In [1]:

```
#https://thinkingneuron.com/german-credit-risk-classification-case-study-in-python/
```

In [2]:

```
import pandas as pd
import numpy as np

path='C:/Users/HANNAH_SOPHIE/Desktop/MISCELANEOUS/MISCELANEOUS/ml_quantitative_Python/ml_quantitative/CreditRiskData.csv'

CRDF=pd.read_csv(path, encoding='latin')
print('Shape before deleting duplicate values:', CRDF.shape)

# Removing duplicate rows if any
CRDF=CRDF.drop_duplicates()
print('Shape After deleting duplicate values:', CRDF.shape)

CRDF.head(10)
```

```
Shape before deleting duplicate values: (1000, 21)
Shape After deleting duplicate values: (1000, 21)
```

Out[2]:

	GoodCredit	checkingstatus	duration	history	purpose	amount	savings	employ	installment	status	...	residence	property	age	otherp
0	0	A11	6	A34	A43	1169	A65	A75	4	A93	...	4	A121	67	
1	1	A12	48	A32	A43	5951	A61	A73	2	A92	...	2	A121	22	
2	0	A14	12	A34	A46	2096	A61	A74	2	A93	...	3	A121	49	
3	0	A11	42	A32	A42	7882	A61	A74	2	A93	...	4	A122	45	
4	1	A11	24	A33	A40	4870	A61	A73	3	A93	...	4	A124	53	
5	0	A14	36	A32	A46	9055	A65	A73	2	A93	...	4	A124	35	
6	0	A14	24	A32	A42	2835	A63	A75	3	A93	...	4	A122	53	
7	0	A12	36	A32	A41	6948	A61	A73	2	A93	...	2	A123	35	
8	0	A14	12	A32	A43	3059	A64	A74	2	A91	...	4	A121	61	
9	1	A12	30	A34	A40	5234	A61	A71	4	A94	...	2	A123	28	

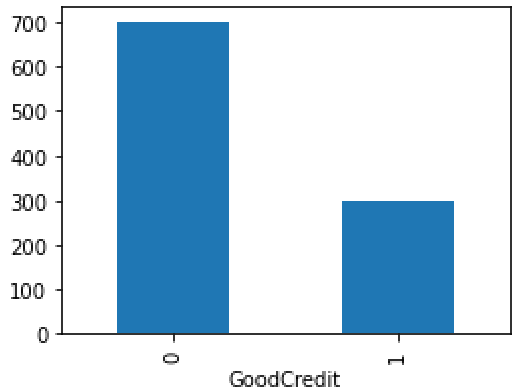
10 rows × 21 columns

In [3]:

```
%matplotlib inline
GroupedData=CRDF.groupby('GoodCredit').size()
GroupedData.plot(kind='bar', figsize=(4,3));
GroupedData
```

Out[3]:

```
GoodCredit
0    700
1    300
dtype: int64
```



In [4]:

```
CRDF.describe(include='all')
```

Out[4]:

	GoodCredit	checkingstatus	duration	history	purpose	amount	savings	employ	installment	status	...	residence	prop
count	1000.000000	1000	1000.000000	1000	1000	1000.000000	1000	1000	1000.000000	1000	...	1000.000000	
unique	NaN	4	NaN	5	10	NaN	5	5	NaN	4	...	NaN	
top	NaN	A14	NaN	A32	A43	NaN	A61	A73	NaN	A93	...	NaN	
freq	NaN	394	NaN	530	280	NaN	603	339	NaN	548	...	NaN	
mean	0.300000	NaN	20.903000	NaN	NaN	3271.258000	NaN	NaN	2.973000	NaN	...	2.845000	
std	0.458487	NaN	12.058814	NaN	NaN	2822.736876	NaN	NaN	1.118715	NaN	...	1.103718	
min	0.000000	NaN	4.000000	NaN	NaN	250.000000	NaN	NaN	1.000000	NaN	...	1.000000	
25%	0.000000	NaN	12.000000	NaN	NaN	1365.500000	NaN	NaN	2.000000	NaN	...	2.000000	
50%	0.000000	NaN	18.000000	NaN	NaN	2319.500000	NaN	NaN	3.000000	NaN	...	3.000000	
75%	1.000000	NaN	24.000000	NaN	NaN	3972.250000	NaN	NaN	4.000000	NaN	...	4.000000	

	GoodCredit	checkingstatus	duration	history	purpose	amount	savings	employ	installment	status	...	residence	prop
max	1.000000	NaN	72.000000	NaN	NaN	18424.000000	NaN	NaN	4.000000	NaN	...	4.000000	

11 rows × 21 columns

In [5]:
```
#Any NA in target?
#pd.isnull(CRDF["GoodCredit"])

CRDF["GoodCredit"].isnull().sum()
```

Out[5]: 0

In [ ]:
```
#from pandas_profiling import ProfileReport
#ProfileReport(CRDF, title="CRDF Profiling Report")
```

In [ ]:
```

```

In [6]:
```
from sklearn.model_selection import *
```

In [7]:
```
CRDF_train, CRDF_test = train_test_split(CRDF, test_size=0.2)
```

In [8]:
```
CRDF_train.describe()
```

Out[8]:

	GoodCredit	duration	amount	installment	residence	age	cards	liable
count	800.000000	800.000000	800.000000	800.00000	800.000000	800.000000	800.000000	800.000000
mean	0.298750	20.975000	3332.690000	2.95250	2.845000	35.513750	1.410000	1.158750
std	0.457996	12.143567	2868.318289	1.13108	1.102268	11.410305	0.578635	0.365671
min	0.000000	4.000000	250.000000	1.00000	1.000000	19.000000	1.000000	1.000000
25%	0.000000	12.000000	1385.000000	2.00000	2.000000	27.000000	1.000000	1.000000

