In [ ]:

```
#Data Set Information:

#This research aimed at the case of customersâ€™ default payments in Taiwan a


#Attribute Information:

#This research employed a binary variable, default payment (Yes = 1, No = 0),
#X1: Amount of the given credit (NT dollar): it includes both the individual 
#X2: Gender (1 = male; 2 = female).
#X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = oth
#X4: Marital status (1 = married; 2 = single; 3 = others).
#X5: Age (year).
#X6 - X11: History of past payment. We tracked the past monthly payment recor
#X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statemen
#X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in Septem
```

In [1]:

```
import pycaret
from pycaret.datasets import get_data
dataset = get_data('credit', profile=True)
```

20000	1548	6.5%
30000	1286	5.4%
40000	187	0.8%
50000	2698	11.2%
60000	649	2.7%
70000	588	2.5%
80000	1282	5.3%
90000	512	2.1%

Value	Count	Frequency (%)
1000000	1	< 0.1%
800000	2	< 0.1%
780000	2	< 0.1%
760000	1	< 0.1%
750000	4	< 0.1%
740000	2	< 0.1%
730000	2	< 0.1%
720000	2	< 0.1%
710000	5	< 0.1%
700000	8	< 0.1%

In [2]:

```
#check the shape of data
dataset.shape
```

Out[2]: (24000, 24)

In [3]:

```
data = dataset.sample(frac=0.95, random_state=786)
data_unseen = dataset.drop(data.index)

data.reset_index(inplace=True, drop=True)
data_unseen.reset_index(inplace=True, drop=True)
```

```
print('Data for Modeling: ' + str(data.shape))
print('Unseen Data For Predictions ' + str(data_unseen.shape))
```

```
Data for Modeling: (22800, 24)
Unseen Data For Predictions (1200, 24)
```

In [4]:
```
from pycaret.classification import *
```

In [5]:
```
exp_clf102 = setup(data = data, target = 'default', session_id=123,
                   normalize = True,
                   transformation = True,
                   ignore_low_variance = True,
                   remove_multicollinearity = True, multicollinearity_threshol
                   bin_numeric_features = ['LIMIT_BAL', 'AGE'],
                   group_features = [['BILL_AMT1', 'BILL_AMT2','BILL_AMT3', 'B
                                     ['PAY_AMT1','PAY_AMT2', 'PAY_AMT3', 'PAY_A
                   log_experiment = True, experiment_name = 'credit1')
```

	Description	Value
0	session_id	123
1	Target	default
2	Target Type	Binary
3	Label Encoded	0: 0, 1: 1
4	Original Data	(22800, 24)
5	Missing Values	False
6	Numeric Features	14
7	Categorical Features	9
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(15959, 117)
12	Transformed Test Set	(6841, 117)
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	True
20	Experiment Name	credit1
21	USI	40f5
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean

	Description	Value
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent
29	Normalize	True
30	Normalize Method	zscore
31	Transformation	True
32	Transformation Method	yeo-johnson
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	True
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	True
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	True
43	Multicollinearity Threshold	0.950000
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	True
51	Feature Selection	False
52	Features Selection Threshold	None
53	Feature Interaction	False
54	Feature Ratio	False
55	Interaction Threshold	None
56	Fix Imbalance	False
57	Fix Imbalance Method	SMOTE

In [ ]:

```
top3 = compare_models(n_select = 3)
```

In [ ]:

```
type(top3)
```

In [12]:
```
print(top3)
```

```
[RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=Tru
e,
                max_iter=None, normalize=False, random_state=123, solver='aut
o',
                tol=0.001), LinearDiscriminantAnalysis(n_components=None, prio
rs=None, shrinkage=None,
                           solver='svd', store_covariance=False, tol=0.0001),
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=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_iter_no_change=None, presort='deprecated',
                           random_state=123, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)]
```

