14	juice	Multivariate	Classification (Binary)	Purchase	
15	nba	Multivariate	Classification (Binary)	TARGET_5Yrs	
16	wine	Multivariate	Classification (Binary)	type	
17	telescope	Multivariate	Classification (Binary)	Class	
18	titanic	Multivariate	Classification (Binary)	Survived	
19	us_presidential_election_results	Multivariate	Classification (Binary)	party_winner	
20	glass	Multivariate	Classification (Multiclass)	Type	
21	iris	Multivariate	Classification (Multiclass)	species	

22	poker	Multivariate	Classification (Multiclass)	CLASS
23	questions	Multivariate	Classification (Multiclass)	Next_Question
24	satellite	Multivariate	Classification (Multiclass)	Class
25	CTG	Multivariate	Classification (Multiclass)	NSP
26	asia_gdp	Multivariate	Clustering	None
27	elections	Multivariate	Clustering	None
28	facebook	Multivariate	Clustering	None
29	ipl	Multivariate	Clustering	None
30	jewellery	Multivariate	Clustering	None
31	mice	Multivariate	Clustering	None
32	migration	Multivariate	Clustering	None
33	perfume	Multivariate	Clustering	None
34	pokemon	Multivariate	Clustering	None
35	population	Multivariate	Clustering	None
36	public_health	Multivariate	Clustering	None
37	seeds	Multivariate	Clustering	None
38	wholesale	Multivariate	Clustering	None
39	tweets	Text	NLP	tweet
40	amazon	Text	NLP / Classification	reviewText
			NLP /	

Get diabetes dataset

```
juiceDataSet = get_data("juice")
```

```
print(type(juiceDataSet))
```

	Id	Purchase	WeekofPurchase	StoreID	PriceCH	PriceMM	DiscCH	DiscMM	S
0	1	CH	237	1	1.75	1.99	0.00	0.0	
1	2	CH	239	1	1.75	1.99	0.00	0.3	
2	3	CH	245	1	1.86	2.09	0.17	0.0	

3	4	MM	227	1	1.69	1.69	0.00	0.0
4	5	CH	228	7	1.69	1.69	0.00	0.0

```
<class 'pandas.core.frame.DataFrame'>
```

```
juiceDataSet.columns
```

```
Index(['Id', 'Purchase', 'WeekofPurchase', 'StoreID', 'PriceCH',
'PriceMM',
       'DiscCH', 'DiscMM', 'SpecialCH', 'SpecialMM', 'LoyalCH',
'SalePriceMM',
       'SalePriceCH', 'PriceDiff', 'Store7', 'PctDiscMM', 'PctDiscCH',
       'ListPriceDiff', 'STORE'],
      dtype='object')
```

```
juiceDataSet.describe()
```

	Id	WeekofPurchase	StoreID	PriceCH	PriceMM
count	1070.000000	1070.000000	1070.000000	1070.000000	1070.000000
mean	535.500000	254.381308	3.959813	1.867421	2.085411
std	309.026698	15.558286	2.308984	0.101970	0.134386
min	1.000000	227.000000	1.000000	1.690000	1.690000
25%	268.250000	240.000000	2.000000	1.790000	1.990000
50%	535.500000	257.000000	3.000000	1.860000	2.090000
75%	802.750000	268.000000	7.000000	1.990000	2.180000
max	1070.000000	278.000000	7.000000	2.090000	2.290000

```
print("type(juiceDataSet)-->",type(juiceDataSet))
```

```
type(juiceDataSet)--> <class 'pandas.core.frame.DataFrame'>
```

```
print("juiceDataSet.shape -->", diabetesDataSet.shape)
print("Rows     -->", diabetesDataSet.shape[0])  ##axis 0---row
print("Columns  -->", diabetesDataSet.shape[1])
```

```
juiceDataSet.shape --> (1070, 19)
Rows     --> 1070
Columns  --> 19
```

```
juiceDataSet.head()
juiceDataSet.head()
```

	Id	Purchase	WeekofPurchase	StoreID	PriceCH	PriceMM	DiscCH	DiscMM	S
0	1	CH	237	1	1.75	1.99	0.00	0.0	
1	2	CH	239	1	1.75	1.99	0.00	0.3	
2	3	CH	245	1	1.86	2.09	0.17	0.0	
3	4	MM	227	1	1.69	1.69	0.00	0.0	
4	5	CH	228	7	1.69	1.69	0.00	0.0	

```
juiceDataSet.loc[10:20 , ['WeekofPurchase','StoreID']]
```

