```
# TEMEL
import numpy as np
import pandas as pd

# GÖRSELLEŞTİRME
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# UYARILARI KAPAT
import warnings
warnings.filterwarnings("ignore")

# MAKİNE ÖĞRENMESİ - TEMEL
from sklearn.model_selection import train_test_split, GridSearchCV,
learning_curve
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.pipeline import Pipeline

# MODELLER
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier

# DEĞERLENDİRME METRİKLERİ
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    classification_report,
    roc_auc_score,
    RocCurveDisplay
)

# OPSİYONEL GÜÇLÜ MODELLER
# !pip install catboost
# from catboost import CatBoostClassifier

# !pip install xgboost
# from xgboost import XGBClassifier

df =
pd.read_csv('/Users/tugcekizilkoca/Desktop/veri-madenciligi/datasets/
heart.csv')

# Returns number of rows and columns of the dataset
df.shape
```

```
(303, 14)
```

```
# Returns an object with all of the column headers
df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
# Returns different datatypes for each columns (float, int, string,
bool, etc.)
df.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

```
# Returns the first x number of rows when head(x). Without a number it
returns 5
df.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   63    1   3       145   233    1        0      150      0      2.3
0
1   37    1   2       130   250    0        1      187      0      3.5
0
2   41    0   1       130   204    0        0      172      0      1.4
2
3   56    1   1       120   236    0        1      178      0      0.8
2
4   57    0   0       120   354    0        1      163      1      0.6
2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
```

```
3   0     2       1
4   0     2       1
```

```
# Returns the last x number of rows when tail(x). Without a number it
returns 5
df.tail()
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak  \
298   57    0   0       140   241    0        1      123      1
0.2
299   45    1   3       110   264    0        1      132      0
1.2
300   68    1   0       144   193    1        1      141      0
3.4
301   57    1   0       130   131    0        1      115      1
1.2
302   57    0   1       130   236    0        0      174      0
0.0