In [13]:
```
dt = create_model('dt', fold = 5)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7284	0.6133	0.4026	0.3844	0.3933	0.2184	0.2185
1	0.7321	0.6182	0.4112	0.3926	0.4017	0.2292	0.2293
2	0.7303	0.6226	0.4269	0.3926	0.4091	0.2347	0.2351
3	0.7356	0.6176	0.4069	0.3978	0.4023	0.2325	0.2326
4	0.7364	0.6151	0.4003	0.3974	0.3989	0.2301	0.2301
Mean	0.7326	0.6173	0.4096	0.3930	0.4010	0.2290	0.2291
SD	0.0031	0.0032	0.0094	0.0048	0.0051	0.0056	0.0057

In [14]:
```
rf = create_model('rf', round = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.81	0.76	0.32	0.61	0.42	0.32	0.34
1	0.82	0.76	0.38	0.66	0.48	0.38	0.41
2	0.82	0.77	0.32	0.67	0.43	0.34	0.38
3	0.82	0.76	0.36	0.65	0.47	0.37	0.39
4	0.82	0.76	0.35	0.66	0.46	0.36	0.39
5	0.83	0.77	0.37	0.69	0.48	0.39	0.41
6	0.82	0.76	0.36	0.66	0.46	0.37	0.39
7	0.82	0.74	0.33	0.69	0.45	0.36	0.39
8	0.81	0.74	0.35	0.62	0.44	0.34	0.36
9	0.82	0.76	0.36	0.65	0.46	0.37	0.39
Mean	0.82	0.76	0.35	0.66	0.46	0.36	0.39
SD	0.01	0.01	0.02	0.03	0.02	0.02	0.02

In [15]:
```
tuned_rf = tune_model(rf, optimize = 'AUC')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7513	0.7702	0.6504	0.4522	0.5335	0.3713	0.3827
1	0.7575	0.7687	0.6562	0.4617	0.5420	0.3838	0.3948
2	0.7538	0.7701	0.6447	0.4555	0.5338	0.3732	0.3836
3	0.7393	0.7826	0.6762	0.4378	0.5315	0.3622	0.3786
4	0.7525	0.7948	0.6648	0.4549	0.5402	0.3789	0.3916
5	0.7575	0.7813	0.6590	0.4618	0.5431	0.3849	0.3962
6	0.7262	0.7670	0.6705	0.4209	0.5171	0.3397	0.3577
7	0.7513	0.7500	0.5989	0.4485	0.5129	0.3505	0.3571
8	0.7318	0.7432	0.6246	0.4233	0.5046	0.3300	0.3417
9	0.7254	0.7600	0.6236	0.4141	0.4977	0.3192	0.3318
Mean	0.7447	0.7688	0.6469	0.4431	0.5256	0.3594	0.3716
SD	0.0121	0.0145	0.0231	0.0169	0.0155	0.0222	0.0218

In [39]:

```
tuned_rf2 = tune_model(rf, optimize = 'Recall')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
1	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
2	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
3	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
4	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
5	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
6	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
7	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
8	0.2187	0.5000	1.0000	0.2187	0.3589	0.0000	0.0000
9	0.2182	0.5000	1.0000	0.2182	0.3582	0.0000	0.0000
Mean	0.2186	0.5000	1.0000	0.2186	0.3588	0.0000	0.0000
SD	0.0001	0.0000	0.0000	0.0001	0.0002	0.0000	0.0000

In [16]:

```
plot_model(tuned_rf, plot = 'parameter')
```

	Parameters
bootstrap	False
ccp_alpha	0.0
class_weight	balanced_subsample
criterion	gini
max_depth	4
max_features	sqrt

	Parameters
max_leaf_nodes	None
max_samples	None
min_impurity_decrease	0.0005
min_impurity_split	None
min_samples_leaf	3
min_samples_split	5
min_weight_fraction_leaf	0.0
n_estimators	260
n_jobs	-1
oob_score	False
random_state	123
verbose	0
warm_start	False

In [40]:

```
plot_model(tuned_rf2, plot = 'parameter')
```

	Parameters
bootstrap	False
ccp_alpha	0.0
class_weight	balanced
criterion	gini
max_depth	2
max_features	log2
max_leaf_nodes	None
max_samples	None
min_impurity_decrease	0.1
min_impurity_split	None
min_samples_leaf	3
min_samples_split	5
min_weight_fraction_leaf	0.0
n_estimators	10
n_jobs	-1
oob_score	False
random_state	123
verbose	0
warm_start	False

