CART (CLASSIFICATION AND REGRESSION TREES)

CLASSIFICATION

DATA

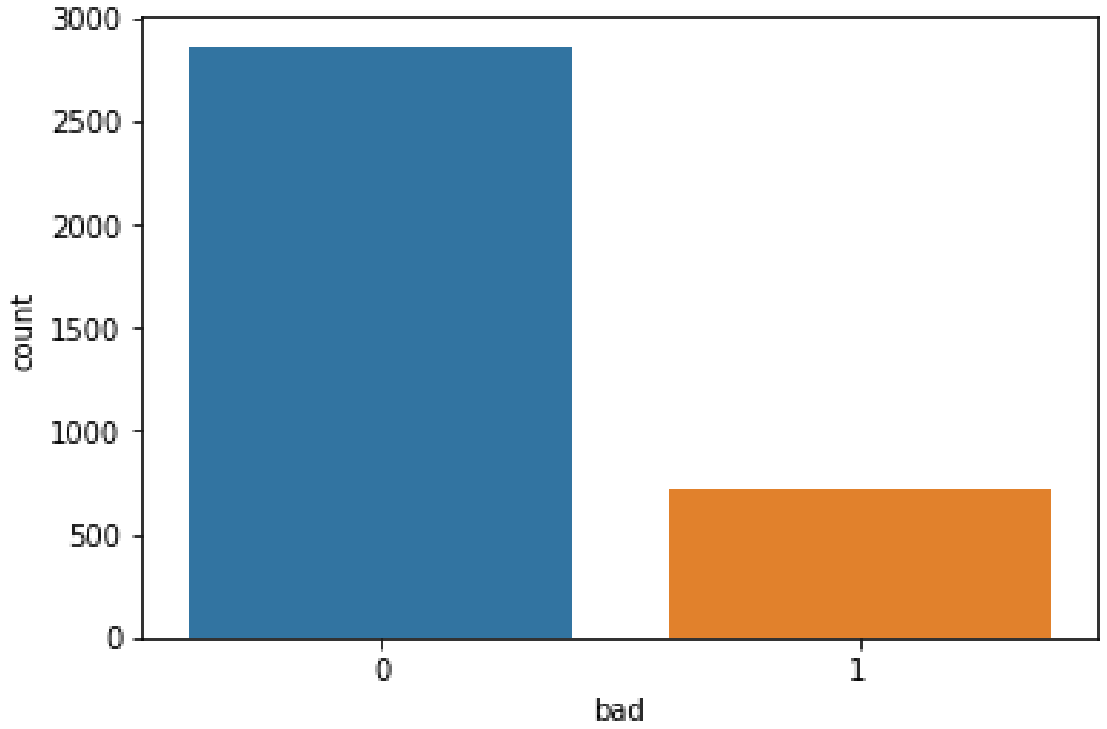
```
information = Classes.Information(data)
information.data_features()
```

```
-------------------------  DATA HEAD -------------------------

   bad   loan   mortdue     value   reason  job   yoj  derog  delinq   clage  \
0    0  81200   18834.0  108355.0  HomeImp  NaN  28.0    0.0     0.0  139.14
1    0  12600  103960.0  127384.0  DebtCon  NaN   2.0    0.0     0.0  129.02
2    0  18000   46865.0   61266.0  DebtCon  NaN   5.0    0.0     0.0  102.59
3    0  10300   57676.0   71027.0  DebtCon  NaN  19.0    0.0     0.0  157.52
4    0   9400   56508.0   78358.0  DebtCon  NaN  17.0    0.0     0.0  141.93

   ninq  clno  debtinc
0   0.0  14.0   34.042
1   0.0  25.0   34.479
2   2.0   9.0   26.354
3   1.0  11.0   33.992
4   0.0  11.0   32.327
```

DATA

```
-------------------------  DATA INFO -------------------------

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3576 entries, 0 to 3575
Data columns (total 13 columns):
bad         3576 non-null int64
loan        3576 non-null int64
mortdue     3262 non-null float64
value       3512 non-null float64
reason      3429 non-null object
job         3409 non-null object
yoj         3264 non-null float64
derog       3149 non-null float64
delinq      3225 non-null float64
clage       3397 non-null float64
ninq        3273 non-null float64
clno        3443 non-null float64
debtinc     2809 non-null float64
dtypes: float64(9), int64(2), object(2)
memory usage: 335.3+ KB
None

-------------------------  DATA SHAPE -------------------------

(3576, 13)
```

DATA



Veri seti içindeki krediyi ödeyip / ödeyememe durumunun yüzdesi :

```
Kredisini Ödeyenler 80.06152125279642
Kredisini Ödemeyenler 19.938478747203582
```

Sınıfların dağılımı dengesizdir.

DATA

```
data.groupby('bad').mean()
```

bad	loan	mortdue	value	yoj	derog	delinq	clage	ninq	clno	debtinc
0	18931.645127	75242.395117	102394.448489	9.031378	0.140732	0.238263	186.338950	1.032692	21.552536	33.179142
1	16915.708275	69029.488140	95308.460184	8.067533	0.716012	1.174888	153.497474	1.780089	21.323572	40.881416

- Kredisini ödeyenlerin(bad =0) , kredi talep miktarı ortalamasının(loan), kredisini ödemeyenlerin(bad =1) kredi talep miktarı ortalamasından yüksektir. Bu durumda kredisini ödeyebilenler yüksek kredi talebinde bulunmuştur diyebiliriz.
- Negatif rapor sayıları fazla olan bireylerin (derog) çoğu kredisini ödememiştir.
- Kredilerini ödeyen bireylerin borç/gelir oranı (debtinc) , kredilerini ödeyemeyen bireylerden daha düşüktür

DATA

```
data.groupby('job').mean()
```

job	bad	loan	mortdue	value	yoj	derog	delinq	clage	ninq	clno	debtinc
Mgr	0.232104	19084.598698	83964.704189	108464.106133	8.919318	0.320707	0.594203	174.285822	1.517564	23.097561	35.307687
Office	0.131810	18048.857645	68058.197973	94675.024670	8.103011	0.136905	0.445076	178.784840	0.936803	21.425795	34.158283
Other	0.232006	18006.918239	60064.432343	84251.694202	9.403457	0.313281	0.417183	174.026556	1.333836	19.572139	34.260072
ProfEx	0.166884	18750.717080	92690.971376	128851.319683	8.731349	0.203911	0.376871	196.769973	0.949728	24.503989	32.622049
Sales	0.348485	15251.515152	79856.864407	105960.969231	7.476667	0.450000	0.274194	202.301667	0.772727	24.272727	38.326064
Self	0.295652	27923.478261	102575.392523	147150.513274	7.210185	0.221239	0.551402	176.590526	1.404040	24.271930	36.824762