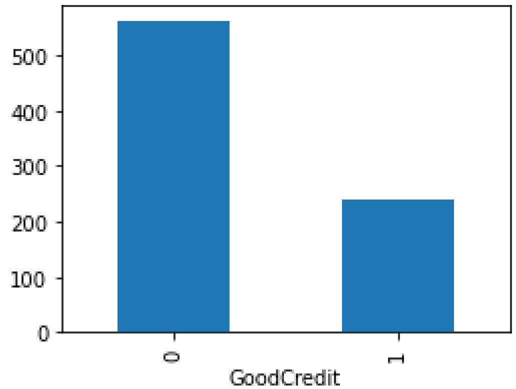
	GoodCredit	duration	amount	installment	residence	age	cards	liable
50%	0.000000	18.000000	2347.000000	3.00000	3.000000	33.000000	1.000000	1.000000
75%	1.000000	24.000000	3976.750000	4.00000	4.000000	41.250000	2.000000	1.000000
max	1.000000	60.000000	18424.000000	4.00000	4.000000	75.000000	4.000000	2.000000

In [9]:

```
%matplotlib inline
GroupedData=CRDF_train.groupby('GoodCredit').size()
GroupedData.plot(kind='bar', figsize=(4,3));
GroupedData
```

Out[9]:

```
GoodCredit
0    561
1    239
dtype: int64
```



In [10]:

```
CRDF_test.describe()
```

Out[10]:

	GoodCredit	duration	amount	installment	residence	age	cards	liable
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	0.305000	20.615000	3025.530000	3.055000	2.845000	35.675000	1.395000	1.140000
std	0.461563	11.739072	2624.974247	1.066613	1.112277	11.262486	0.575003	0.347858
min	0.000000	4.000000	338.000000	1.000000	1.000000	20.000000	1.000000	1.000000
25%	0.000000	12.000000	1246.000000	2.000000	2.000000	26.750000	1.000000	1.000000

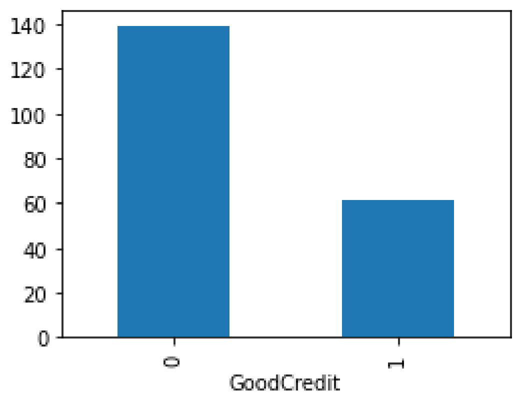
	GoodCredit	duration	amount	installment	residence	age	cards	liable
50%	0.000000	18.000000	2066.000000	3.000000	3.000000	33.000000	1.000000	1.000000
75%	1.000000	24.000000	3773.250000	4.000000	4.000000	43.250000	2.000000	1.000000
max	1.000000	72.000000	14318.000000	4.000000	4.000000	68.000000	4.000000	2.000000

In [11]:

```
%matplotlib inline
GroupedData=CRDF_test.groupby('GoodCredit').size()
GroupedData.plot(kind='bar', figsize=(4,3));
GroupedData
```

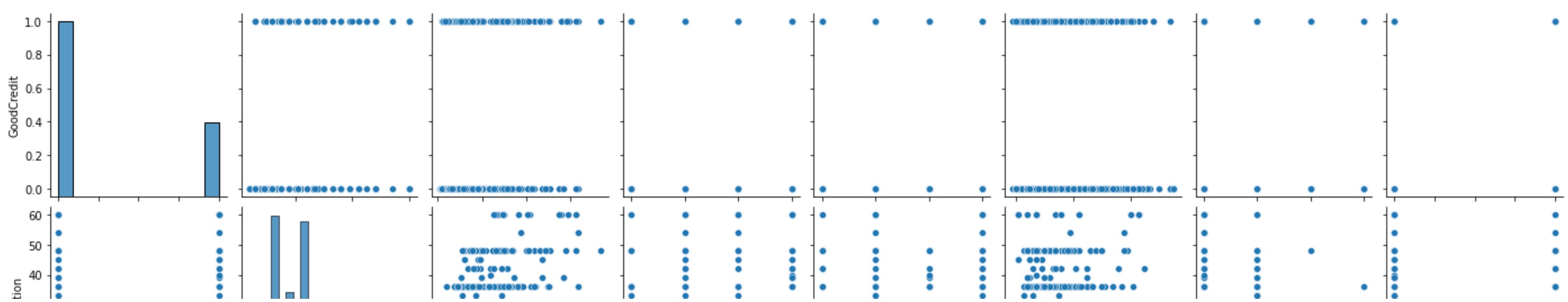
Out[11]:

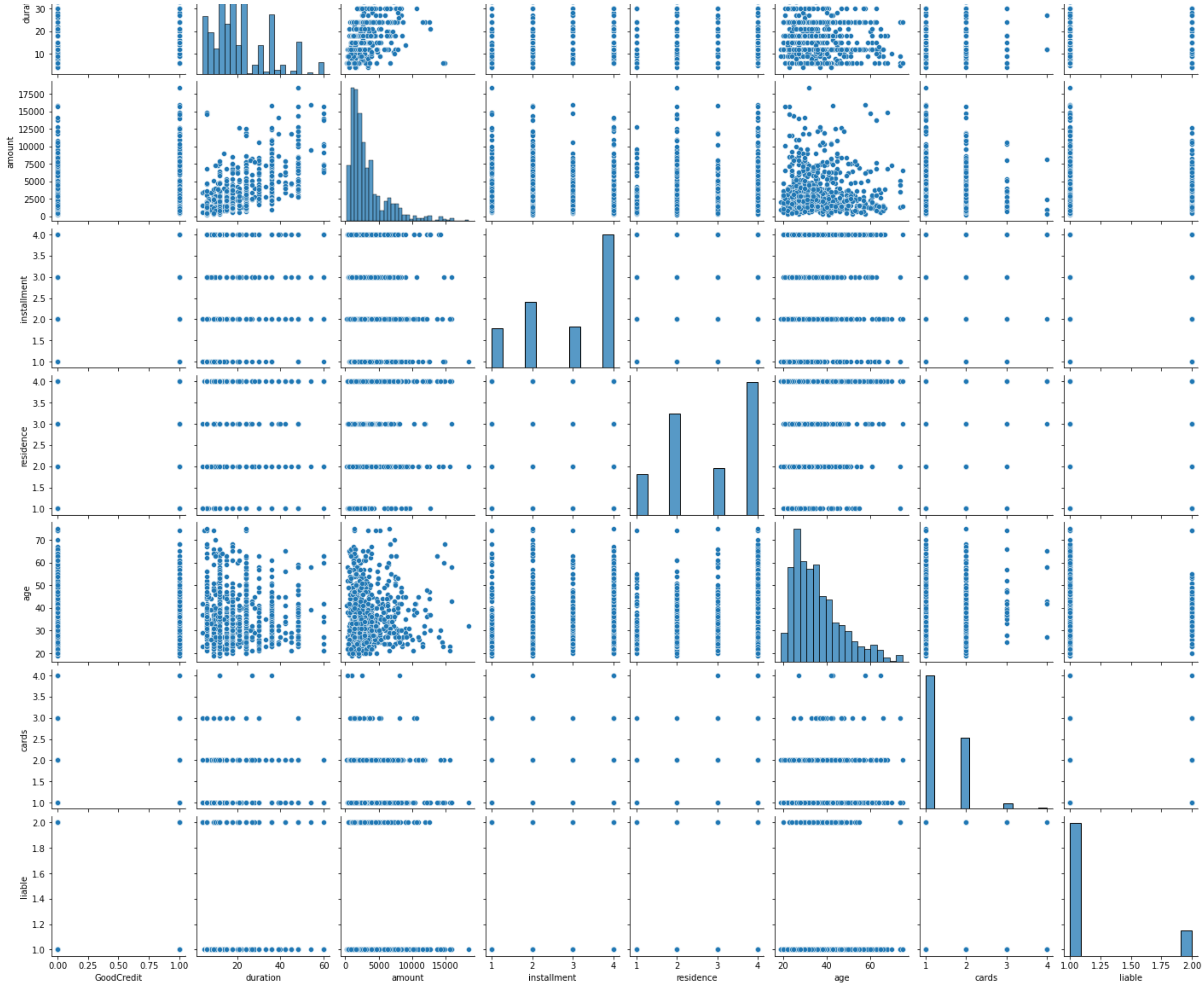
```
GoodCredit
0    139
1     61
dtype: int64
```



In [12]:

```
import seaborn as sn
sn.pairplot(CRDF_train);
```





durat
amount
installment
residence
age
cards
liable
GoodCredit
duration
amount
installment
residence
age
cards
liable

In [15]:
```
from pycaret.classification import *
```

In [16]:
```
pycaret.classification.models
```

Out[16]:
```
<function pycaret.classification.models(type: Union[str, NoneType] = None, internal: bool = False, raise_errors: bool = T
rue) -> pandas.core.frame.DataFrame>
```

In [20]:
```
from imblearn.under_sampling import RandomUnderSampler
RUS = RandomUnderSampler()
```

In [21]:
```
clf=setup(CRDF_train,  target = 'GoodCredit',
          fold_strategy='kfold',fold=10,
          fix_imbalance=True, fix_imbalance_method = RUS,
          session_id=123)
```

	Description	Value
0	session_id	123
1	Target	GoodCredit
2	Target Type	Binary
3	Label Encoded	0: 0, 1: 1
4	Original Data	(800, 21)
5	Missing Values	False
6	Numeric Features	3
7	Categorical Features	17
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(559, 68)
12	Transformed Test Set	(241, 68)
13	Shuffle Train-Test	True

	Description	Value
14	Stratify Train-Test	False
15	Fold Generator	KFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	5938
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

	Description	Value
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	Feature Selection Method	classic
53	Features Selection Threshold	None
54	Feature Interaction	False
55	Feature Ratio	False
56	Interaction Threshold	None
57	Fix Imbalance	True
58	Fix Imbalance Method	RandomUnderSampler