	WeekofPurchase	StoreID
10	240	7
11	263	7
12	276	7
13	268	7
14	278	7
15	278	7
16	240	1
17	268	2
18	269	2
19	254	7
20	257	7

```
diabetesDataSet.max()
```

```
Id                    1070
Purchase                MM
WeekofPurchase         278
StoreID                  7
PriceCH               2.09
PriceMM               2.29
DiscCH                 0.5
DiscMM                 0.8
SpecialCH                1
SpecialMM                1
LoyalCH           0.999947
SalePriceMM           2.29
SalePriceCH           2.09
PriceDiff             0.64
PriceDiff            0.64
Store7                Yes
PctDiscMM         0.40201
PctDiscCH        0.252688
ListPriceDiff        0.44
STORE                   4
dtype: object
```

```
juiceDataSet.isnull().sum()
```

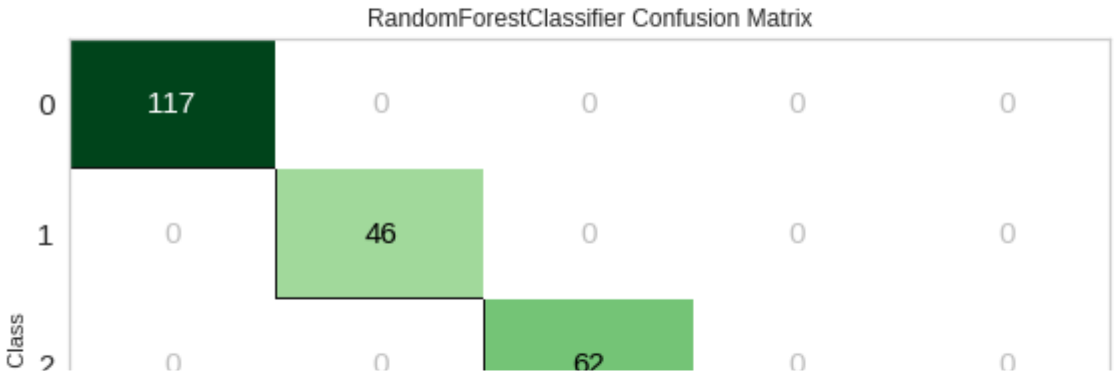
```
Id                0
Purchase          0
WeekofPurchase    0
StoreID           0
PriceCH           0
PriceMM           0
DiscCH            0
DiscMM            0
SpecialCH         0
SpecialMM         0
LoyalCH           0
SalePriceMM       0
SalePriceCH       0
PriceDiff         0
Store7            0
PctDiscMM         0
PctDiscCH         0
ListPriceDiff     0
STORE             0
dtype: int64
```

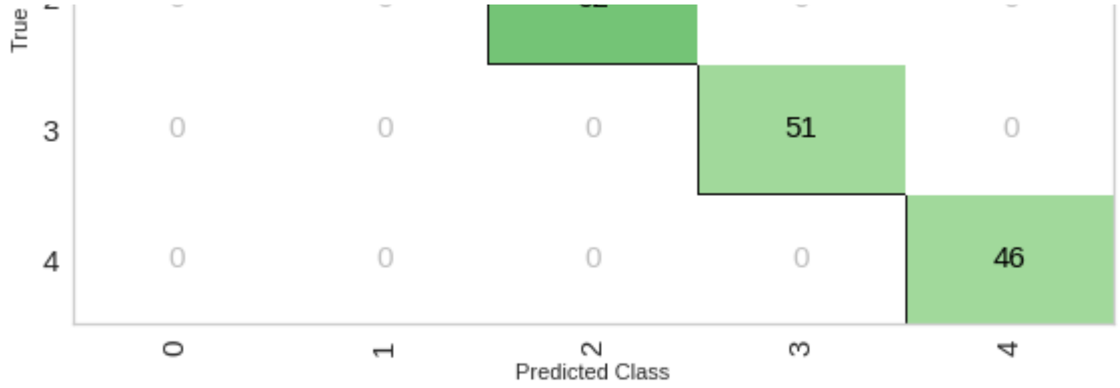
Build a single model - "RandomForest"

```
#from pycaret.datasets import get_data
from pycaret.classification import *

#diabetesDataSet = get_data("diabetes")
s = setup(data=juiceDataSet, target='STORE', silent=True)

rfModel = create_model('rf')
plot_model(rfModel, plot='confusion_matrix')
#Explore more parameters
```





```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully completed..............................
```

Save the trained model

```
sm = save_model(rfModel, 'rfModelFile')
```

Make prediction on the new dataset

Get new dataset

```
newDataSet = get_data("juice").iloc[:10]
```

	Id	Purchase	WeekofPurchase	StoreID	PriceCH	PriceMM	DiscCH	DiscMM	S
0	1	CH	237	1	1.75	1.99	0.00	0.0	
1	2	CH	239	1	1.75	1.99	0.00	0.3	
2	3	CH	245	1	1.86	2.09	0.17	0.0	
3	4	MM	227	1	1.69	1.69	0.00	0.0	
4	5	CH	228	7	1.69	1.69	0.00	0.0	