- Mesleklere göre kredi talep miktarları değişmektedir.(loan)

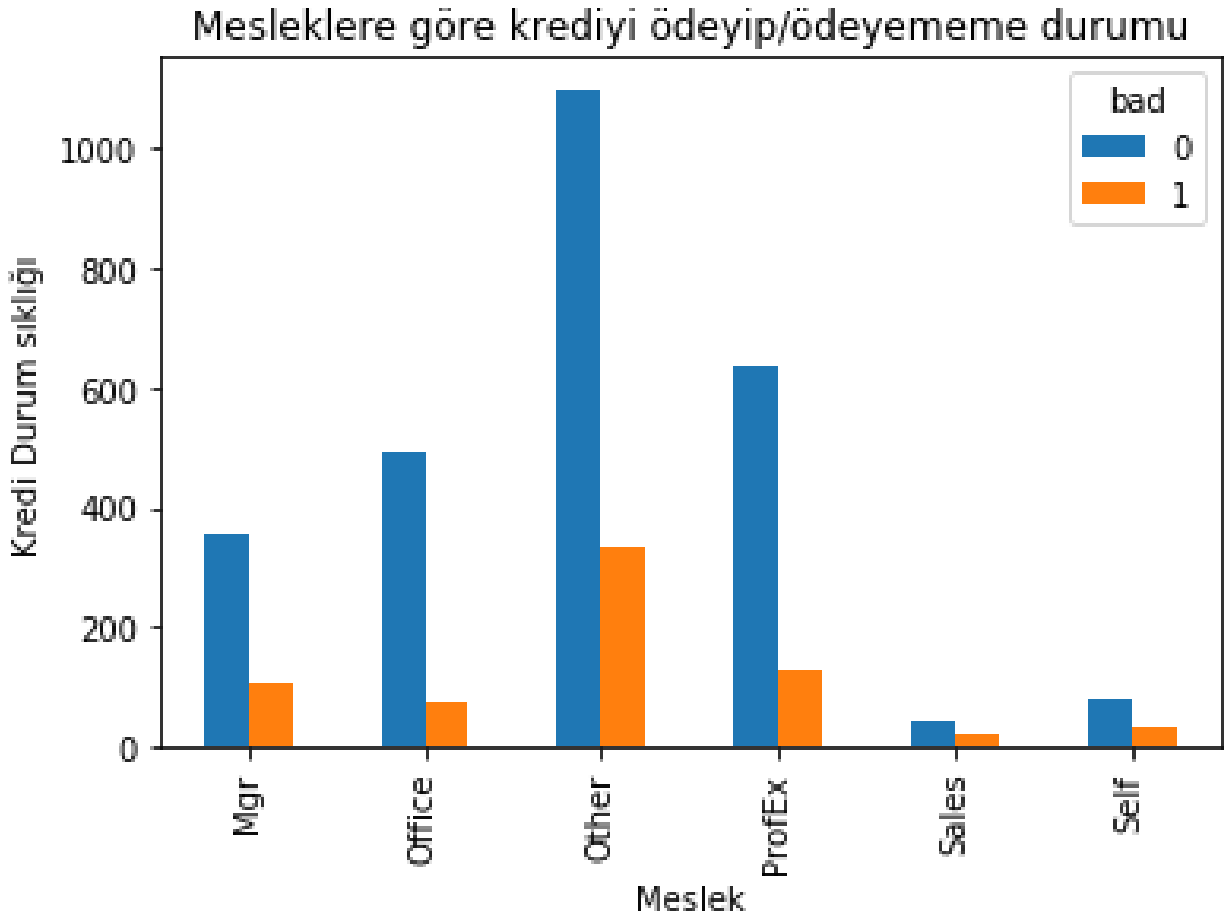
DATA

```
data.groupby('reason').mean()
```

reason	bad	loan	mortdue	value	yoj	derog	delinq	clage	ninq	clno	debtinc
DebtCon	0.185576	19868.705188	74483.615277	101611.714495	8.551114	0.261098	0.409427	176.176174	1.343708	22.287742	34.301599
HomeImp	0.230624	15892.911153	73308.909702	100007.497760	9.411429	0.245596	0.455852	185.208453	0.845361	19.905273	33.496014

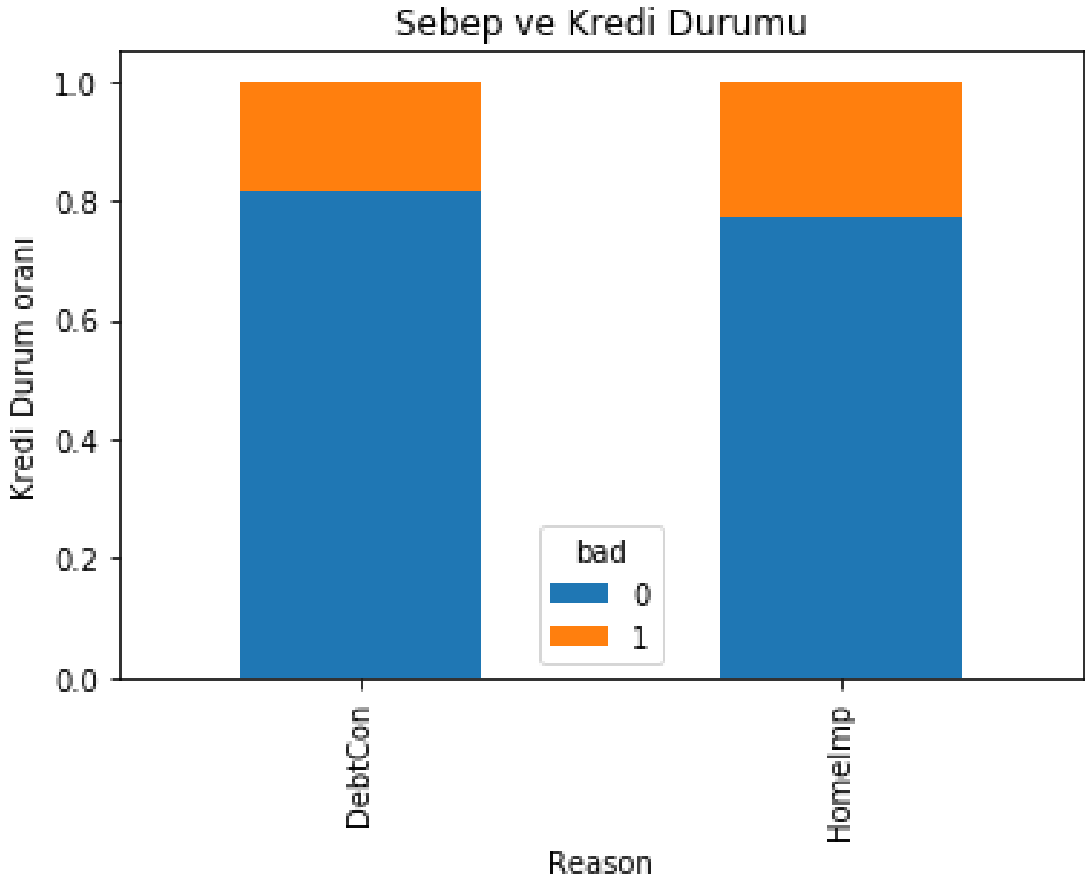
- Ev kredisi alanların kredi talep tutarları , borç kredisi alanların kredi talep tutarlarından düşüktür.(loan)