In [22]:

```
clf_fits=compare_models(include = ['lr','dt','svm','rf','xgboost'], sort='AUC')
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.7103	0.7803	0.7446	0.5310	0.6113	0.3905	0.4125	0.0930
lr	Logistic Regression	0.7084	0.7671	0.7213	0.5236	0.6011	0.3787	0.3956	0.0270

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.6870	0.7592	0.7227	0.5057	0.5862	0.3489	0.3695	0.1320
dt	Decision Tree Classifier	0.6440	0.6483	0.6524	0.4565	0.5278	0.2611	0.2775	0.0100
svm	SVM - Linear Kernel	0.4095	0.0000	0.6967	0.2470	0.3507	0.0107	0.0206	0.0080

In [23]:

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

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6429	0.7398	0.7778	0.4667	0.5833	0.3035	0.3341
1	0.7143	0.8166	0.6923	0.4286	0.5294	0.3402	0.3604
2	0.6071	0.7389	0.7143	0.3571	0.4762	0.2143	0.2474
3	0.7143	0.8977	0.9167	0.4231	0.5789	0.4043	0.4737
4	0.6250	0.7259	0.7222	0.4483	0.5532	0.2594	0.2815
5	0.7143	0.8146	0.8500	0.5667	0.6800	0.4400	0.4697
6	0.5536	0.6170	0.4545	0.4348	0.4444	0.0716	0.0717
7	0.7321	0.8717	0.6250	0.7143	0.6667	0.4444	0.4472
8	0.7321	0.8009	0.7647	0.5417	0.6341	0.4324	0.4484
9	0.7636	0.8692	0.8824	0.5769	0.6977	0.5172	0.5488
Mean	0.6799	0.7892	0.7400	0.4958	0.5844	0.3427	0.3683
SD	0.0645	0.0807	0.1279	0.0986	0.0816	0.1267	0.1334

In [24]:

```
rf_CRDF
```

Out[24]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       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=123, verbose=0,
                       warm_start=False)
```

In [25]:

```
tuned_rf_CRDF = tune_model(rf_CRDF)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6607	0.7602	0.7222	0.4815	0.5778	0.3127	0.3307
1	0.6786	0.7835	0.7692	0.4000	0.5263	0.3180	0.3570
2	0.6607	0.7772	0.7857	0.4074	0.5366	0.3091	0.3508
3	0.6607	0.8807	0.9167	0.3793	0.5366	0.3350	0.4168
4	0.6786	0.6681	0.7778	0.5000	0.6087	0.3571	0.3824
5	0.7321	0.7861	0.8000	0.5926	0.6809	0.4588	0.4741
6	0.5893	0.6417	0.4091	0.4737	0.4390	0.1178	0.1186
7	0.7679	0.8529	0.7083	0.7391	0.7234	0.5236	0.5239
8	0.7321	0.8175	0.7059	0.5455	0.6154	0.4150	0.4232
9	0.7636	0.8390	0.8235	0.5833	0.6829	0.5031	0.5222
Mean	0.6924	0.7807	0.7418	0.5102	0.5928	0.3650	0.3900
SD	0.0529	0.0724	0.1257	0.1039	0.0826	0.1121	0.1114

In [26]:

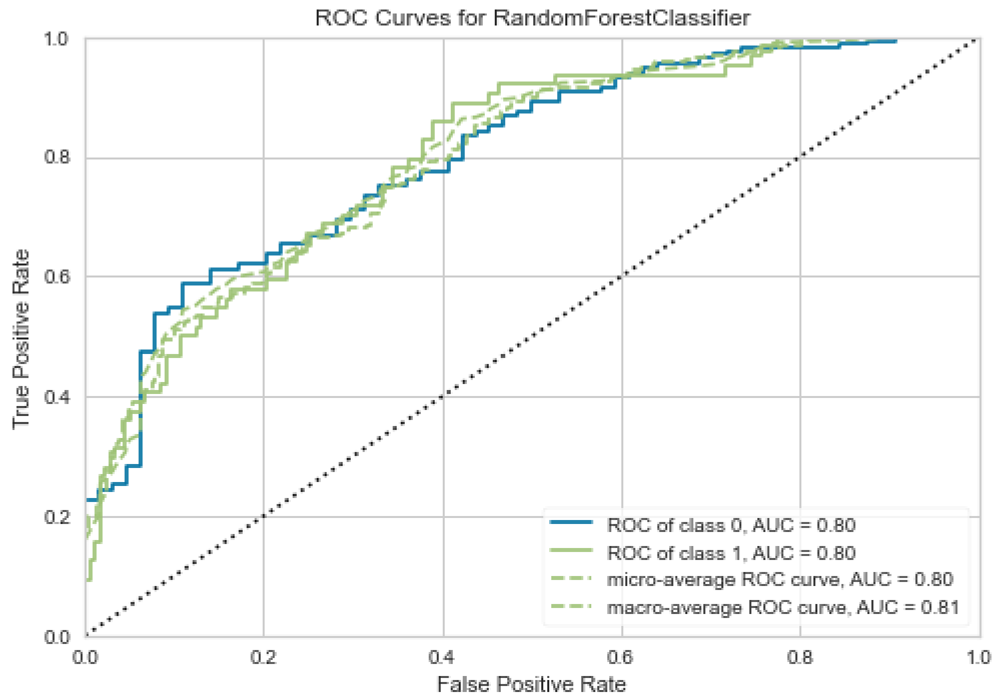
```
tuned_rf_CRDF
```

Out[26]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                       class_weight='balanced_subsample', criterion='entropy',
                       max_depth=4, max_features='log2', max_leaf_nodes=None,
                       max_samples=None, min_impurity_decrease=0.0002,
                       min_impurity_split=None, min_samples_leaf=5,
                       min_samples_split=9, min_weight_fraction_leaf=0.0,
                       n_estimators=130, n_jobs=-1, oob_score=False,
                       random_state=123, verbose=0, warm_start=False)
```