     slope  ca  thal  target
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0
```

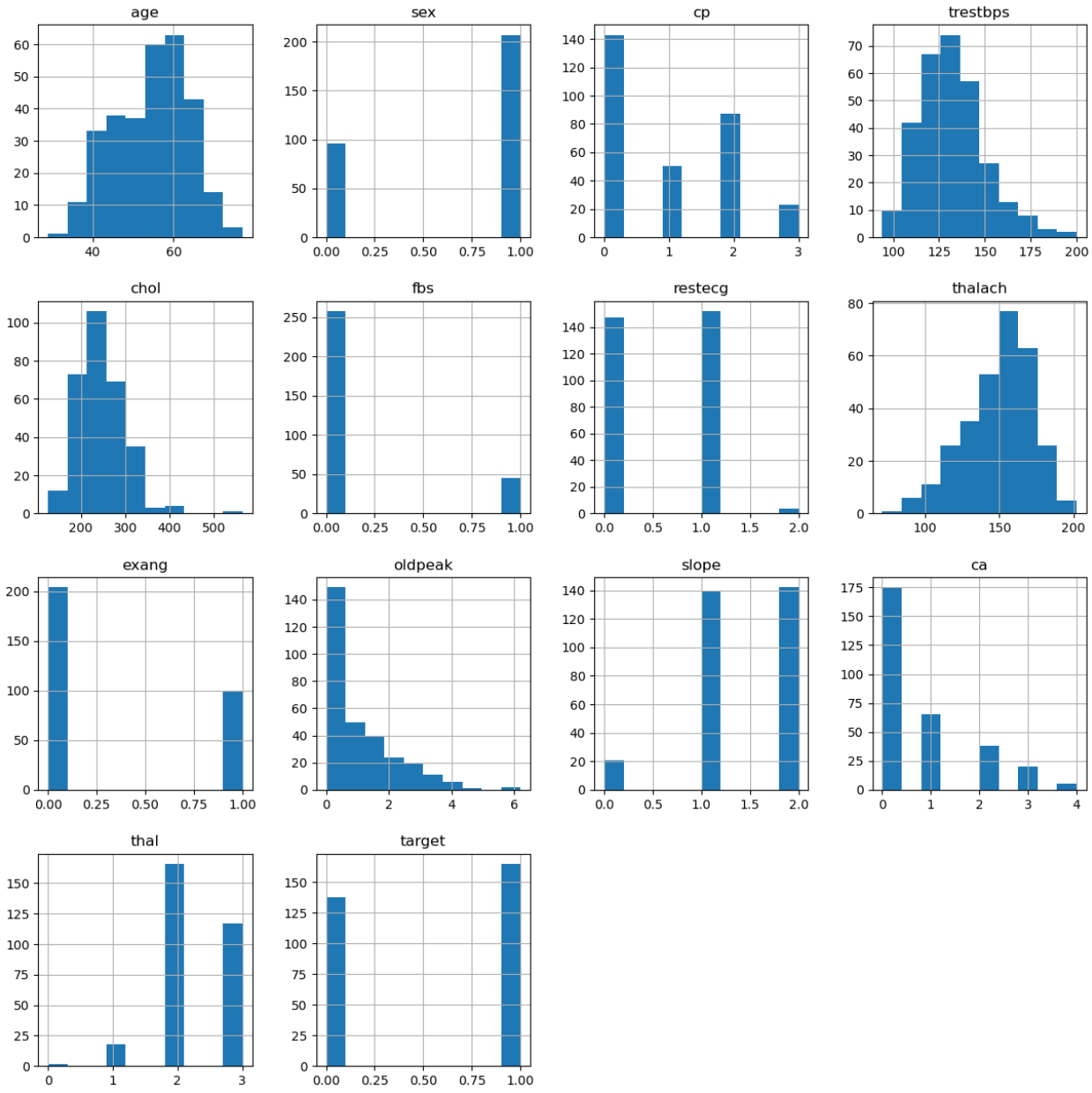
```
# Returns true for a column having null values, else false
df.isnull().any()
```

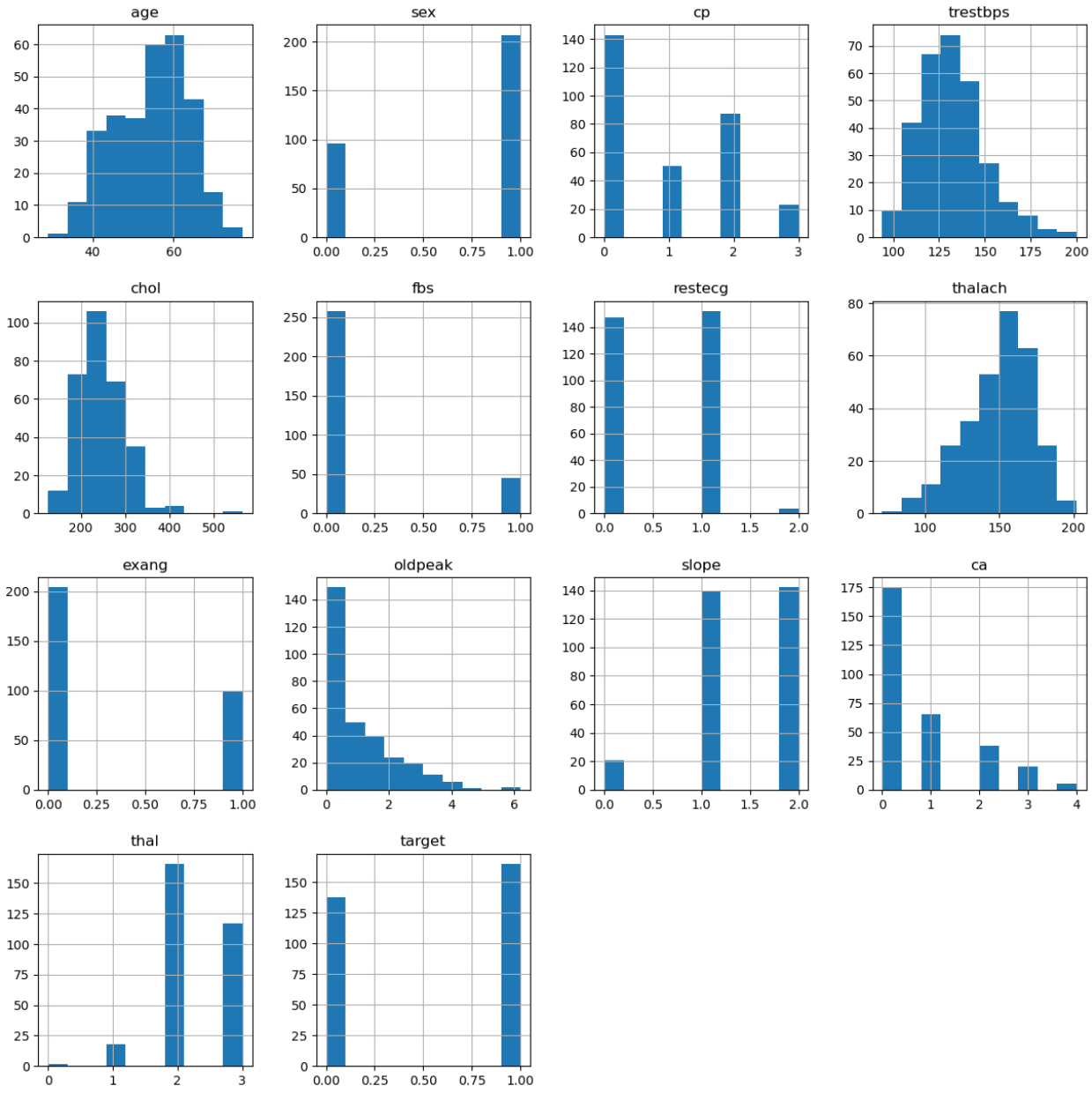
```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