GÖRSELLEŞTİRME



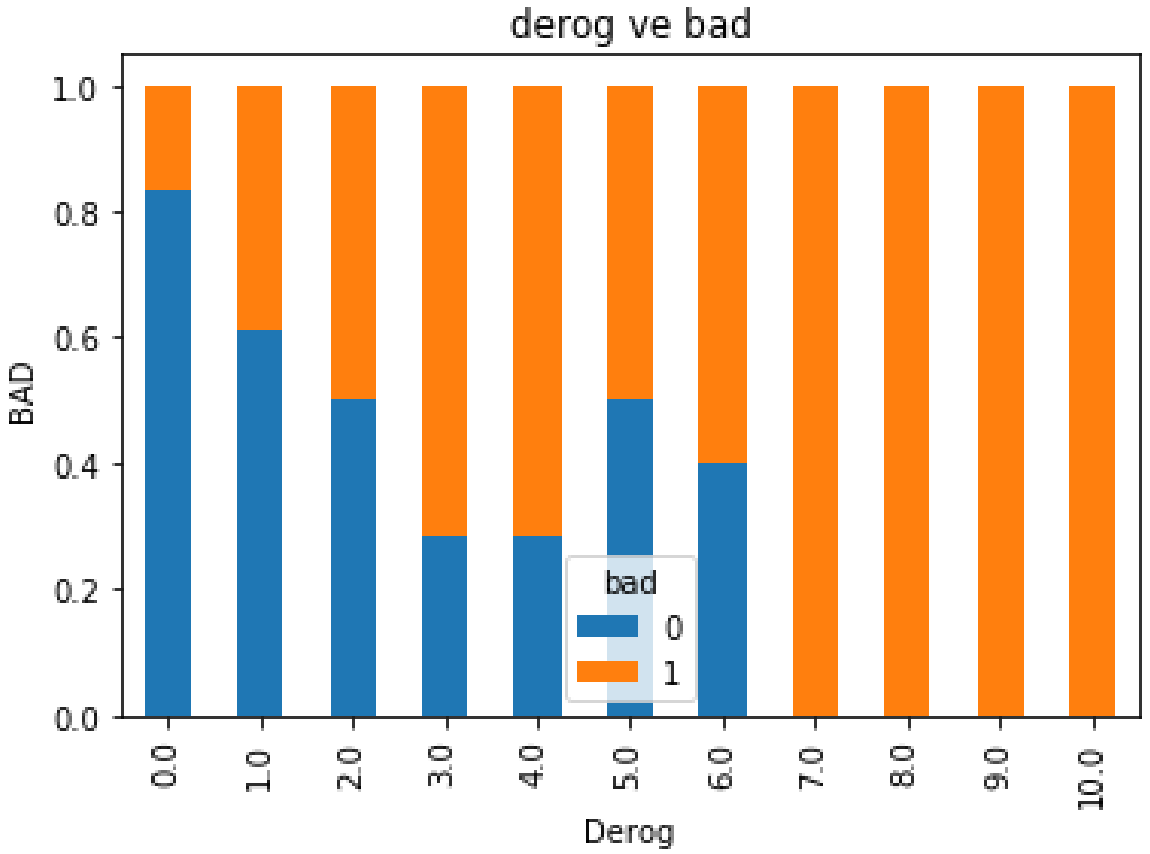
- Krediyi ödeyebilme(bad = 0) , büyük ölçüde mesleklere bağlı .Dolayısıyla iş unvanı, sonuç değişkeninin iyi bir öngörücüsü olabilir.