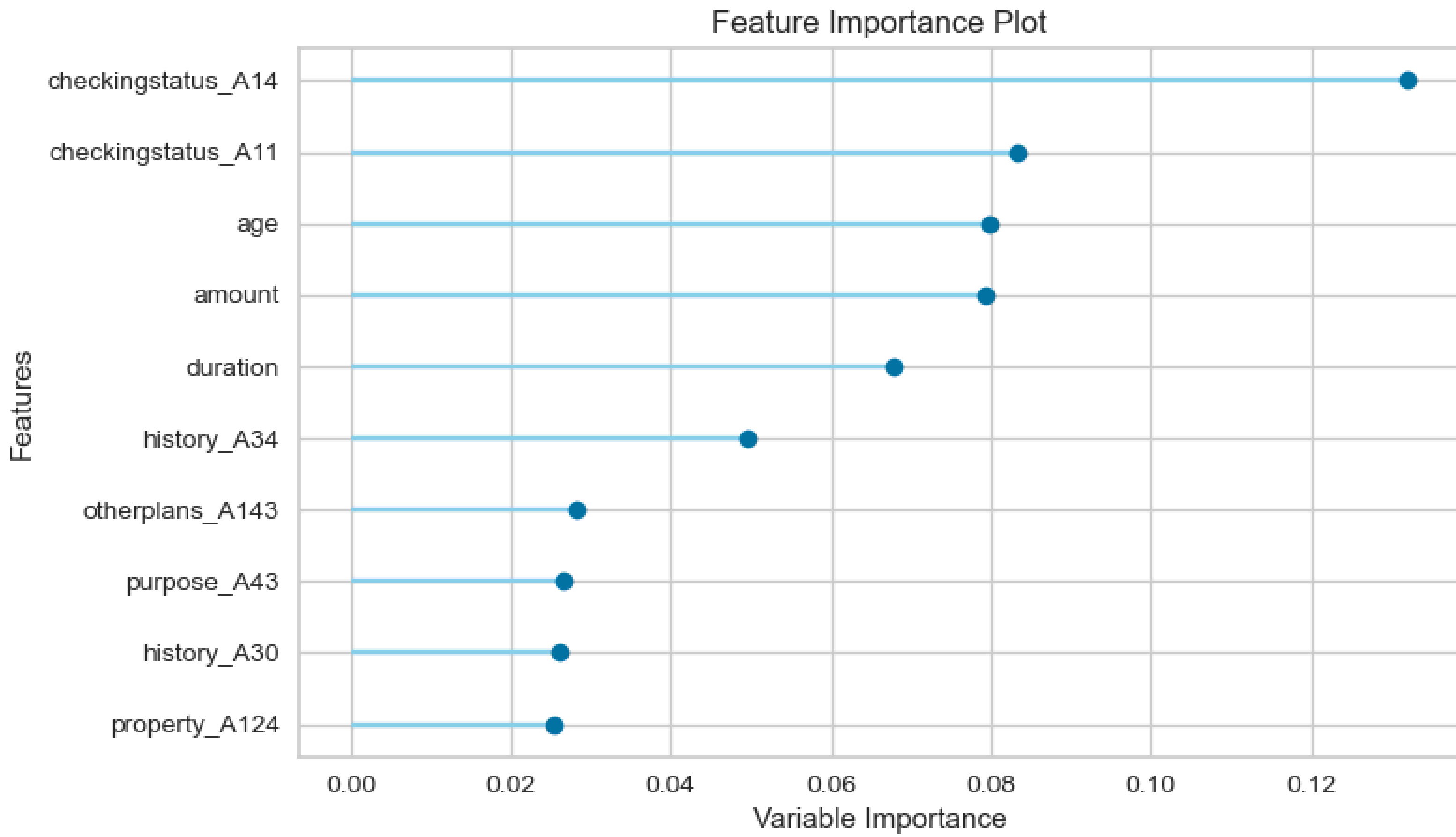
In [45]:

```
plot_model(tuned_rf_CRDF)
```



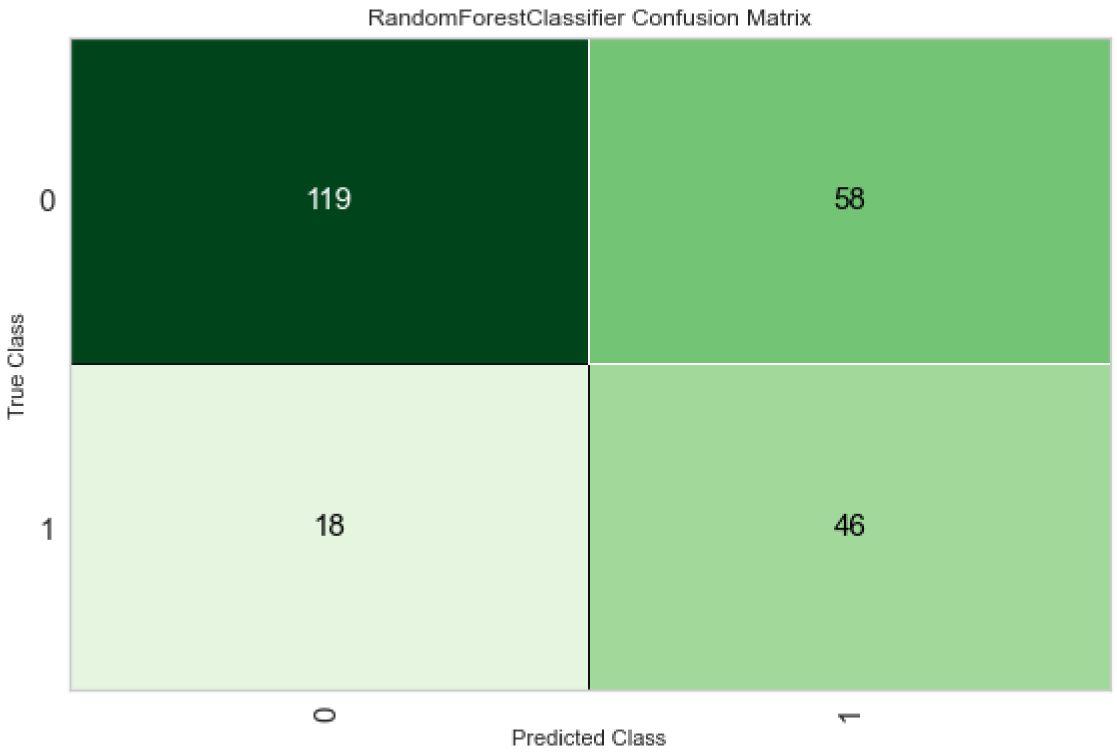
In [32]:

```
plot_model(tuned_rf_CRDF, plot='feature')
```