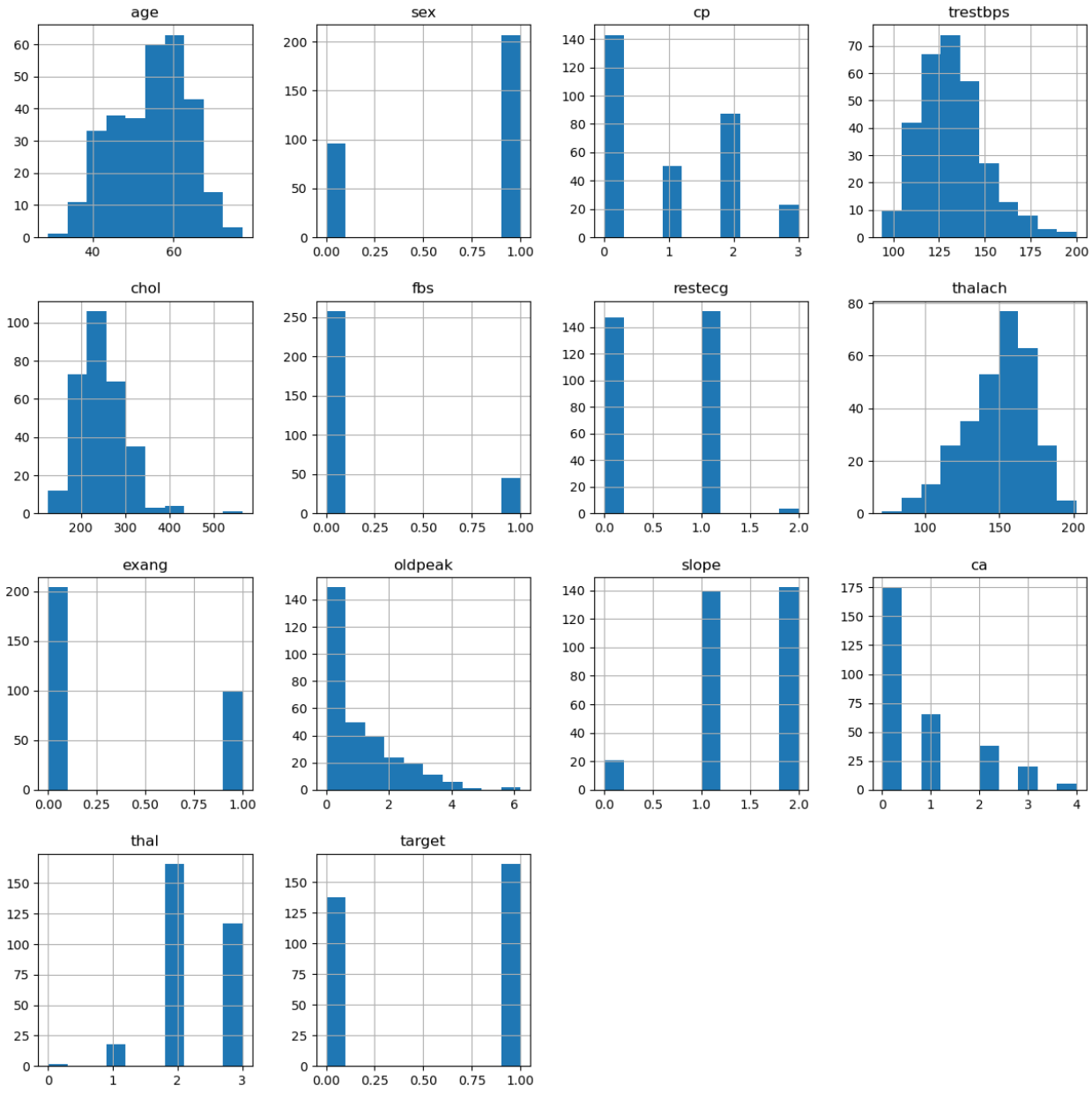
```
# Returns basic statistics on numeric columns
df.describe().T
```

