Heart Diseases(DAV Mini project)

I have collected the dataset from Github (https://github.com/kb22/Heart-Disease-Prediction/blob/dbd27c35db3a128f7f87a2d1b8200f1f14e4affb/dataset.csv) and I will be using Machine Learning to make predictions on whether a person is suffering from Heart Disease or not.

Import libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Import dataset

```
df = pd.read_csv('/content/heart.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
1 df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	52	1	0	125	212	0	1	168	0	1.0	2	2	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	

```
1 df.shape
```

```
(1025, 14)
```

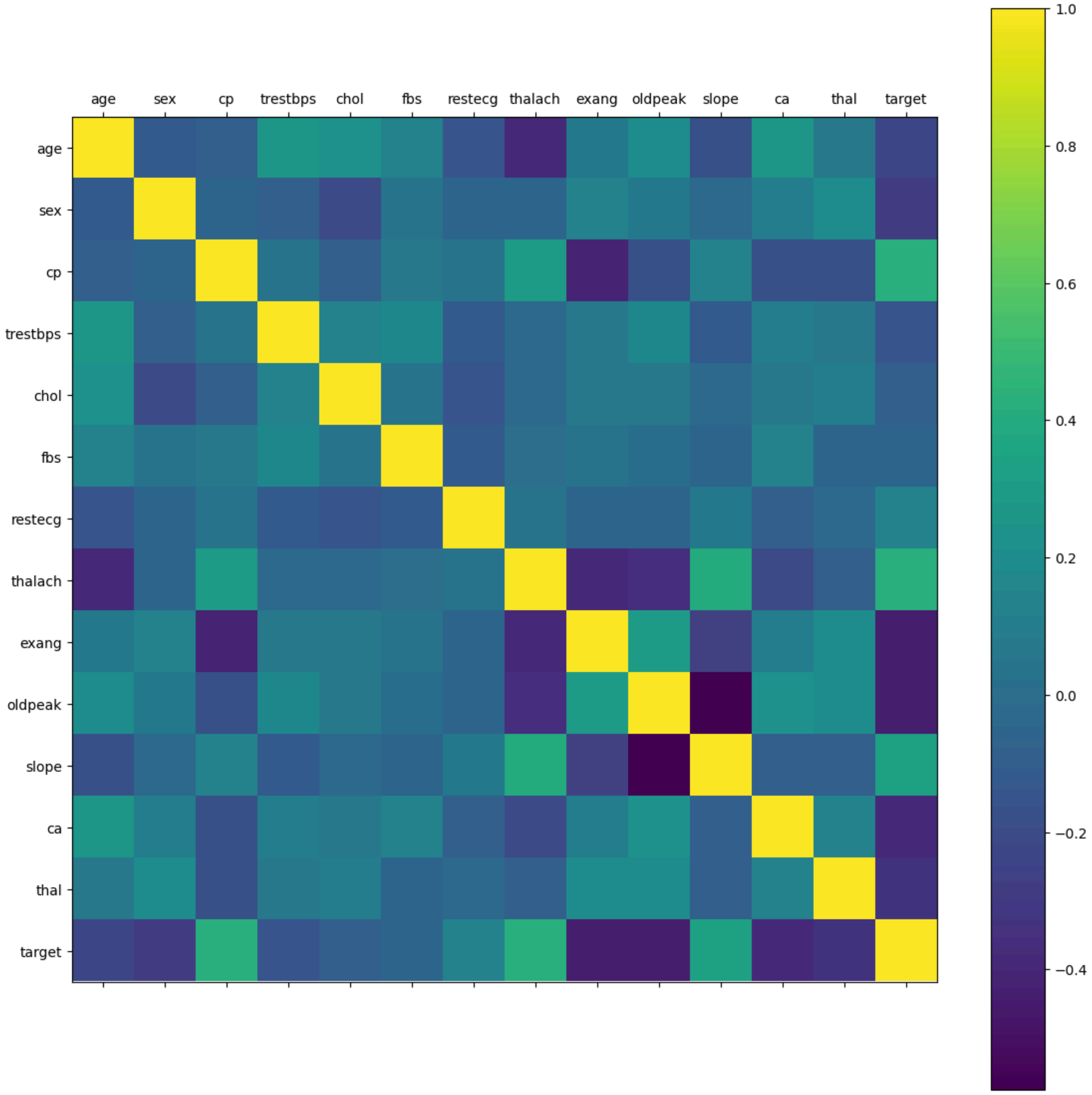
```
1 df.describe()
```

	age	sex	cp	trestbps	chol	fbs
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000

Understanding the data or Analyzing the data

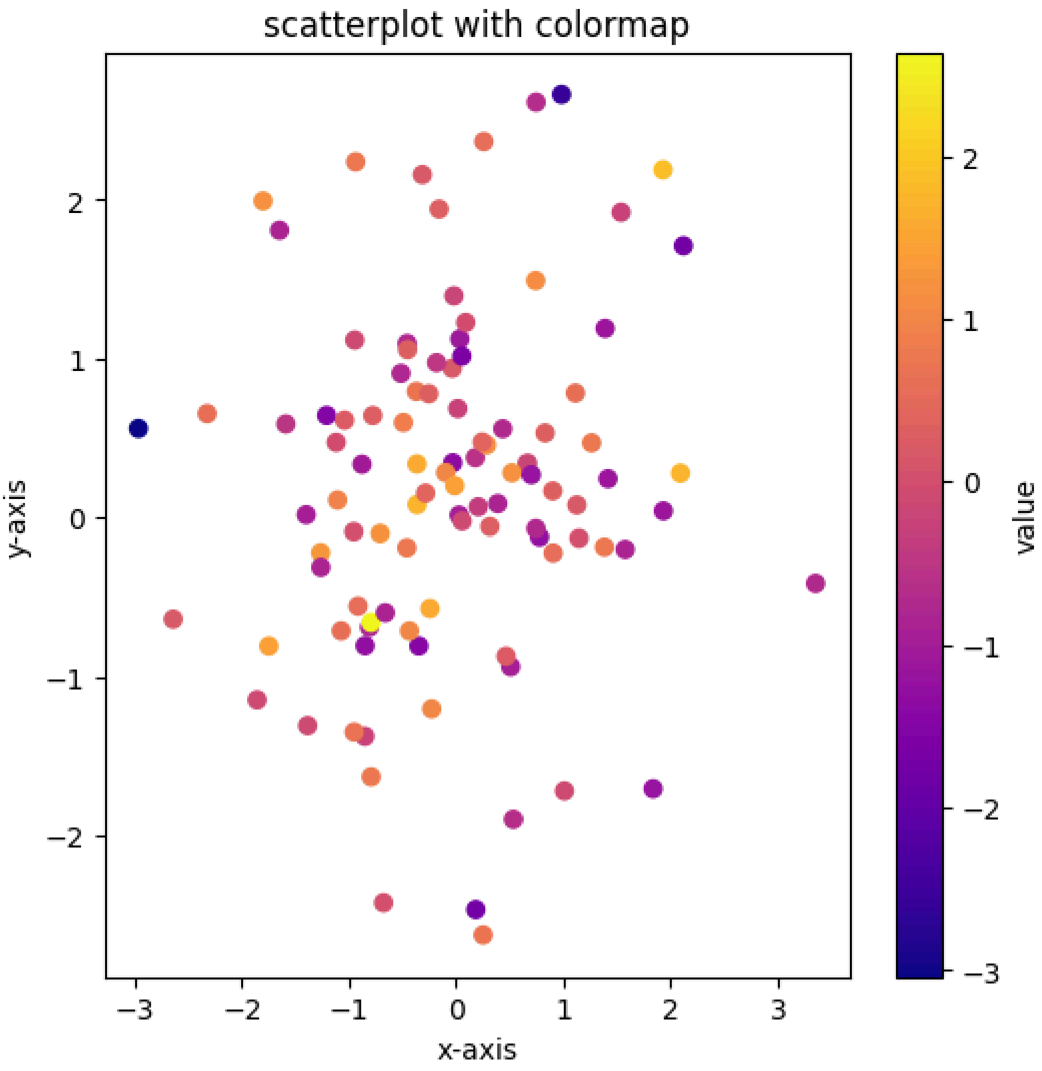
```
rcParams['figure.figsize'] = 20, 14
plt.matshow(df.corr())