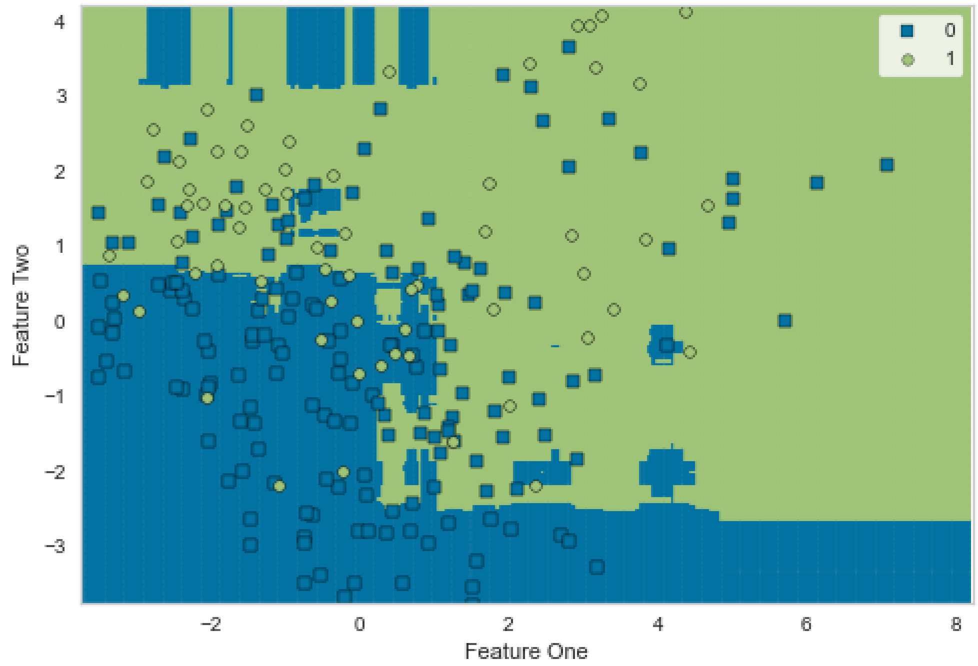
In [46]:

```
plot_model(tuned_rf_CRDF,plot='confusion_matrix')
```