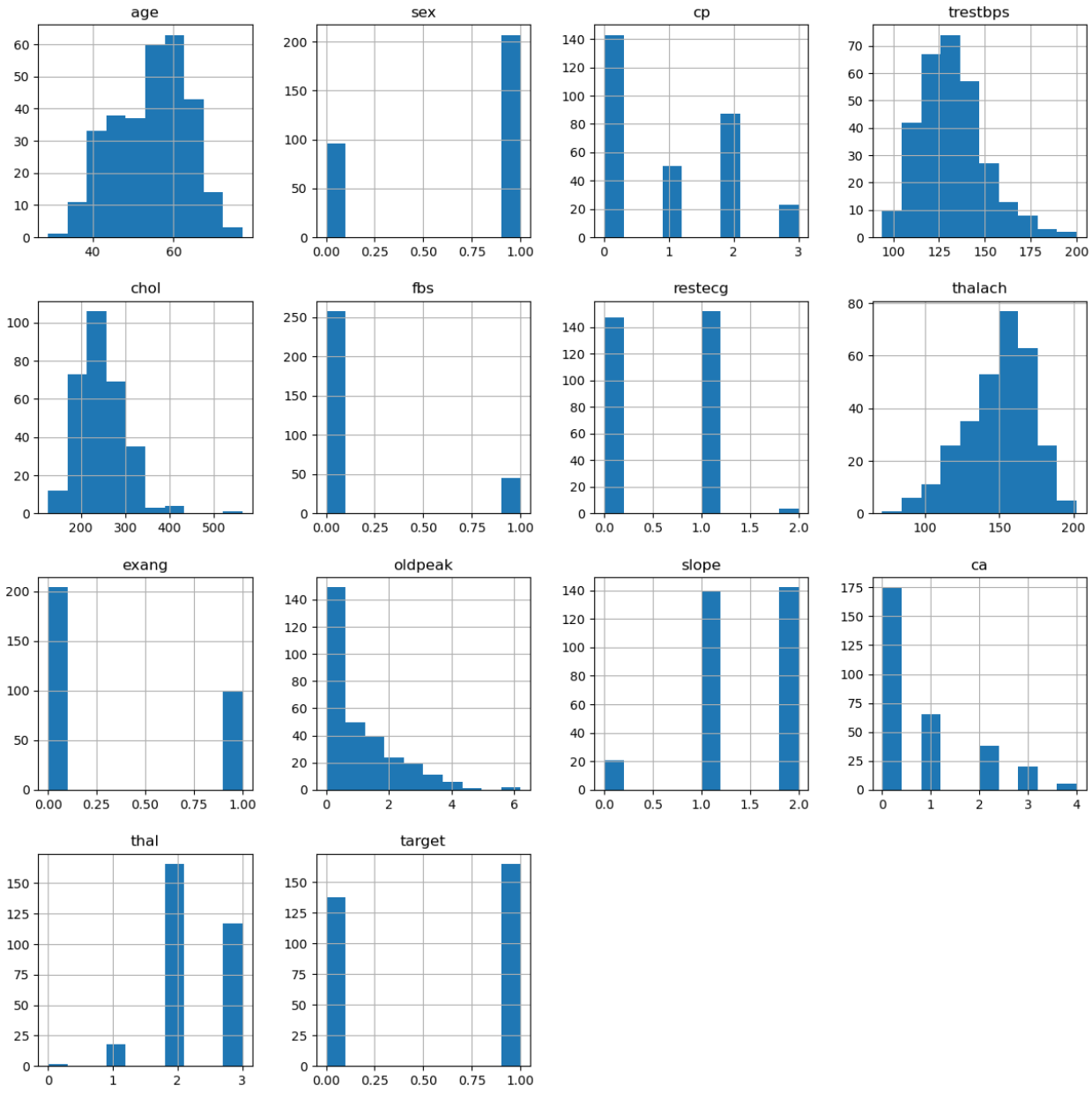
```
          count        mean        std    min    25%    50%    75%
max
```