In [20]:

```
# lets create a simple decision tree model that we will use for ensembling
```

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

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7356	0.6225	0.4183	0.4000	0.4090	0.2388	0.2389
1	0.7393	0.6346	0.4413	0.4107	0.4254	0.2571	0.2574
2	0.7343	0.6120	0.3926	0.3926	0.3926	0.2225	0.2225
3	0.7268	0.6175	0.4212	0.3858	0.4027	0.2261	0.2265
4	0.7393	0.6261	0.4183	0.4067	0.4124	0.2450	0.2450
5	0.7256	0.5987	0.3754	0.3732	0.3743	0.1985	0.1985
6	0.7312	0.6155	0.4126	0.3913	0.4017	0.2285	0.2286
7	0.7431	0.6135	0.3840	0.4073	0.3953	0.2324	0.2325
8	0.7318	0.6204	0.4212	0.3941	0.4072	0.2342	0.2344
9	0.7354	0.6260	0.4339	0.4016	0.4171	0.2463	0.2466
Mean	0.7343	0.6187	0.4119	0.3963	0.4038	0.2329	0.2331
SD	0.0053	0.0093	0.0202	0.0108	0.0135	0.0152	0.0152

In [21]:

```
bagged_dt = ensemble_model(dt)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8064	0.7300	0.3238	0.6075	0.4224	0.3189	0.3417
1	0.8152	0.7289	0.3725	0.6311	0.4685	0.3655	0.3841
2	0.8102	0.7341	0.3209	0.6292	0.4250	0.3254	0.3519
3	0.8095	0.7394	0.3266	0.6230	0.4286	0.3274	0.3520
4	0.8051	0.7262	0.3152	0.6044	0.4143	0.3110	0.3348
5	0.8114	0.7368	0.3524	0.6212	0.4497	0.3462	0.3665
6	0.7964	0.7279	0.3095	0.5625	0.3993	0.2889	0.3076
7	0.8145	0.7235	0.3181	0.6568	0.4286	0.3335	0.3648
8	0.8008	0.7073	0.3095	0.5838	0.4045	0.2982	0.3198
9	0.8144	0.7223	0.3764	0.6238	0.4695	0.3653	0.3824
Mean	0.8084	0.7276	0.3325	0.6143	0.4310	0.3280	0.3506
SD	0.0059	0.0085	0.0240	0.0251	0.0231	0.0244	0.0239

In [22]:

```
# check the parameters of bagged_dt
print(bagged_dt)
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                        class_weight=None,
                                                        criterion='gini',
                                                        max_depth=None,
                                                        max_features=None,
                                                        max_leaf_nodes=None,
                                                        min_impurity_decrease=
0.0,

                                                        min_impurity_split=Non
e,
```

```
                                                        min_samples_leaf=1,
                                                        min_samples_split=2,
                                                        min_weight_fraction_le
af=0.0,
                                                        presort='deprecated',
                                                        random_state=123,
                                                        splitter='best'),
                              bootstrap=True, bootstrap_features=False, max_features=1.0,
                              max_samples=1.0, n_estimators=10, n_jobs=None,
                              oob_score=False, random_state=123, verbose=0,
                              warm_start=False)
```

In [23]:

```
boosted_dt = ensemble_model(dt, method = 'Boosting')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7563	0.6397	0.3610	0.4315	0.3931	0.2422	0.2437
1	0.7701	0.6684	0.3668	0.4672	0.4109	0.2706	0.2737
2	0.7832	0.6625	0.3123	0.5070	0.3865	0.2638	0.2752
3	0.7694	0.6478	0.2894	0.4570	0.3544	0.2226	0.2312
4	0.7788	0.7193	0.4241	0.4933	0.4561	0.3183	0.3197
5	0.7569	0.6793	0.4011	0.4389	0.4192	0.2658	0.2663
6	0.7732	0.6685	0.3696	0.4760	0.4161	0.2781	0.2816
7	0.7826	0.6933	0.3152	0.5046	0.3880	0.2643	0.2751
8	0.7638	0.6917	0.3524	0.4489	0.3949	0.2507	0.2536
9	0.7925	0.6843	0.3448	0.5381	0.4203	0.3012	0.3123
Mean	0.7727	0.6755	0.3537	0.4762	0.4040	0.2678	0.2732
SD	0.0112	0.0222	0.0388	0.0324	0.0257	0.0261	0.0262