In [47]:

```
plot_model(tuned_rf_CRDF, plot = 'boundary')
```

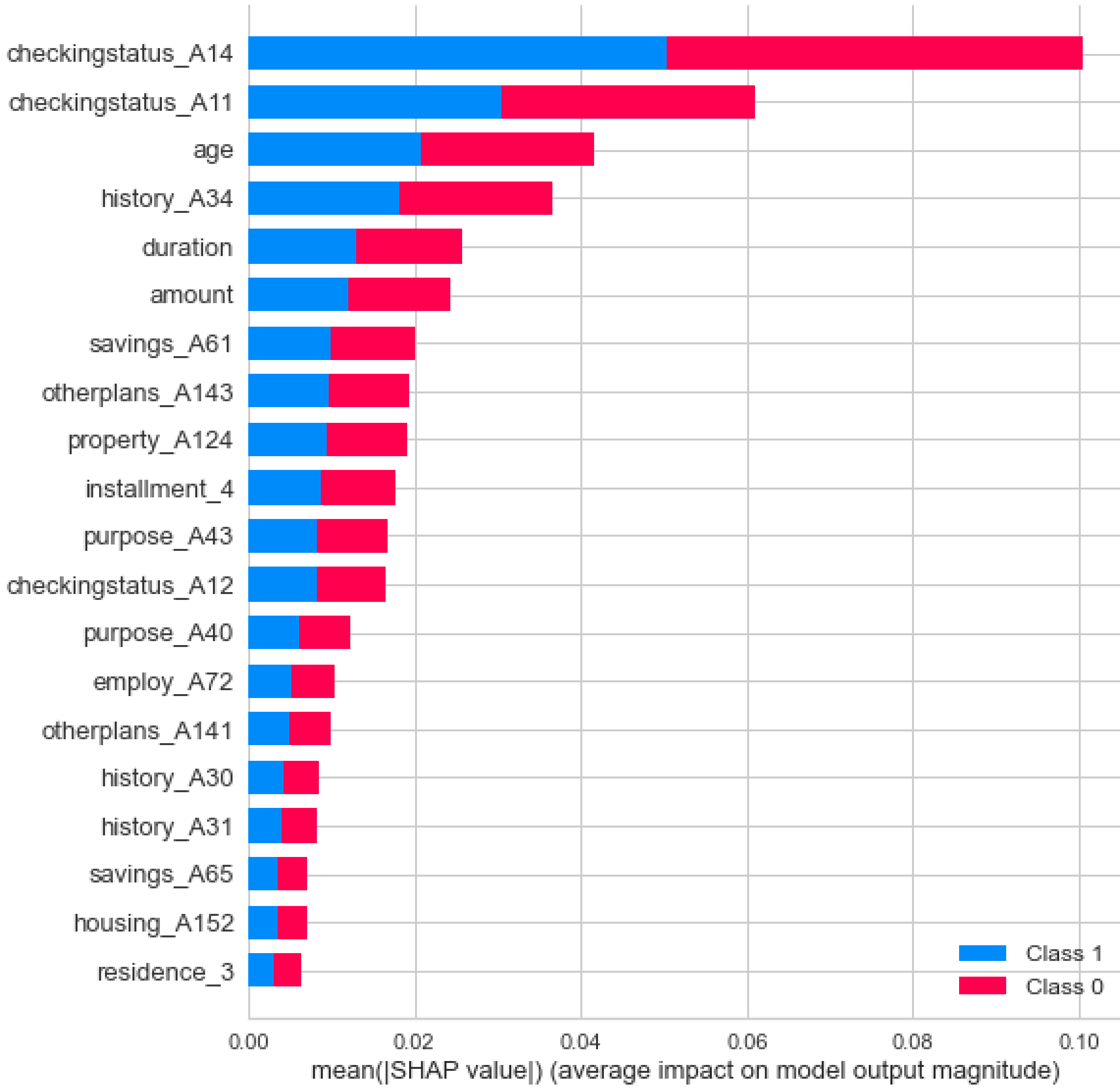


In [48]:

```
eval_rf_CRDF = evaluate_model(tuned_rf_CRDF)
```

In [42]:

```
interpret_model(tuned_rf_CRDF)
```

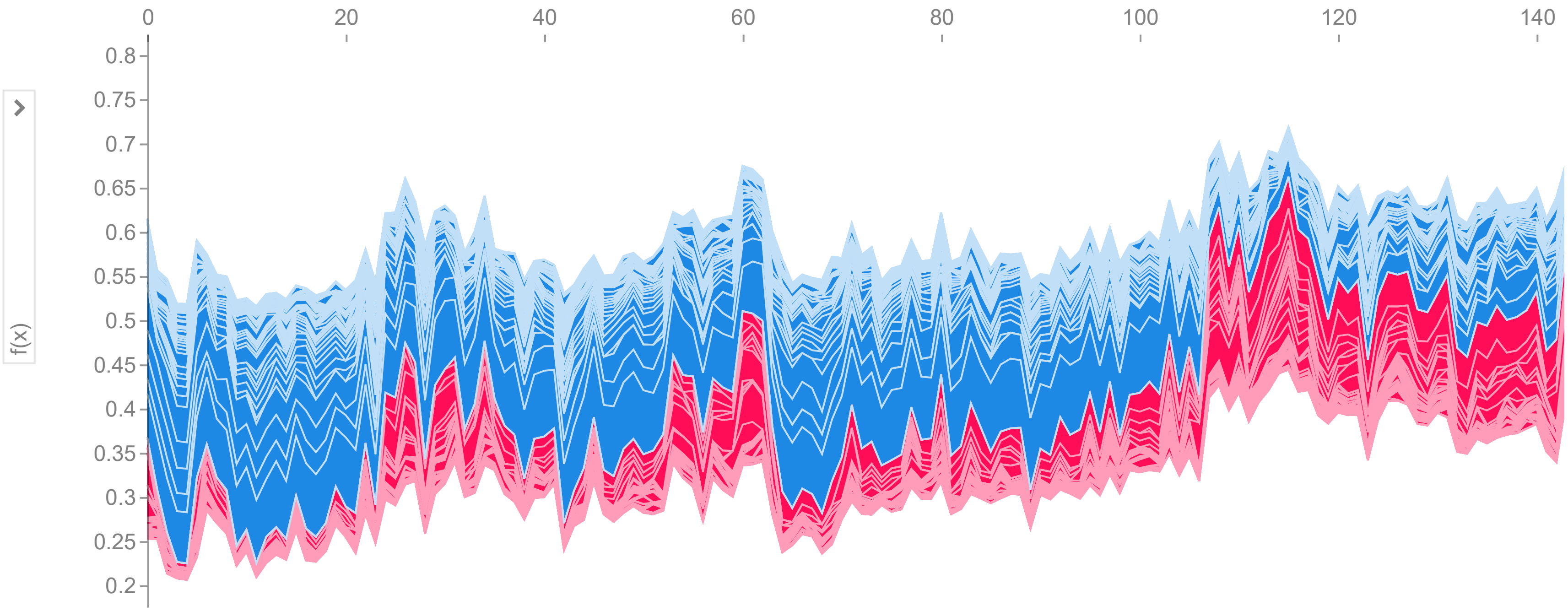


In [50]:

```
interpret_model(tuned_rf_CRDF,plot = 'reason')
```

Out[50]:

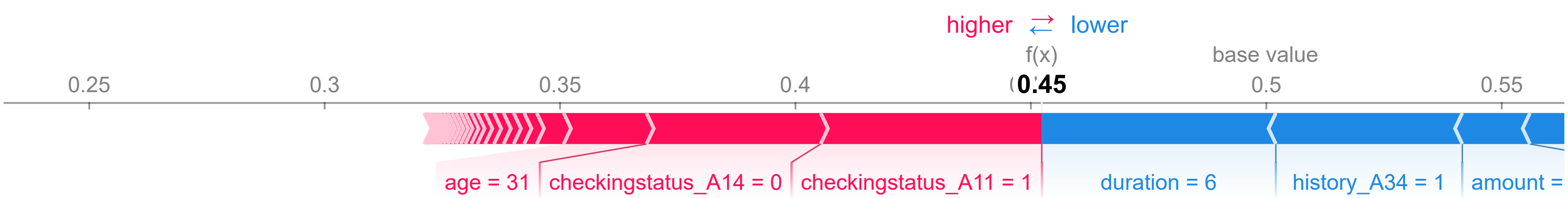
sample order by similarity



In [53]:

```
interpret_model(tuned_rf_CRDF,plot = 'reason', observation=55) #chose an arbitrary observation for local contribution and
```