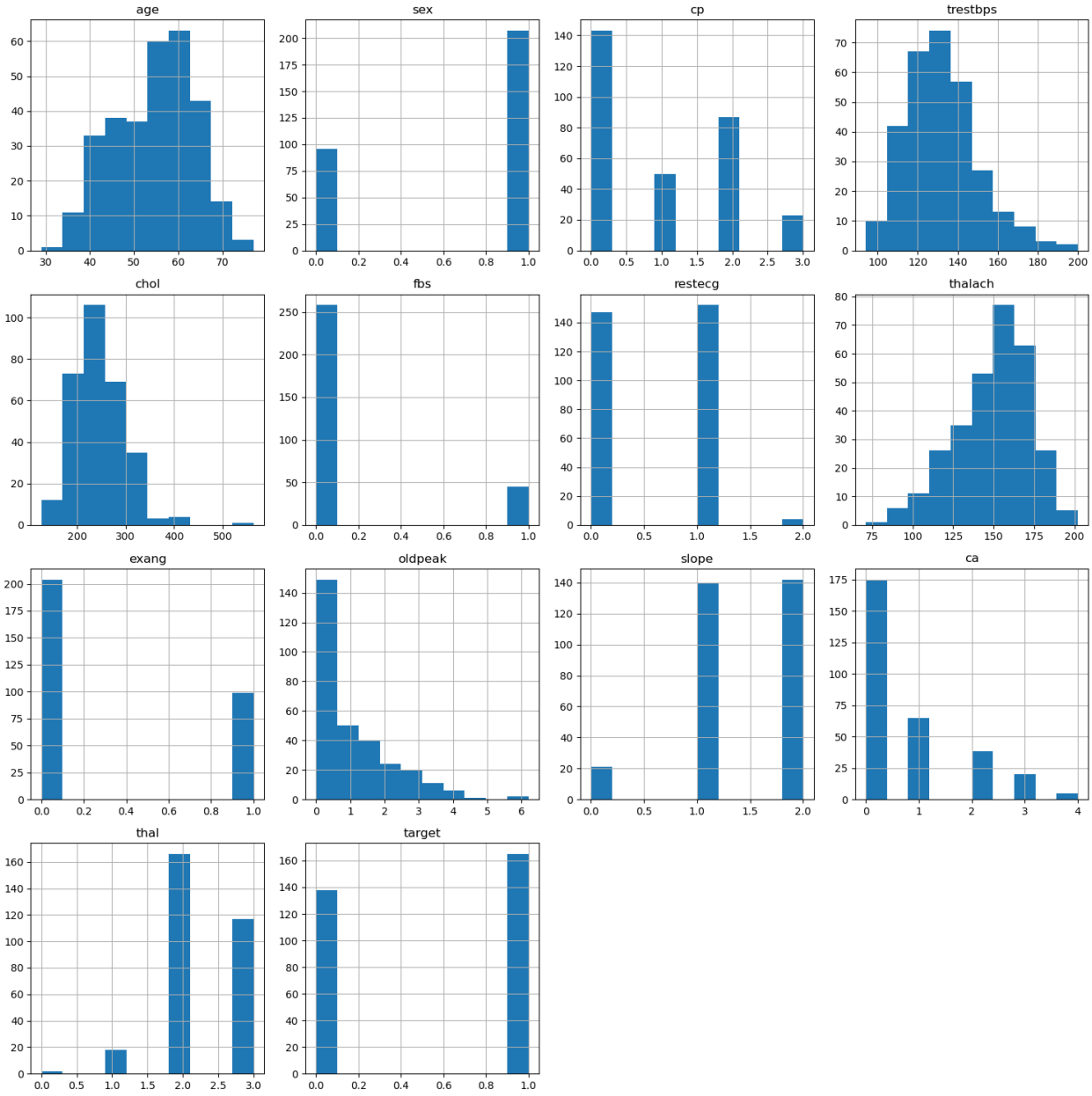
```
age       303.0   54.366337   9.082101   29.0   47.5   55.0   61.0
77.0
sex       303.0    0.683168   0.466011    0.0    0.0    1.0    1.0
1.0
cp        303.0    0.966997   1.032052    0.0    0.0    1.0    2.0
3.0
trestbps  303.0  131.623762  17.538143   94.0  120.0  130.0  140.0
200.0
chol      303.0  246.264026  51.830751  126.0  211.0  240.0  274.5
564.0
fbs       303.0    0.148515   0.356198    0.0    0.0    0.0    0.0
1.0
restecg   303.0    0.528053   0.525860    0.0    0.0    1.0    1.0
2.0
thalach   303.0  149.646865  22.905161   71.0  133.5  153.0  166.0
202.0
exang     303.0    0.326733   0.469794    0.0    0.0    0.0    1.0
1.0
oldpeak   303.0    1.039604   1.161075    0.0    0.0    0.8    1.6
6.2
slope     303.0    1.399340   0.616226    0.0    1.0    1.0    2.0
2.0
ca        303.0    0.729373   1.022606    0.0    0.0    0.0    1.0
4.0
thal      303.0    2.313531   0.612277    0.0    2.0    2.0    3.0
3.0
target    303.0    0.544554   0.498835    0.0    0.0    1.0    1.0
1.0

df.hist(figsize=(15,15))
plt.tight_layout()
plt.show()
```