GÖRSELLEŞTİRME



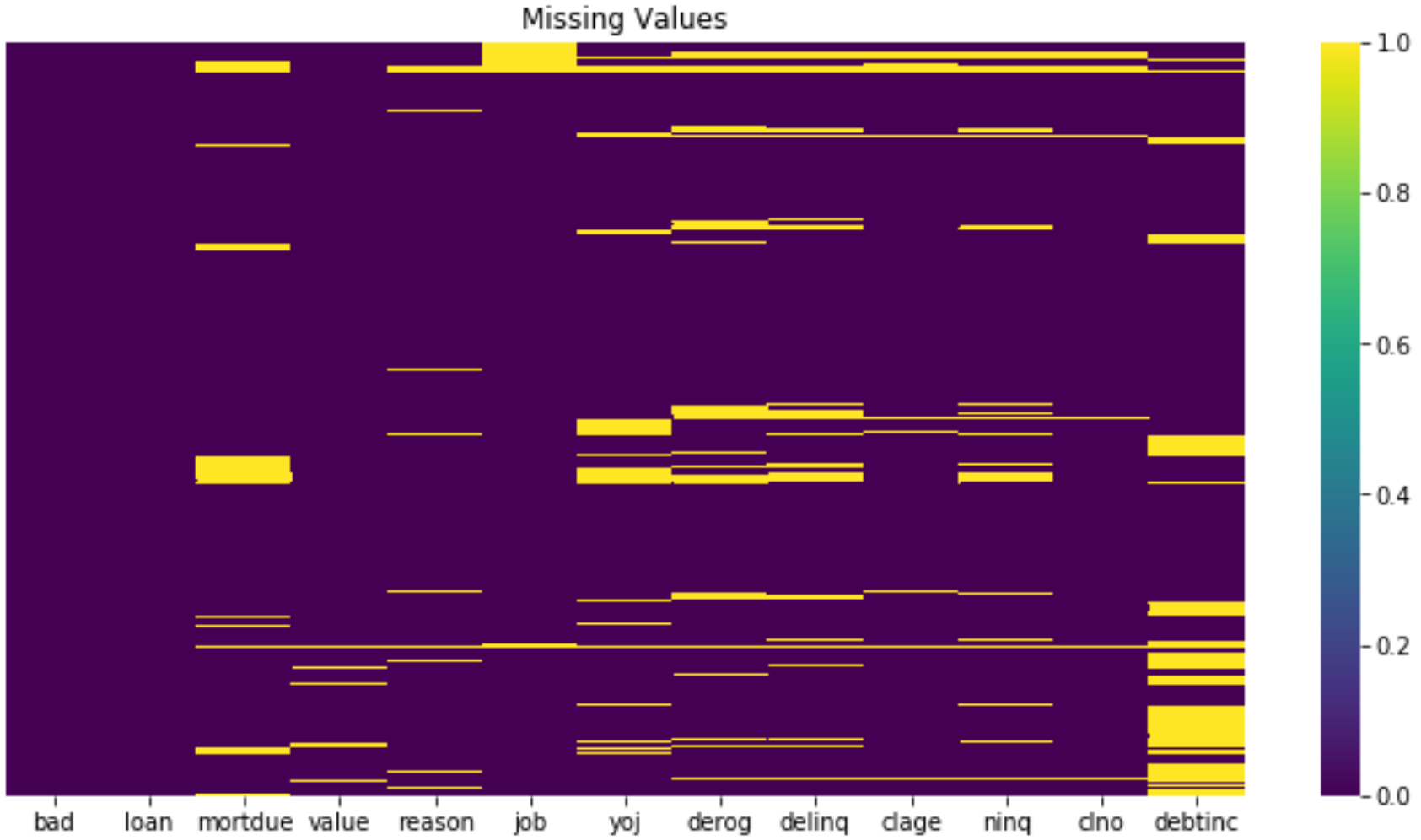
- Kredi talep sebebi , y değişkeni için güçlü bir yorumlayıcı görünmemektedir.

GÖRSELLEŞTİRME



- Negatif raporlara sahip bireylerin kredilerini ödeyip ödeyememelerinde iyi bir öngörücü olabilir

PREPROCESS



---------- Missing Values --

debtinc	767
derog	427
delinq	351
mortdue	314
yoj	312
ninq	303
clage	179
job	167
reason	147
clno	133
value	64
loan	0
bad	0

dtype: int64

PREPROCESS

```
p.drop('any')
```

```
Drop Öncesi Data Shape -->    (3576, 13)
Drop Sonrası Data Shape -->    (2018, 13)
------------  Missing Values  -------------

debtinc     0
clno        0
ninq        0
clage       0
delinq      0
derog       0
yoj         0
job         0
reason      0
value       0
mortdue     0
loan        0
bad         0
dtype: int64
```

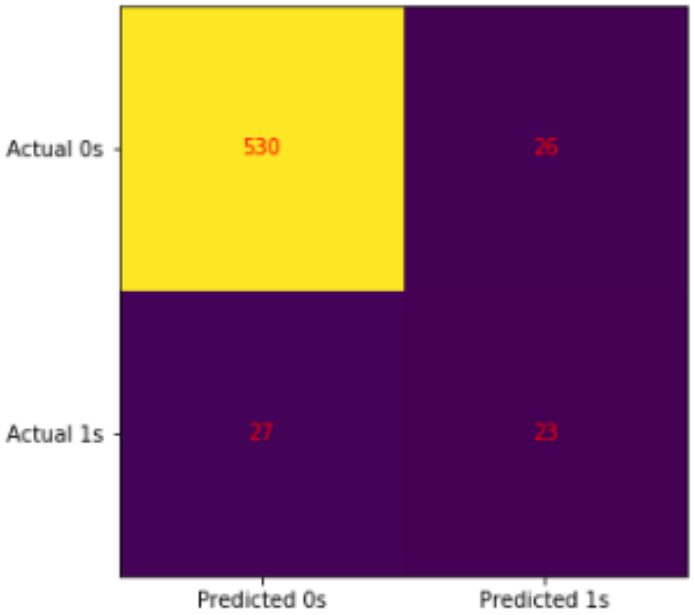
```
# DUMMIES

HomeImp = pd.get_dummies(data['reason'], drop_first =True)
jobs = pd.get_dummies(data['job'], drop_first =True)
data=pd.concat([data,HomeImp,jobs],axis=1)
data.head()
```

	bad	loan	mortdue	value	reason	job	yoj	derog	delinq	clage	ninq	clno	debtinc	HomeImp	Office	Other	ProfEx	Sales	Self
153	0	18200	94727.0	136877.0	DebtCon	Mgr	15.0	0.0	0.0	168.96	2.0	26.0	36.056	0	0	0	0	0	0
154	0	21700	79240.0	96784.0	DebtCon	Mgr	5.0	0.0	0.0	64.51	6.0	24.0	38.079	0	0	0	0	0	0
155	0	34100	241931.0	36486.0	DebtCon	Mgr	1.0	0.0	2.0	196.01	3.0	50.0	42.459	0	0	0	0	0	0
156	0	8400	62989.0	76718.0	HomeImp	Mgr	3.0	0.0	2.0	131.47	0.0	22.0	29.200	1	0	0	0	0	0
157	0	17400	25859.0	43684.0	DebtCon	Mgr	16.0	1.0	0.0	95.36	1.0	17.0	27.108	0	0	0	0	0	0

```
data.drop(['reason','job'],axis =1,inplace=True)
data.head()
```