In [24]:

```
bagged_dt2 = ensemble_model(dt, n_estimators=50)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8120	0.7460	0.3496	0.6256	0.4485	0.3460	0.3673
1	0.8195	0.7563	0.3840	0.6473	0.4820	0.3813	0.4004
2	0.8102	0.7550	0.3352	0.6223	0.4358	0.3337	0.3569
3	0.8120	0.7462	0.3467	0.6269	0.4465	0.3444	0.3663
4	0.8189	0.7587	0.3553	0.6596	0.4618	0.3645	0.3898
5	0.8195	0.7638	0.3610	0.6597	0.4667	0.3691	0.3934
6	0.7995	0.7505	0.3295	0.5721	0.4182	0.3075	0.3246
7	0.8214	0.7359	0.3467	0.6798	0.4592	0.3655	0.3952
8	0.8108	0.7445	0.3324	0.6270	0.4345	0.3335	0.3577
9	0.8226	0.7588	0.3879	0.6585	0.4882	0.3895	0.4095
Mean	0.8147	0.7516	0.3528	0.6379	0.4541	0.3535	0.3761
SD	0.0067	0.0080	0.0191	0.0286	0.0207	0.0237	0.0247

In [25]:

```
# train individual models to blend
```

```
lightgbm = create_model('lightgbm', verbose = False)
dt = create_model('dt', verbose = False)
lr = create_model('lr', verbose = False)# train individual models to blend
```

In [26]:

```
# blend individual models
blend_soft = blend_models(estimator_list = [lightgbm, dt, lr], method = 'soft
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7776	0.7415	0.3868	0.4891	0.4320	0.2960	0.2992
1	0.8045	0.7477	0.3668	0.5845	0.4507	0.3393	0.3529
2	0.8033	0.7394	0.3381	0.5871	0.4291	0.3205	0.3383
3	0.7920	0.7533	0.3467	0.5378	0.4216	0.3019	0.3127
4	0.7976	0.7724	0.3381	0.5619	0.4222	0.3086	0.3232
5	0.7675	0.7464	0.3438	0.4580	0.3928	0.2526	0.2566
6	0.7726	0.7364	0.3754	0.4746	0.4192	0.2802	0.2832
7	0.7763	0.7358	0.3352	0.4835	0.3959	0.2642	0.2708
8	0.7945	0.7368	0.3381	0.5488	0.4184	0.3021	0.3152
9	0.7981	0.7376	0.3678	0.5565	0.4429	0.3258	0.3363
Mean	0.7884	0.7447	0.3537	0.5282	0.4225	0.2991	0.3088
SD	0.0129	0.0107	0.0178	0.0452	0.0172	0.0258	0.0296

In [27]:

```
# blend individual models
blend_hard = blend_models(estimator_list = [lightgbm, dt, lr], method = 'hard
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8139	0.0000	0.2980	0.6667	0.4119	0.3200	0.3567
1	0.8233	0.0000	0.3181	0.7161	0.4405	0.3535	0.3947
2	0.8102	0.0000	0.2693	0.6620	0.3829	0.2935	0.3352
3	0.8114	0.0000	0.2779	0.6644	0.3919	0.3019	0.3422
4	0.8177	0.0000	0.2865	0.7042	0.4073	0.3215	0.3671
5	0.8133	0.0000	0.2837	0.6735	0.3992	0.3097	0.3505
6	0.8158	0.0000	0.3152	0.6667	0.4280	0.3346	0.3680
7	0.8108	0.0000	0.2607	0.6741	0.3760	0.2893	0.3349
8	0.8139	0.0000	0.2693	0.6912	0.3876	0.3020	0.3489
9	0.8132	0.0000	0.2960	0.6603	0.4087	0.3164	0.3524
Mean	0.8143	0.0000	0.2875	0.6779	0.4034	0.3143	0.3551
SD	0.0037	0.0000	0.0183	0.0183	0.0191	0.0185	0.0171

In [28]:

```
# blend top3 models from compare_models
blender_top3 = blend_models(top3)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8189	0.0000	0.3582	0.6579	0.4638	0.3661	0.3906
1	0.8315	0.0000	0.3897	0.7083	0.5028	0.4114	0.4381
2	0.8214	0.0000	0.3410	0.6839	0.4551	0.3623	0.3937
3	0.8296	0.0000	0.3754	0.7081	0.4906	0.3997	0.4288
4	0.8221	0.0000	0.3582	0.6757	0.4682	0.3732	0.4003
5	0.8321	0.0000	0.4011	0.7035	0.5109	0.4186	0.4427
6	0.8221	0.0000	0.3610	0.6738	0.4701	0.3747	0.4011
7	0.8277	0.0000	0.3553	0.7126	0.4742	0.3847	0.4180
8	0.8202	0.0000	0.3553	0.6667	0.4636	0.3674	0.3936
9	0.8245	0.0000	0.3793	0.6735	0.4853	0.3893	0.4126
Mean	0.8250	0.0000	0.3674	0.6864	0.4785	0.3847	0.4120
SD	0.0046	0.0000	0.0174	0.0189	0.0174	0.0187	0.0182

In [29]:

```
print(blender_top3.estimators_)
```

```
[RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=Tru
e,
                 max_iter=None, normalize=False, random_state=123, solver='aut
o',
                 tol=0.001), LinearDiscriminantAnalysis(n_components=None, prio
rs=None, shrinkage=None,
                            solver='svd', store_covariance=False, tol=0.0001),
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                            learning_rate=0.1, loss='deviance', max_depth=3,
                            max_features=None, max_leaf_nodes=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_iter_no_change=None, presort='deprecated',
                            random_state=123, subsample=1.0, tol=0.0001,
                            validation_fraction=0.1, verbose=0,
                            warm_start=False)]
```

In [30]:

```
stack_soft = stack_models(top3)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7870	0.7003	0.0802	0.5957	0.1414	0.0944	0.1589
1	0.7888	0.7038	0.0888	0.6200	0.1554	0.1064	0.1746
2	0.7857	0.6954	0.0860	0.5660	0.1493	0.0972	0.1558
3	0.7838	0.7146	0.0860	0.5357	0.1481	0.0933	0.1463
4	0.7957	0.7281	0.1117	0.7091	0.1931	0.1420	0.2241
5	0.7895	0.7057	0.0831	0.6444	0.1472	0.1024	0.1755
6	0.7926	0.7204	0.0917	0.6957	0.1620	0.1171	0.1988
7	0.7882	0.7138	0.0602	0.6774	0.1105	0.0776	0.1562
8	0.7926	0.7135	0.0946	0.6875	0.1662	0.1197	0.1997
9	0.7875	0.7018	0.0805	0.5957	0.1418	0.0948	0.1593

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
Mean	0.7891	0.7097	0.0863	0.6327	0.1515	0.1045	0.1749
SD	0.0034	0.0096	0.0123	0.0563	0.0200	0.0170	0.0238

In [31]:

```
xgboost = create_model('xgboost')
stack_soft2 = stack_models(top3, meta_model=xgboost)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8051	0.7575	0.3410	0.5950	0.4335	0.3262	0.3446
1	0.8195	0.7630	0.3926	0.6432	0.4875	0.3857	0.4031
2	0.8177	0.7569	0.3410	0.6611	0.4499	0.3537	0.3816
3	0.8208	0.7722	0.3868	0.6522	0.4856	0.3856	0.4049
4	0.8108	0.7650	0.3524	0.6181	0.4489	0.3449	0.3647
5	0.8152	0.7540	0.3553	0.6392	0.4567	0.3561	0.3784
6	0.8145	0.7554	0.3696	0.6293	0.4657	0.3625	0.3813
7	0.8183	0.7530	0.3295	0.6725	0.4423	0.3486	0.3803
8	0.8114	0.7438	0.3467	0.6237	0.4457	0.3430	0.3645
9	0.8157	0.7358	0.3649	0.6350	0.4635	0.3619	0.3821
Mean	0.8149	0.7557	0.3580	0.6369	0.4579	0.3568	0.3786
SD	0.0045	0.0098	0.0194	0.0212	0.0169	0.0175	0.0169