age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target

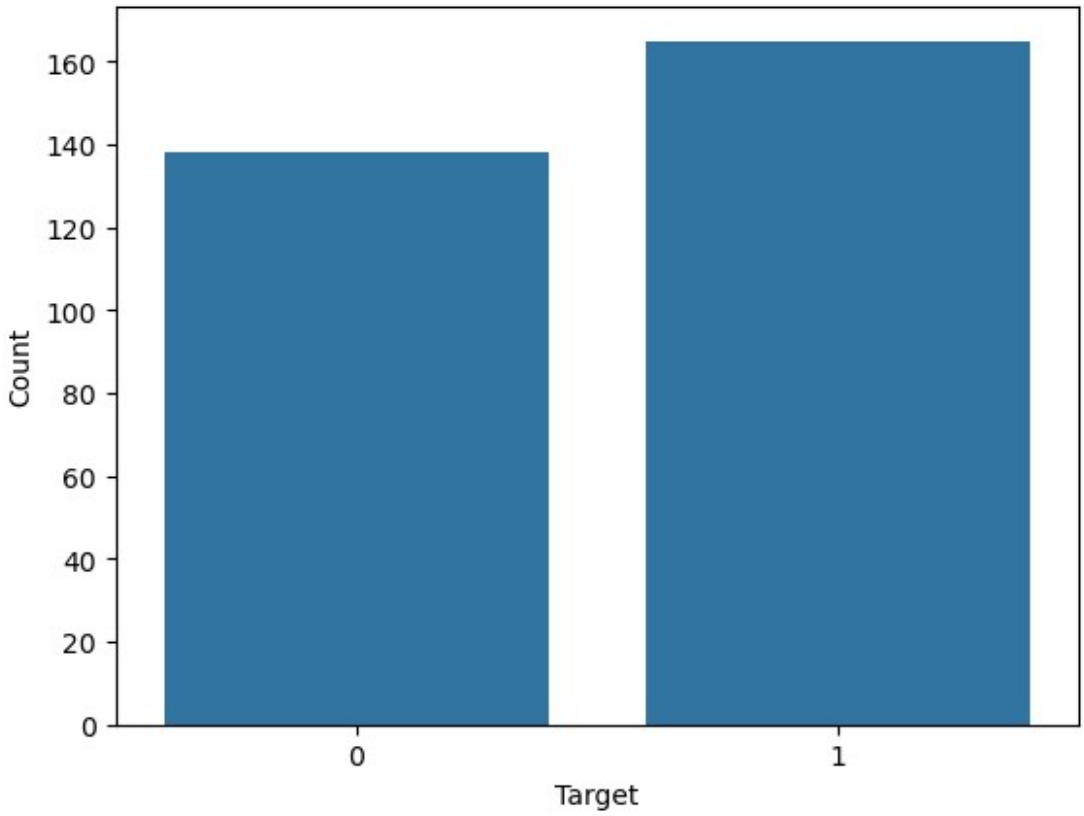

age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target


age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target

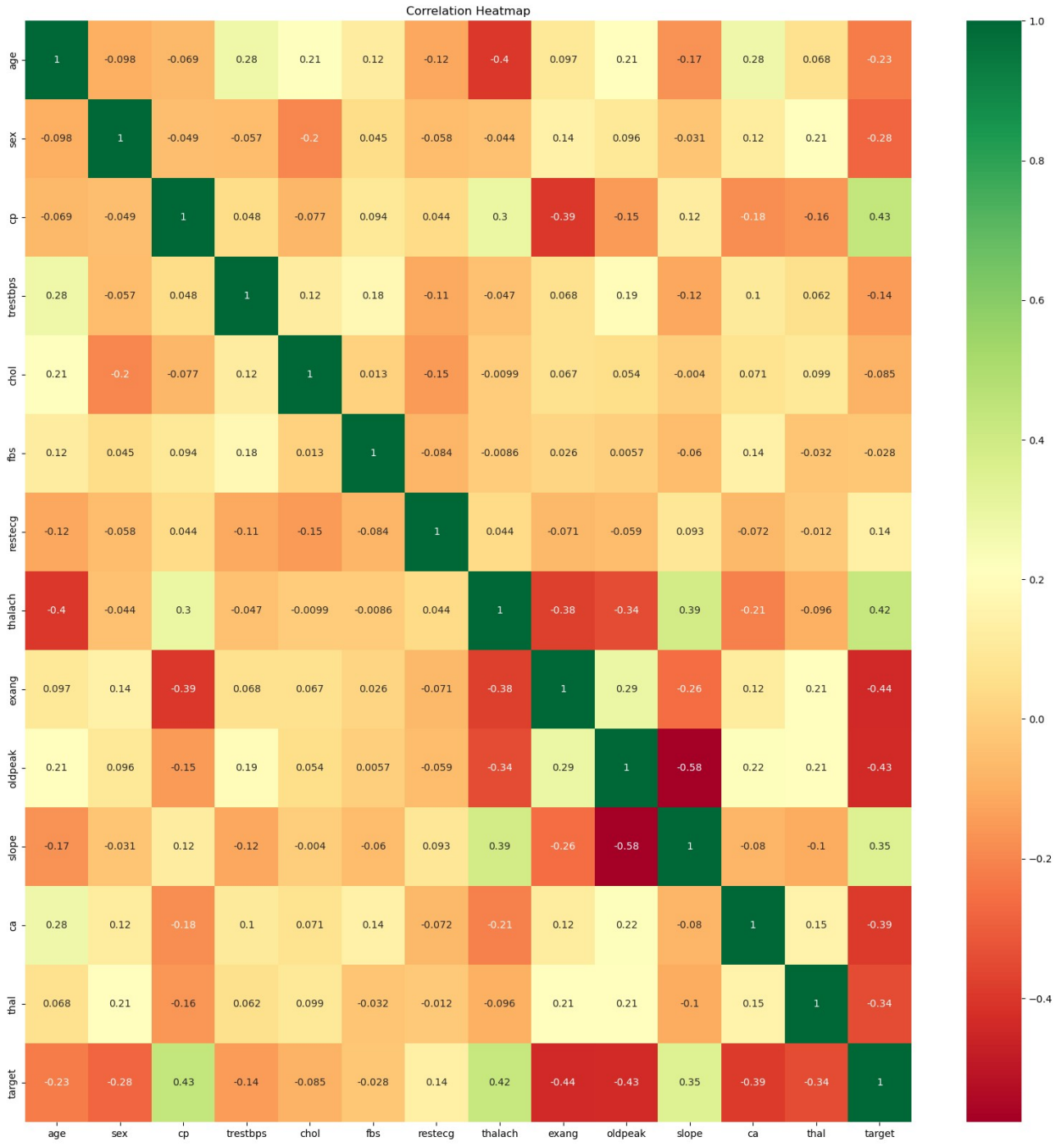

age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target




```
sns.countplot(x='target', data=df)
plt.xlabel("Target")
plt.ylabel("Count")
plt.show()
```




```
# Selecting correlated features using Heatmap

# Get correlation of all the features of the dataset
corr_matrix = df.corr()
top_corr_features = corr_matrix.index

# Plotting the heatmap
plt.figure(figsize=(20,20))
sns.heatmap(df[top_corr_features].corr(), annot=True, cmap='RdYlGn')
plt.title("Correlation Heatmap")
plt.show()
```




```
dataset = pd.get_dummies(df, columns=['sex', 'cp', 'fbs', 'restecg',
'exang', 'slope', 'ca', 'thal'])

dataset.columns

Index(['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target',
'sex_0',
       'sex_1', 'cp_0', 'cp_1', 'cp_2', 'cp_3', 'fbs_0', 'fbs_1',
'restecg_0',
```