DECISION TREE MODEL

```
g = Classes.GridSearchHelper()
a,m = g.DecisionTreeCls(X,y)
```

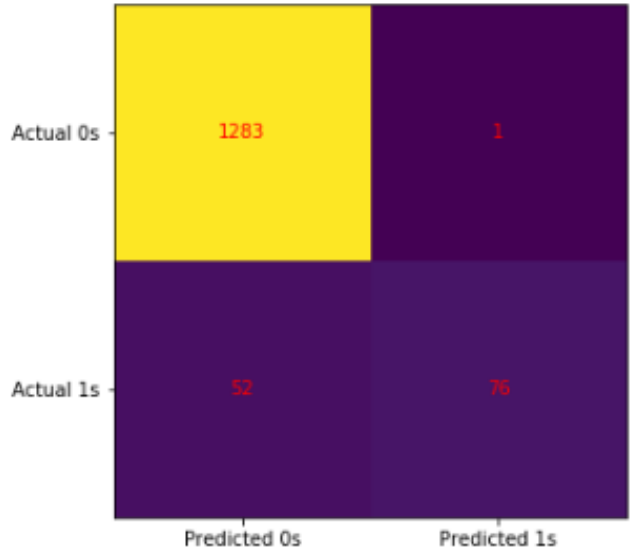



```
TEST Accuracy  Score :   0.9125412541254125
Classification Report  :
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       556
           1       0.47      0.46      0.46        50

    accuracy                           0.91       606
   macro avg       0.71      0.71      0.71       606
weighted avg       0.91      0.91      0.91       606
```

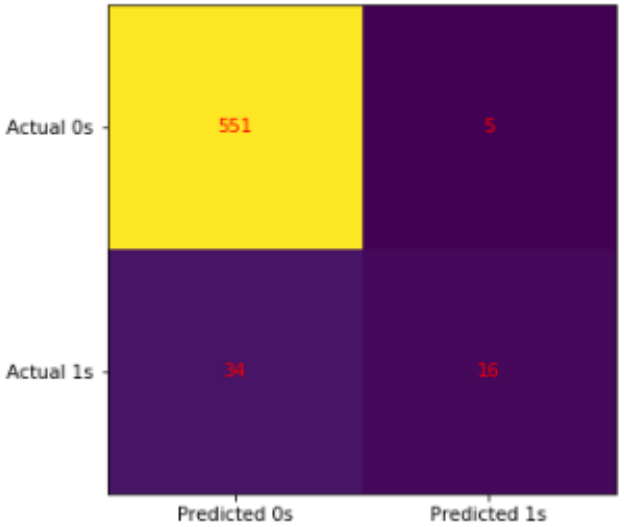
En iyi parametreler seçilerek oluşturulan yeni model

```
{'criterion': 'gini',
 'max_depth': 8,
 'min_samples_split': 2}
```

```
TRAIN Accuracy  Score :   0.9624645892351275
```




```
TEST Accuracy  Score :   0.9356435643564357
```



DECISION TREE MODEL

Classification Report :

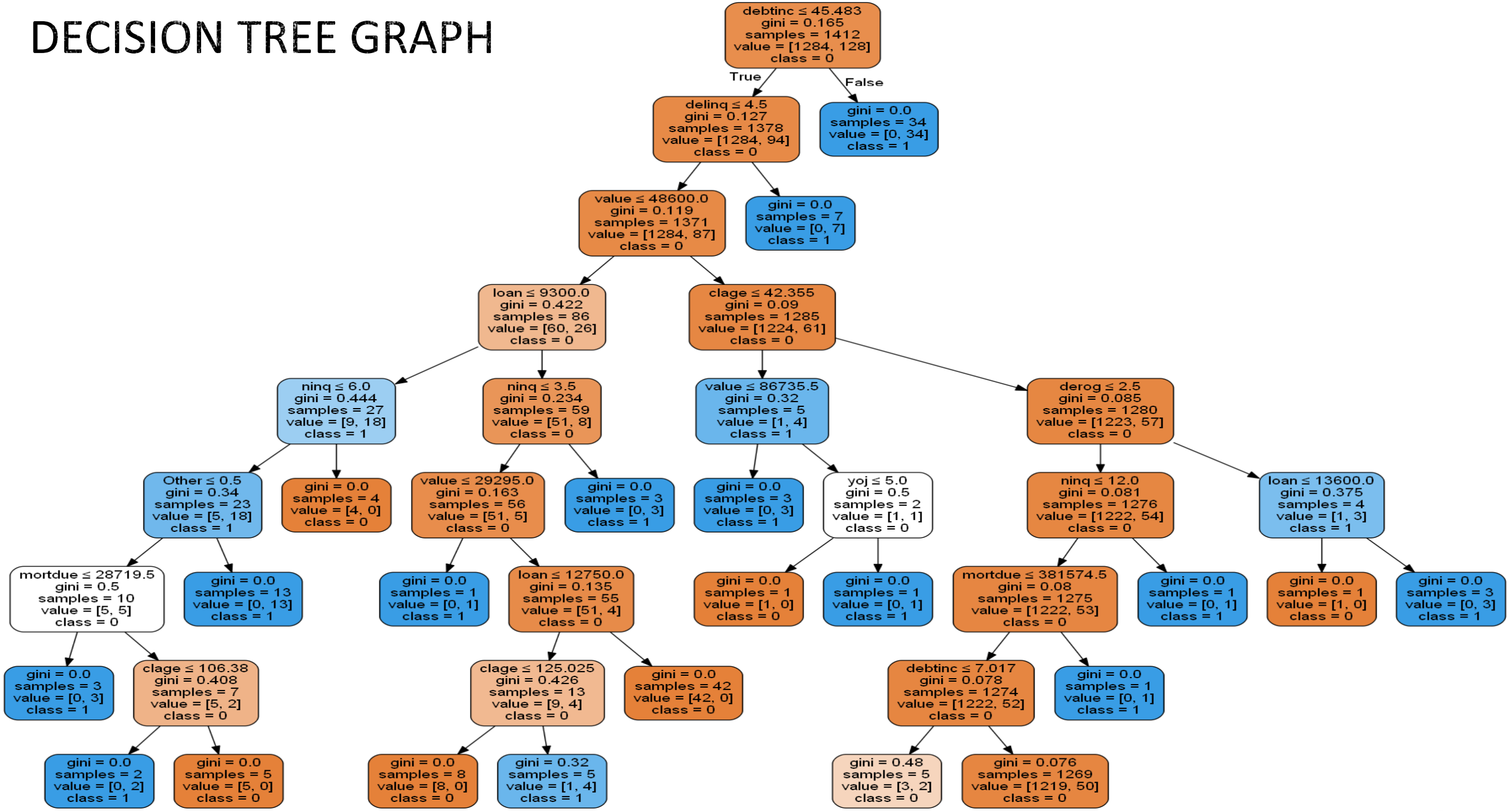
	precision	recall	f1-score	support
0	0.94	0.99	0.97	556
1	0.76	0.32	0.45	50
accuracy			0.94	606
macro avg	0.85	0.66	0.71	606
weighted avg	0.93	0.94	0.92	606