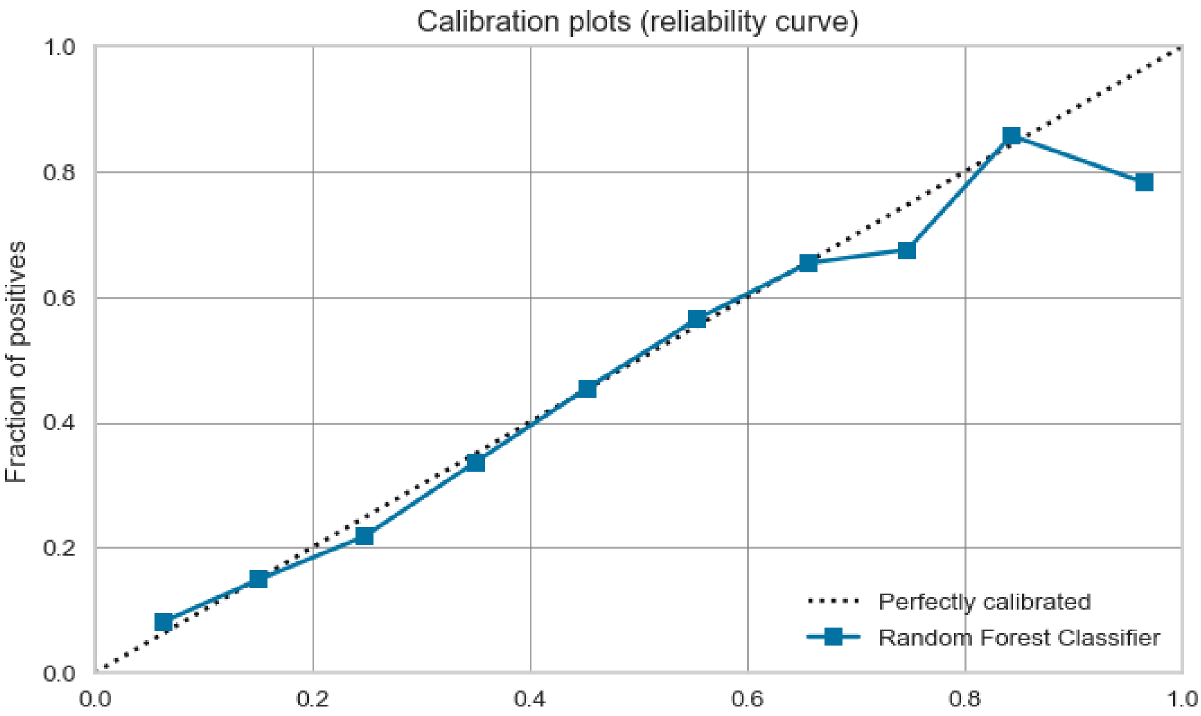
In [32]:

```
rf = create_model('rf')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8070	0.7557	0.3181	0.6133	0.4189	0.3168	0.3414
1	0.8221	0.7599	0.3754	0.6650	0.4799	0.3824	0.4052
2	0.8177	0.7652	0.3209	0.6747	0.4350	0.3422	0.3759
3	0.8189	0.7578	0.3639	0.6546	0.4678	0.3692	0.3923
4	0.8189	0.7572	0.3524	0.6613	0.4598	0.3630	0.3889
5	0.8258	0.7748	0.3668	0.6919	0.4794	0.3864	0.4145
6	0.8189	0.7593	0.3582	0.6579	0.4638	0.3661	0.3906
7	0.8221	0.7438	0.3324	0.6946	0.4496	0.3589	0.3936
8	0.8102	0.7433	0.3467	0.6173	0.4440	0.3403	0.3609
9	0.8188	0.7559	0.3592	0.6545	0.4638	0.3657	0.3896
Mean	0.8180	0.7573	0.3494	0.6585	0.4562	0.3591	0.3853
SD	0.0053	0.0087	0.0186	0.0255	0.0184	0.0198	0.0201

In [33]:

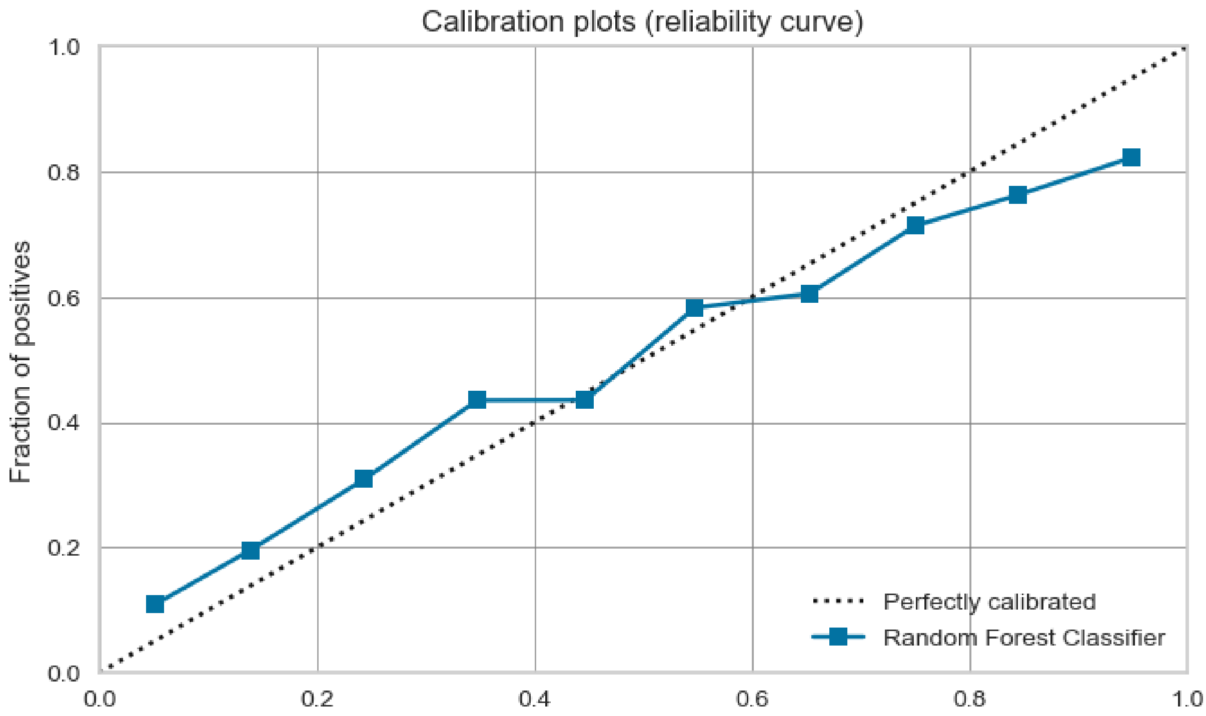
```
plot_model(rf, plot='calibration')
```



In [34]:
```
calibrated_rf = calibrate_model(rf)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8058	0.7653	0.3009	0.6140	0.4038	0.3037	0.3313
1	0.8315	0.7657	0.3725	0.7222	0.4915	0.4026	0.4343
2	0.8189	0.7693	0.3095	0.6923	0.4277	0.3383	0.3771
3	0.8258	0.7650	0.3668	0.6919	0.4794	0.3864	0.4145
4	0.8170	0.7708	0.3324	0.6629	0.4427	0.3474	0.3771
5	0.8252	0.7759	0.3610	0.6923	0.4746	0.3819	0.4111
6	0.8183	0.7689	0.3295	0.6725	0.4423	0.3486	0.3803
7	0.8271	0.7506	0.3352	0.7267	0.4588	0.3721	0.4117
8	0.8152	0.7524	0.3295	0.6534	0.4381	0.3416	0.3703
9	0.8201	0.7642	0.3420	0.6723	0.4533	0.3590	0.3885
Mean	0.8205	0.7648	0.3379	0.6801	0.4512	0.3582	0.3896
SD	0.0069	0.0074	0.0222	0.0314	0.0248	0.0272	0.0278