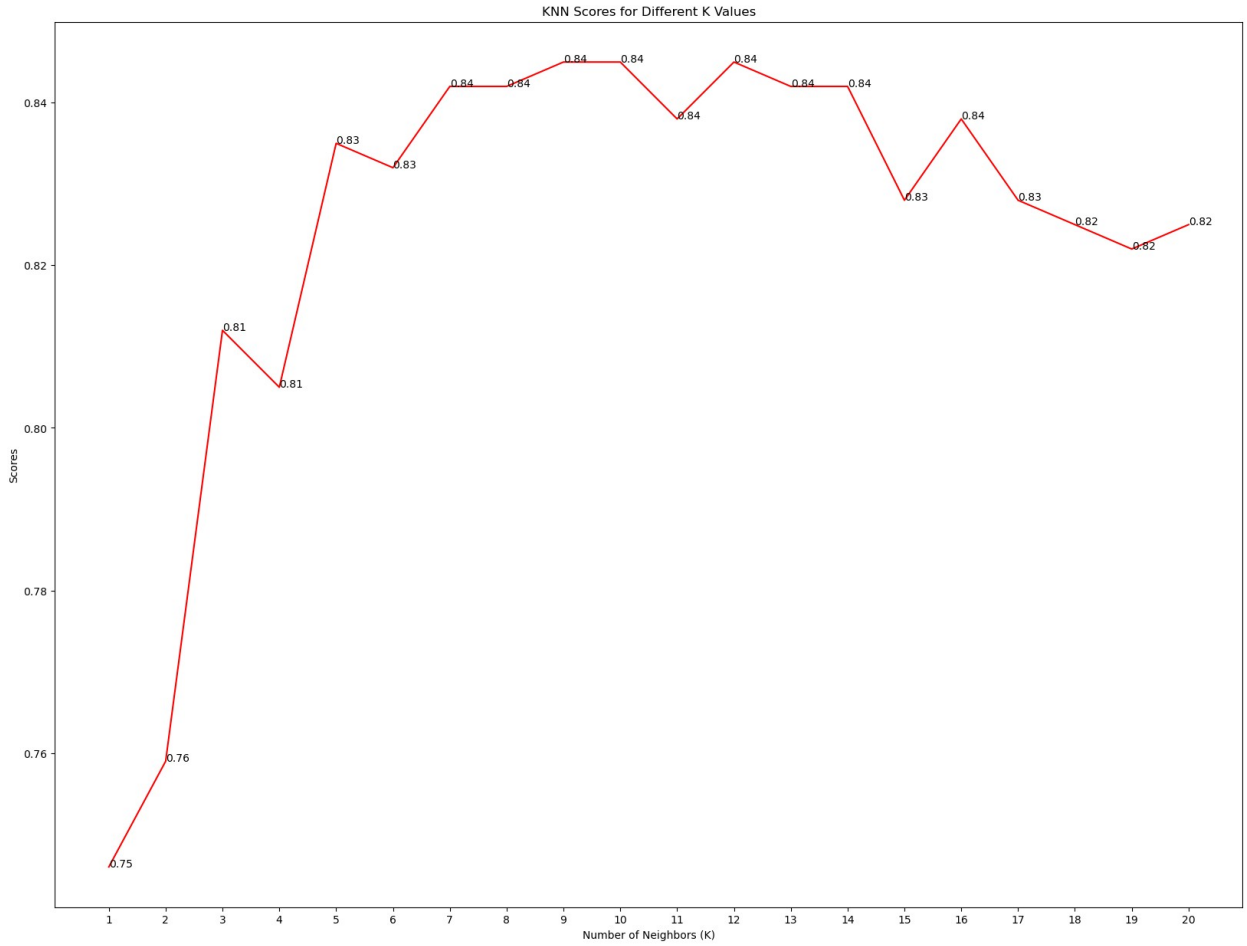
```
       'restecg_1', 'restecg_2', 'exang_0', 'exang_1', 'slope_0',
'slope_1',
       'slope_2', 'ca_0', 'ca_1', 'ca_2', 'ca_3', 'ca_4', 'thal_0',
'thal_1',
       'thal_2', 'thal_3'],
      dtype='object')
```

```
from sklearn.preprocessing import StandardScaler
standScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] =
standScaler.fit_transform(dataset[columns_to_scale])
```

```
dataset.head()
```

```
        age  trestbps      chol   thalach   oldpeak  target  sex_0
sex_1  \
0  0.952197  0.763956 -0.256334  0.015443  1.087338       1  False
True
1 -1.915313 -0.092738  0.072199  1.633471  2.122573       1  False
True
2 -1.474158 -0.092738 -0.816773  0.977514  0.310912       1   True
False
3  0.180175 -0.663867 -0.198357  1.239897 -0.206705       1  False
True
4  0.290464 -0.663867  2.082050  0.583939 -0.379244       1   True
False

    cp_0   cp_1  ...  slope_2  ca_0   ca_1   ca_2   ca_3   ca_4
thal_0  \
0  False  False  ...    False  True  False  False  False  False
False
1  False  False  ...    False  True  False  False  False  False
False
2  False   True  ...     True  True  False  False  False  False
False
3  False   True  ...     True  True  False  False  False  False
False
4   True  False  ...     True  True  False  False  False  False
False

   thal_1  thal_2  thal_3
0    True   False   False
1   False    True   False
2   False    True   False
3   False    True   False
4   False    True   False

[5 rows x 31 columns]
```

```python
# Splitting the dataset into dependent and independent features
X = dataset.drop('target', axis=1)
y = dataset['target']

#KNN

# Importing essential libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

# Finding the best accuracy for knn algorithm using cross_val_score
knn_scores = []
for i in range(1, 21):
  knn_classifier = KNeighborsClassifier(n_neighbors=i)
  cvs_scores = cross_val_score(knn_classifier, X, y, cv=10)
  knn_scores.append(round(cvs_scores.mean(),3))

plt.figure(figsize=(20,15))

# skor grafiği
plt.plot(range(1,21), knn_scores, color='red')

# her noktanın üzerine skor yaz
for i, score in enumerate(knn_scores, start=1):
    plt.text(i, score, f"{score:.2f}")

# eksen ayarları
plt.xticks(range(1,21))
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('KNN Scores for Different K Values')

plt.show()
```