Modelde önemli olan feature lar

Importances Feature and Graph :

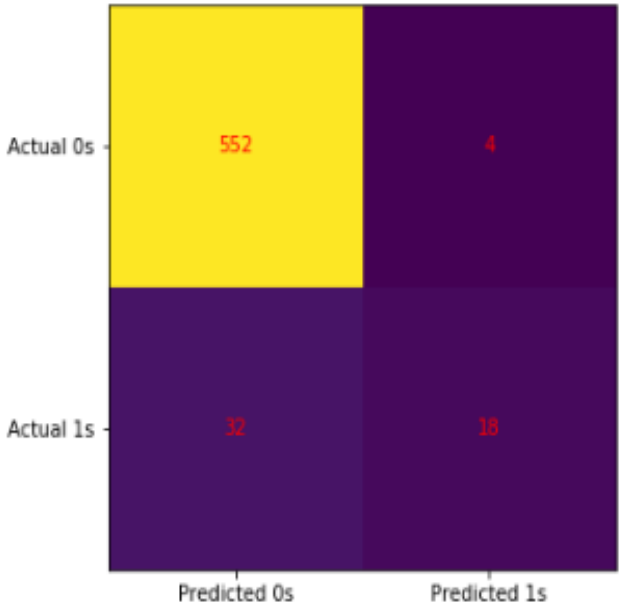
	feature	importance
9	debtinc	0.444
0	loan	0.104
2	value	0.096
6	clage	0.094
5	delinq	0.092
7	ninq	0.081
1	mortdue	0.030
4	derog	0.030
12	Other	0.021
3	yoj	0.008
8	clno	0.000
10	HomeImp	0.000
11	Office	0.000
13	ProfEx	0.000
14	Sales	0.000
15	Self	0.000

DECISION TREE GRAPH



RANDOM FOREST MODEL

```
accuracy,random_forest_model = g.RandomForestCls(X,y)
```

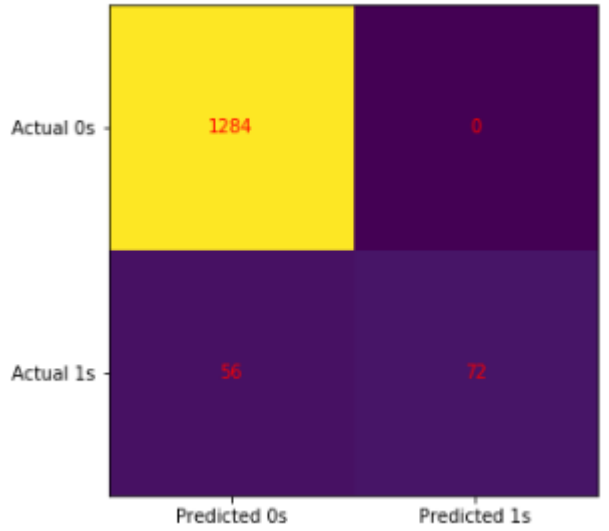


En iyi parametreler seçilerek oluşturulan yeni model

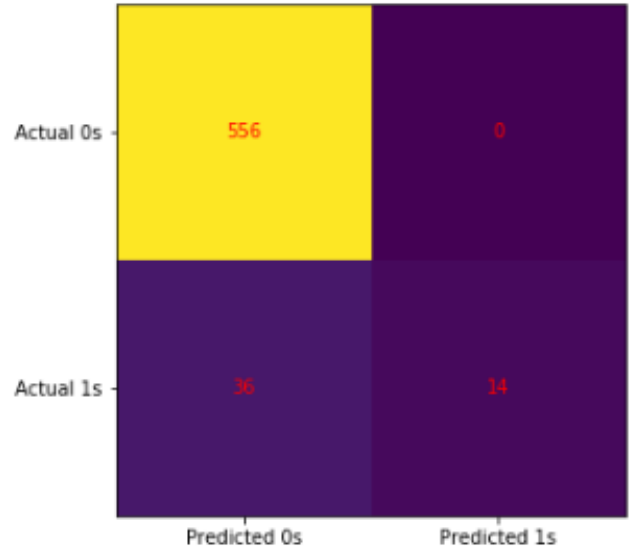
```
{'criterion': 'gini',
 'max_depth': 8,
 'max_features': 'log2',
 'min_samples_split': 5,
 'n_estimators': 50}
```

```
TEST Accuracy  Score :   0.9405940594059405
Classification Report  :
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       556
           1       0.82      0.36      0.50        50

    accuracy                           0.94       606
   macro avg       0.88      0.68      0.73       606
weighted avg       0.93      0.94      0.93       606
```

```
TRAIN Accuracy  Score :   0.9603399433427762
```




```
TEST Accuracy  Score :   0.9405940594059405
```



RANDOM FOREST MODEL

Classification Report :

	precision	recall	f1-score	support
0	0.94	1.00	0.97	556
1	1.00	0.28	0.44	50
accuracy			0.94	606
macro avg	0.97	0.64	0.70	606
weighted avg	0.94	0.94	0.92	606