In [35]:
```
plot_model(calibrated_rf, plot='calibration')
```



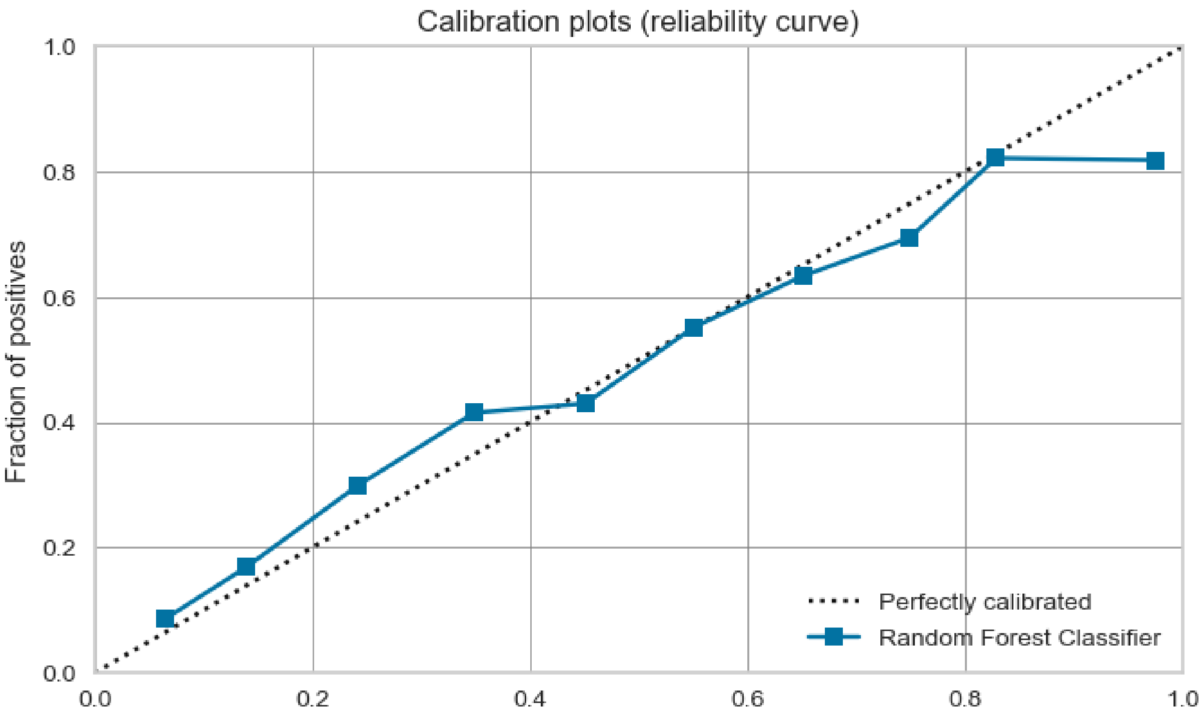
In [36]:

```
calibrated_rf_isotonic = calibrate_model(rf, method = 'isotonic')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8064	0.7634	0.3123	0.6124	0.4137	0.3120	0.3375
1	0.8308	0.7648	0.3983	0.6985	0.5073	0.4143	0.4381
2	0.8202	0.7686	0.3209	0.6914	0.4384	0.3480	0.3844
3	0.8221	0.7623	0.3696	0.6684	0.4760	0.3794	0.4035
4	0.8145	0.7704	0.3266	0.6514	0.4351	0.3385	0.3674
5	0.8289	0.7753	0.3840	0.6979	0.4954	0.4027	0.4288
6	0.8202	0.7688	0.3524	0.6685	0.4615	0.3658	0.3928
7	0.8246	0.7497	0.3496	0.6971	0.4656	0.3743	0.4062
8	0.8133	0.7532	0.3352	0.6393	0.4398	0.3407	0.3663
9	0.8169	0.7626	0.3649	0.6414	0.4652	0.3647	0.3858
Mean	0.8198	0.7639	0.3514	0.6666	0.4598	0.3640	0.3911
SD	0.0070	0.0073	0.0267	0.0285	0.0273	0.0292	0.0285

In [37]:

```
plot_model(calibrated_rf_isotonic, plot='calibration')
```


Calibration plots (reliability curve)
Fraction of positives
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated
Random Forest Classifier

```
In [ ]:
```

```
In [ ]:
```