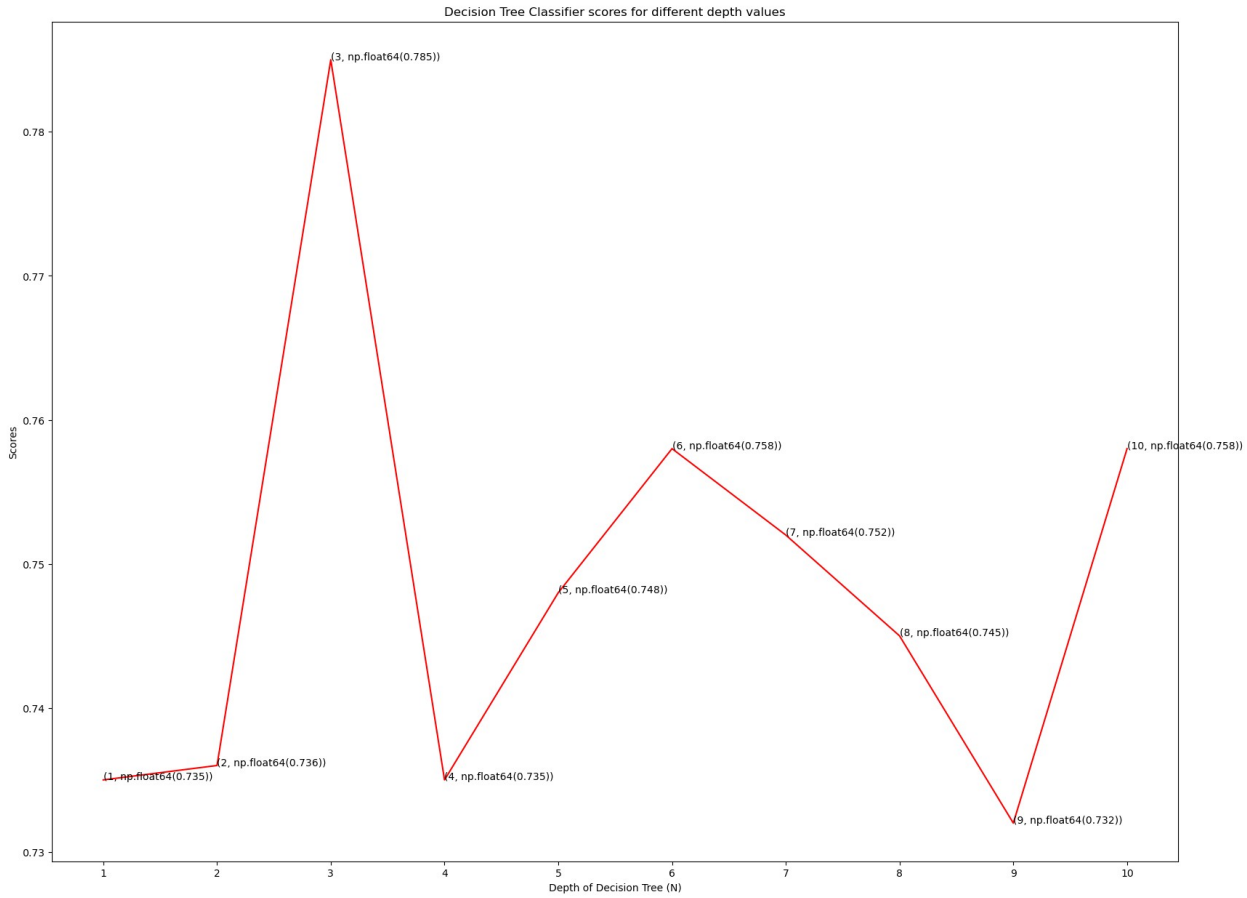



```
# Training the knn classifier model with k value as 12
knn_classifier = KNeighborsClassifier(n_neighbors=12)
cvs_scores = cross_val_score(knn_classifier, X, y, cv=10)
print("KNeighbours Classifier Accuracy with K=12 is: {}
%".format(round(cvs_scores.mean(), 4)*100))
```

```
KNeighbours Classifier Accuracy with K=12 is: 84.48%
```

```
# Importing essential libraries
from sklearn.tree import DecisionTreeClassifier

# Finding the best accuracy for decision tree algorithm using
cross_val_score
decision_scores = []
for i in range(1, 11):
  decision_classifier = DecisionTreeClassifier(max_depth=i)
  cvs_scores = cross_val_score(decision_classifier, X, y, cv=10)
  decision_scores.append(round(cvs_scores.mean(),3))

# Plotting the results of decision_scores
plt.figure(figsize=(20,15))
plt.plot([i for i in range(1, 11)], decision_scores, color = 'red')
```

```
for i in range(1,11):
    plt.text(i, decision_scores[i-1], (i, decision_scores[i-1]))
plt.xticks([i for i in range(1, 11)])
plt.xlabel('Depth of Decision Tree (N)')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different depth
values')
plt.show()
```




```
# Training the decision tree classifier model with max_depth value as
3
decision_classifier = DecisionTreeClassifier(max_depth=3)
cvs_scores = cross_val_score(decision_classifier, X, y, cv=10)
print("Decision Tree Classifier Accuracy with max_depth=3 is: {}
%".format(round(cvs_scores.mean(), 4)*100))
```

```
Decision Tree Classifier Accuracy with max_depth=3 is: 78.51%
```

```
# Importing essential libraries
from sklearn.ensemble import RandomForestClassifier
```

```
# Finding the best accuracy for random forest algorithm using
cross_val_score
forest_scores = []
for i in range(10, 101, 10):
  forest_classifier = RandomForestClassifier(n_estimators=i)
  cvs_scores = cross_val_score(forest_classifier, X, y, cv=5)
  forest_scores.append(round(cvs_scores.mean(),3))

# Plotting the results of forest_scores
plt.figure(figsize=(20,15))
plt.plot([n for n in range(10, 101, 10)], forest_scores, color =
'red')
for i in range(1,11):
    plt.text(i*10, forest_scores[i-1], (i*10, forest_scores[i-1]))
plt.xticks([i for i in range(10, 101, 10)])
plt.xlabel('Number of Estimators (N)')
plt.ylabel('Scores')
plt.title('Random Forest Classifier scores for different N values')
```

```
Text(0.5, 1.0, 'Random Forest Classifier scores for different N
values')
```

```
# Training the random forest classifier model with n value as 90
forest_classifier = RandomForestClassifier(n_estimators=90)
cvs_scores = cross_val_score(forest_classifier, X, y, cv=5)
print("Random Forest Classifier Accuracy with n_estimators=90 is: {}
%".format(round(cvs_scores.mean(), 4)*100))
```

```
Random Forest Classifier Accuracy with n_estimators=90 is: 81.83%
```

```
# Hedef değişken ismini burada ayarla:
TARGET_COL = 'target'   # kalp hastalığı datası için
# TARGET_COL = 'Completed'   # kurs tamamlama datası için

# X, y ayrımı
X = df.drop(columns=[TARGET_COL])
y = df[TARGET_COL]

# Train–test ayırma
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

X_train.shape, X_test.shape
```

```
((242, 13), (61, 13))
```