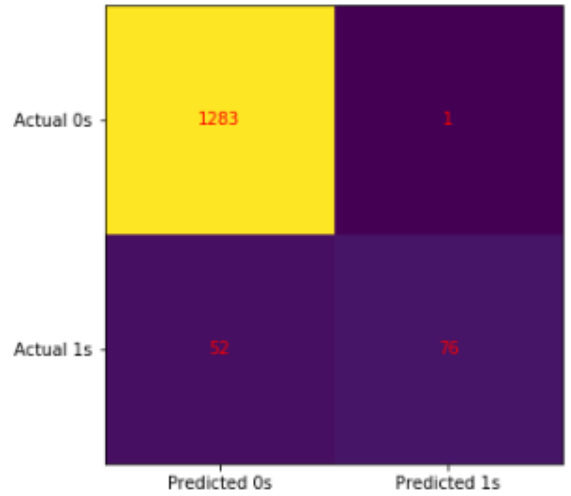
Modelde önemli olan feature lar

Importances Feature and Graph :

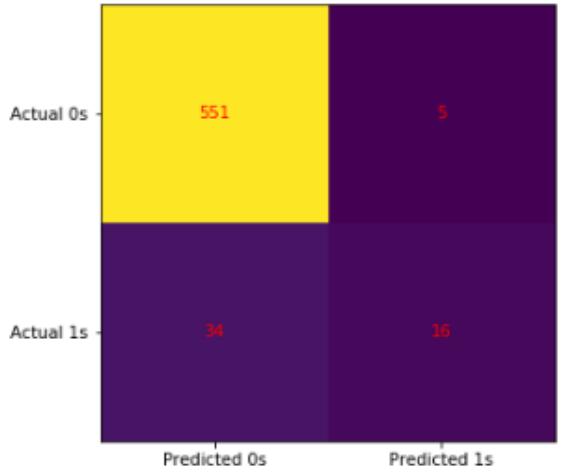
	feature	importance
9	debtinc	0.413
5	delinq	0.091
2	value	0.088
0	loan	0.082
6	clage	0.081
1	mortdue	0.058
8	clno	0.049
4	derog	0.044
7	ninq	0.038
3	yoj	0.034
11	Office	0.006
10	HomeImp	0.005
12	Other	0.005
13	ProfEx	0.002
14	Sales	0.001
15	Self	0.001

DECISION TREE

TRAIN Accuracy Score : 0.9624645892351275

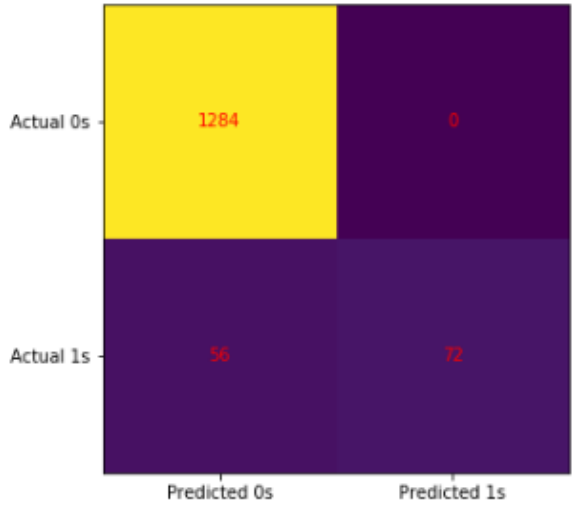


TEST Accuracy Score : 0.9356435643564357

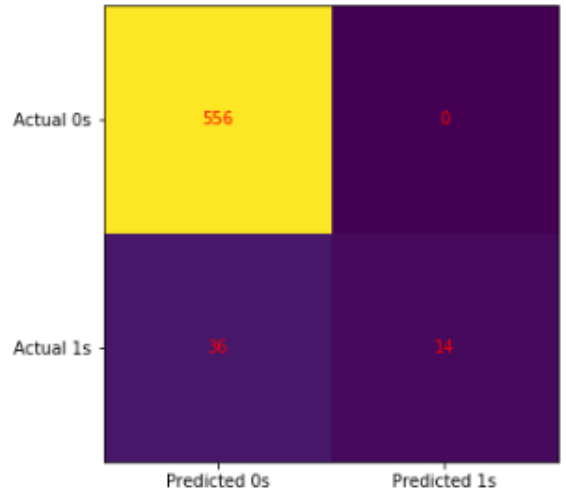


RANDOM FOREST

TRAIN Accuracy Score : 0.9603399433427762



TEST Accuracy Score : 0.9405940594059405



REGRESSION

DATA

```
data = pd.read_excel("HW_Data_Set.xlsx")
```

```
information = Classes.Information(data)
information.data_features()
```

```
-------------------------  DATA HEAD -------------------------

   ind_5  ind_6  ind_8      ind_9     ind_10     ind_12     ind_13  ind_14  \
0     19     17  100.0  85.714286  14.285714  72.363515  60.808814   23.80
1     24     19  100.0  78.571429  21.428571  74.275883  64.366798   11.45
2     30     24  100.0  71.428571  28.571429  75.140402  65.915803    8.75
3     37     30  100.0  64.285714  35.714286  76.677846  68.584234    7.80
4     41     37  100.0  57.142857  42.857143  81.603007  76.455495   14.90

   ind_15  ind_16  ...  ind_416  ind_418  ind_420  ind_422  ind_424   ind_426  \
0   17.62   11.73  ...    -49.6      -54     -152     -353      1.0  0.498547
1   18.16   12.22  ...    -55.6      -60     -158     -359      1.0  0.537088
2   17.86   12.28  ...    -58.4      -60     -160     -362      1.0  0.615169
3   14.76   12.61  ...    -61.8      -65     -166     -367      1.0  0.661517
4   11.92   14.25  ...    -79.8      -86     -186     -388      1.0  0.747204

    ind_428  20_target  50_target  90_target
0  0.701906  15.135802  35.625252  36.997753
1  0.690833  15.143348  35.643013  37.016198
2  0.693040  15.146870  35.651301 -37.024805
3  0.673418  15.153283   0.000000 -37.040483
4  0.700522 -15.179065 -35.727079 -37.103503

[5 rows x 136 columns]
```

DATA

```
data = data[data['ind_420'] != '?']
data = data[data['ind_422'] != '?']

#dummy

RED = pd.get_dummies(data['ind_109'], drop_first =True)
data=pd.concat([data,RED],axis=1)
data.drop(['ind_109'],axis =1,inplace=True)
```

```
X = data.iloc[:, 0:132]
y = data.loc[:, data.columns == '20_target']
```

```
cate =X.select_dtypes(include='object')
cate
```

	ind_420	ind_422
0	-152	-353
1	-158	-359
2	-160	-362
3	-166	-367
4	-186	-388
...	...	...
6162	-11	-270
6163	-12	-271
6164	-21	-280
6165	-33	-292
6166	-28	-288