Make prediction on new dataset

```
newPredictions = predict_model(rfModel, data = newDataSet)
newPredictions
```

```
INFO:logs:Initializing predict_model()
```

```
INFO:logs:predict_model(estimator=RandomForestClassifier(bootstrap=True, cc
                       criterion='gini', max_depth=None, max_features='auto
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=3145, verbo
                       warm_start=False), probability_threshold=None, encod
INFO:logs:Checking exceptions
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Random Forest Classifier	1.0	1.0	1.0	1.0	1.0	1.0	1.0

	Id	Purchase	WeekofPurchase	StoreID	PriceCH	PriceMM	DiscCH	DiscMM	S
0	1	CH	237	1	1.75	1.99	0.00	0.0	
1	2	CH	239	1	1.75	1.99	0.00	0.3	
2	3	CH	245	1	1.86	2.09	0.17	0.0	
3	4	MM	227	1	1.69	1.69	0.00	0.0	
4	5	CH	228	7	1.69	1.69	0.00	0.0	
5	6	CH	230	7	1.69	1.99	0.00	0.0	
6	7	CH	232	7	1.69	1.99	0.00	0.4	
7	8	CH	234	7	1.75	1.99	0.00	0.4	
8	9	CH	235	7	1.75	1.99	0.00	0.4	
9	10	CH	238	7	1.75	1.99	0.00	0.4	

10 rows × 21 columns

Save prediction results to csv

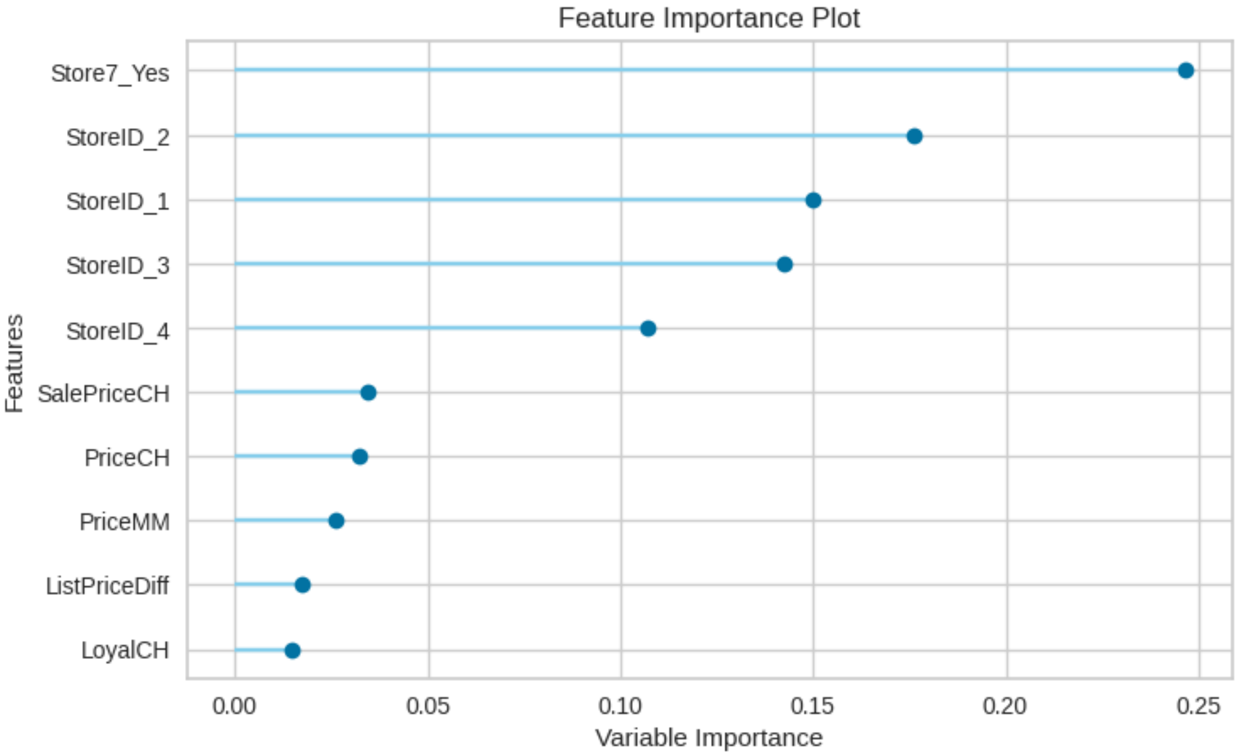
```
newPredictions.to_csv("NewPredictions.csv")
print('predictions saved successfully')
```

```
predictions saved successfully
```

Feature Importance

Feature Importance using Random Forest