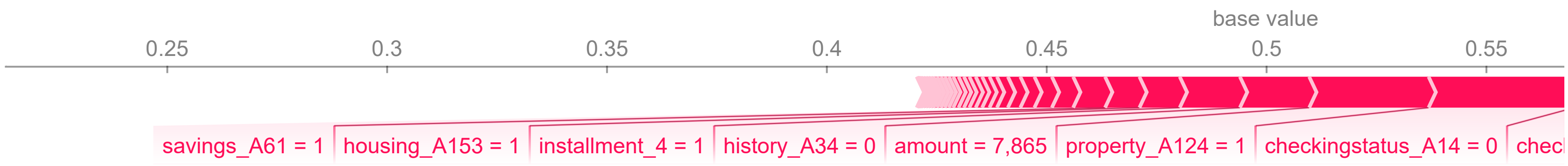
Out[53]:



In [71]:

```
interpret_model(tuned_rf_CRDF,plot = 'reason', observation=150) #chose an arbitrary observation for local contribution ar
```

Out[71]:



In [72]:
```
predict_model(tuned_rf_CRDF)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Random Forest Classifier	0.6846	0.8030	0.7188	0.4423	0.5476	0.3260	0.3487

Out[72]:

	duration	amount	age	checkingstatus_A11	checkingstatus_A12	checkingstatus_A13	checkingstatus_A14	history_A30	history_A31	histor
0	15.0	1300.0	45.0	0.0	0.0	0.0	1.0	0.0	0.0	
1	18.0	1961.0	23.0	0.0	0.0	1.0	0.0	0.0	0.0	
2	48.0	6143.0	58.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	21.0	2767.0	61.0	0.0	1.0	0.0	0.0	0.0	0.0	
4	6.0	518.0	29.0	0.0	0.0	0.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
236	15.0	2728.0	35.0	0.0	1.0	0.0	0.0	0.0	0.0	
237	9.0	1313.0	20.0	0.0	0.0	0.0	1.0	0.0	0.0	
238	14.0	8978.0	45.0	1.0	0.0	0.0	0.0	0.0	0.0	
239	24.0	6403.0	33.0	0.0	1.0	0.0	0.0	0.0	0.0	
240	24.0	3757.0	62.0	0.0	0.0	0.0	1.0	0.0	0.0	

241 rows × 71 columns

In [ ]:
```
#final_et_bos = finalize_model(tuned_et_bos)
```

In [ ]:

```
#print(final_et_bos)
```

In [73]:

```
rf_CRED_train_pred=predict_model(tuned_rf_CRDF,data=CRDF_train)
rf_CRED_test_pred=predict_model(tuned_rf_CRDF,data=CRDF_test)
```

In [74]:

```
rf_CRED_train_pred.head()
```

Out[74]:

	GoodCredit	checkingstatus	duration	history	purpose	amount	savings	employ	installment	status	...	age	otherplans	housing	car
506	0	A13	15	A34	A41	2360	A63	A73	2	A93	...	36	A143	A152	
420	0	A14	15	A32	A40	3186	A64	A74	2	A92	...	20	A143	A151	
542	1	A11	30	A32	A42	6350	A65	A75	4	A93	...	31	A143	A152	
412	1	A14	12	A34	A49	2292	A61	A71	4	A93	...	42	A142	A152	
520	0	A14	24	A34	A45	5507	A61	A75	3	A93	...	44	A143	A153	

5 rows × 23 columns

In [77]:

```
rf_CRED_test_pred.head()
```

Out[77]:

	GoodCredit	checkingstatus	duration	history	purpose	amount	savings	employ	installment	status	...	age	otherplans	housing	car
129	1	A11	12	A34	A40	3499	A61	A73	3	A92	...	29	A143	A152	
81	0	A14	15	A32	A43	1213	A63	A75	4	A93	...	47	A142	A152	
836	0	A14	12	A32	A43	886	A65	A73	4	A92	...	21	A143	A152	
375	1	A11	48	A31	A49	7685	A61	A74	2	A92	...	37	A143	A151	
377	0	A14	7	A33	A43	846	A65	A75	3	A93	...	36	A143	A153	

5 rows × 23 columns

In [78]:

```
from sklearn import metrics
```

In [89]:

```
[metrics.accuracy_score(rf_CRED_train_pred['GoodCredit'], rf_CRED_train_pred['Label']),
 metrics.precision_score(rf_CRED_train_pred['GoodCredit'], rf_CRED_train_pred['Label']),
 metrics.recall_score(rf_CRED_train_pred['GoodCredit'], rf_CRED_train_pred['Label']),
 metrics.f1_score(rf_CRED_train_pred['GoodCredit'], rf_CRED_train_pred['Label'])]
```

Out[89]:

```
[0.72625, 0.5277777777777778, 0.7949790794979079, 0.6343906510851419]
```

In [90]:

```
[metrics.accuracy_score(rf_CRED_test_pred['GoodCredit'], rf_CRED_test_pred['Label']),
 metrics.precision_score(rf_CRED_test_pred['GoodCredit'], rf_CRED_test_pred['Label']),
 metrics.recall_score(rf_CRED_test_pred['GoodCredit'], rf_CRED_test_pred['Label']),
 metrics.f1_score(rf_CRED_test_pred['GoodCredit'], rf_CRED_test_pred['Label'])]
```

Out[90]:

```
[0.725, 0.5348837209302325, 0.7540983606557377, 0.6258503401360543]
```

In [ ]:

```

```