5126 rows × 2 columns

```
X.astype('float64').dtypes
```

Veri seti içindeki eksik gözlem bulunan satırlar silinip veriseti X ve y değişkenlerine bölündü

DECISION TREE MODEL

```
reg_model = DecisionTreeRegressor()
```

```
reg_model.fit(X_train,y_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=None, 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,
                      presort=False, random_state=None, splitter='best')
```

```
y_pred = reg_model.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
13.980320214001953
```

DECISION TREE MODEL

```
cart_cv_model.best_params_
```

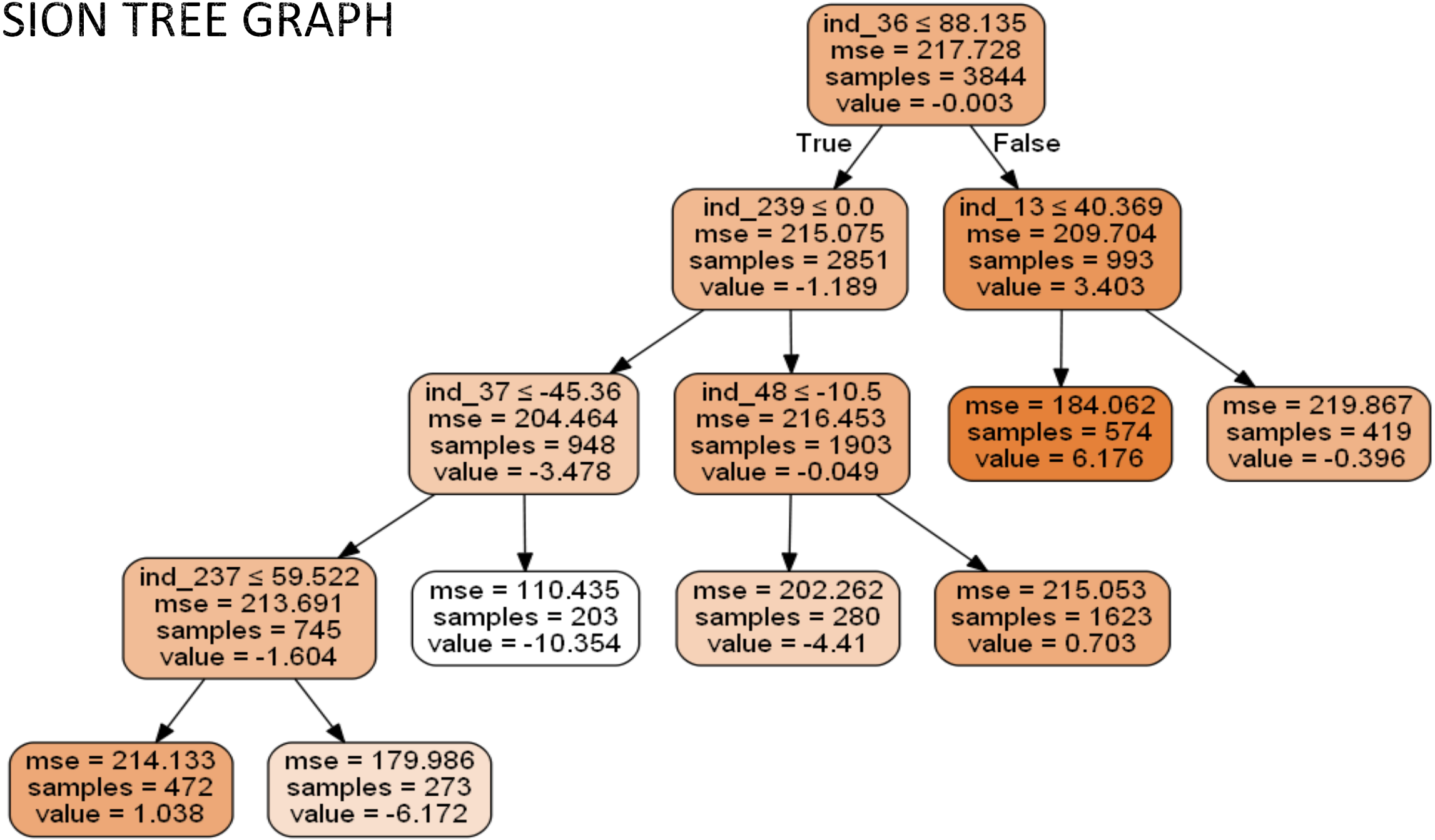
```
{'max_depth': 7,
 'max_leaf_nodes': 7,
 'min_samples_leaf': 10,
 'min_samples_split': 27}
```

En iyi parametreler seçilerek oluşturulan yeni model

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
14.490464186560065
```

DECISION TREE GRAPH



RANDOM FOREST MODEL

```
regressor = RandomForestRegressor(n_estimators = 100, random_state = 42)
regressor.fit(X_train, y_train)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', 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_jobs=None, oob_score=False, random_state=42, verbose=0,
                      warm_start=False)
```

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
10.205892872015024
```

RANDOM FOREST MODEL

```
# Model Tuning

rf_params = {"max_features" : [3,5,10,15],
             "n_estimators" :[100,200,500,1000,2000]}

rf_model = RandomForestRegressor(random_state =42)
```

En iyi parametreler seçilerek oluşturulan yeni model

```
{'max_features': 15, 'n_estimators': 2000}
```

```
rf_tuned = RandomForestRegressor(max_features =15,n_estimators = 2000, random_state =42)
```

```
rf_tuned.fit(X_train, y_train)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features=15, 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=2000,
                      n_jobs=None, oob_score=False, random_state=42, verbose=0,
                      warm_start=False)
```

```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
10.205892872015024
```

RANDOM FOREST MODEL

Modelde önemli olan feature lar

	feature	importance
128	ind_422	0.020
127	ind_420	0.020
75	ind_156	0.018
29	ind_37	0.017
96	ind_313	0.017
...	...	...
111	ind_349	0.000
110	ind_347	0.000
109	ind_345	0.000
108	ind_344	0.000
20	ind_28	0.000

132 rows × 2 columns

DECISION TREE

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
14.490464186560065
```

RANDOM FOREST

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
10.205892872015024
```