```
rfModel = create_model('rf', verbose=True)
plot_model(rfModel, plot='feature')
```



```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully completed.................................
```

Run and compare the Model Performance

```
cm = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
ridge	Ridge Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000

ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
knn	K Neighbors Classifier	0.9479	0.9978	0.9307	0.9518	0.9468	0.9326	0.933

Model Performance using data "Normalization"

```
s = setup(data=juiceDataSet, target='STORE', normalize = True, normalize_method
cm = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
ridge	Ridge Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000

knn	K Neighbors Classifier	0.9625	0.9963	0.9535	0.9651	0.9623	0.9514	0.952

Model Performance using "Feature Selection"

```
s = setup(data=juiceDataSet, target='STORE', feature_selection = True, feature_s
cm = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
ridge	Ridge Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
svm	SVM - Linear Kernel	0.9813	0.0000	0.9800	0.9714	0.9748	0.9757	0.979
knn	K Neighbors Classifier	0.9599	0.9968	0.9423	0.9632	0.9581	0.9476	0.948

Model Performance using "Outlier Removal"

```
s = setup(data=juiceDataSet, target='STORE', remove_outliers = True, outliers_th
cm = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
ridge	Ridge Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
knn	K Neighbors Classifier	0.9521	0.9976	0.9423	0.9563	0.9509	0.9386	0.940

Model Performance using "Transformation"

```
s = setup(data=juiceDataSet, target='STORE', transformation = True, transformati
cm = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
lr	Logistic Regression	1.0000	1.0000	1.000	1.0000	1.0000	1.0000	1.000
nb	Naive Bayes	1.0000	1.0000	1.000	1.0000	1.0000	1.0000	1.000
dt	Decision Tree Classifier	1.0000	1.0000	1.000	1.0000	1.0000	1.0000	1.000
svm	SVM - Linear Kernel	1.0000	0.0000	1.000	1.0000	1.0000	1.0000	1.000

ridge	Ridge Classifier	1.0000	0.0000	1.000	1.0000	1.0000	1.0000	1.000
rf	Random Forest Classifier	1.0000	1.0000	1.000	1.0000	1.0000	1.0000	1.000
ada	Ada Boost Classifier	1.0000	1.0000	1.000	1.0000	1.0000	1.0000	1.000
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.000	1.0000	1.0000	1.0000	1.000
et	Extra Trees Classifier	1.0000	1.0000	1.000	1.0000	1.0000	1.0000	1.000
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.000	1.0000	1.0000	1.0000	1.000
knn	K Neighbors Classifier	0.9491	0.9942	0.940	0.9537	0.9489	0.9344	0.935

Model Performance using "PCA"

```
s = setup(data=juiceDataSet, target='STORE', pca = True, pca_method = 'linear',
cm = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
dt	Decision Tree Classifier	0.4934	0.6765	0.4569	0.5037	0.4933	0.3437	0.345
rf	Random Forest Classifier	0.4934	0.7223	0.4569	0.5037	0.4933	0.3437	0.345
et	Extra Trees Classifier	0.4934	0.6936	0.4567	0.5045	0.4939	0.3443	0.346
gbc	Gradient Boosting Classifier	0.4413	0.6845	0.3717	0.4192	0.4119	0.2512	0.259
knn	K Neighbors Classifier	0.4038	0.6818	0.3230	0.3617	0.3691	0.2019	0.208
lightgbm	Light Gradient Boosting Machine	0.3997	0.6601	0.3393	0.3811	0.3789	0.2048	0.208
lr	Logistic Regression	0.3369	0.5247	0.2000	0.1135	0.1698	0.0000	0.000

nb	Naive Bayes	0.3369	0.5292	0.2000	0.1135	0.1698	0.0000	0.000
ridge	Ridge Classifier	0.3369	0.0000	0.2000	0.1135	0.1698	0.0000	0.000
qda	Quadratic Discriminant Analysis	0.3369	0.5291	0.2000	0.1135	0.1698	0.0000	0.000
lda	Linear Discriminant	0.3369	0.5254	0.2000	0.1135	0.1698	0.0000	0.000

Model Performance using "Outlier Removal" + "Normalization"

```
s = setup(data=juiceDataSet, target='STORE', remove_outliers = True, outliers_th
cm = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
ridge	Ridge Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
knn	K Neighbors Classifier	0.9690	0.9965	0.9623	0.9713	0.9688	0.9604	0.961

Model Performance using "Outlier Removal" + "Normalization" +

Model Performance using "Outlier Removal" + "Normalization" + "Transformation"

```
s = setup(data=juiceDataSet, target='STORE', remove_outliers = True, outliers_th
cm = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
ridge	Ridge Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.000
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
knn	K Neighbors Classifier	0.9521	0.9969	0.9415	0.9552	0.9506	0.9386	0.939