plt.xticks(np.arange(df.shape[1]), df.columns)
plt.yticks(np.arange(df.shape[1]), df.columns)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7f4140fb1cc0>
```


age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

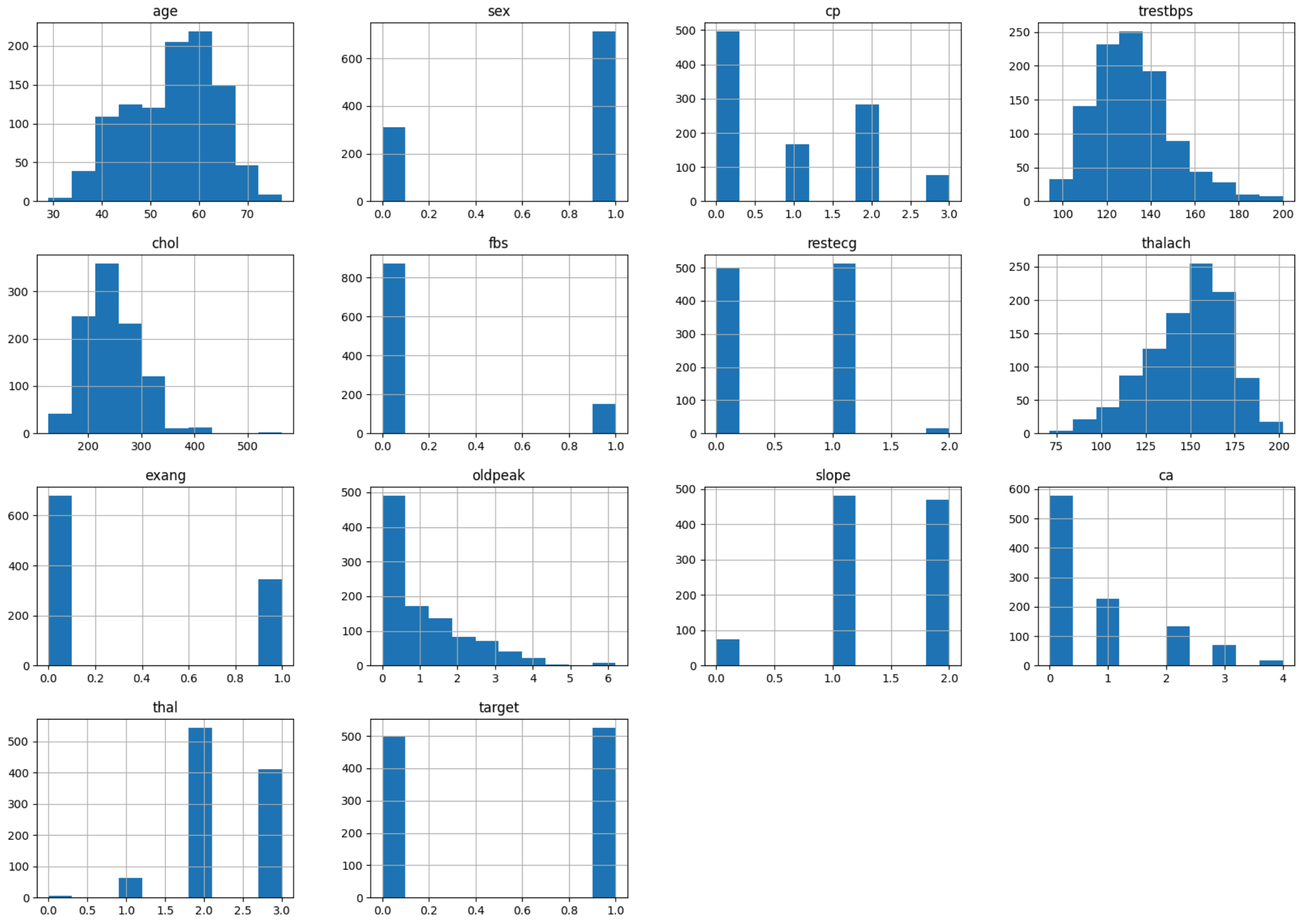
```
data=pd.DataFrame({
    "x":np.random.randn(100),
    "y":np.random.randn(100),
    "value":np.random.randn(100)
})
cmap="plasma"
alpha=1
plt.figure(figsize=(6,6))
plt.scatter(data["x"],data["y"],c=data["value"],cmap=cmap,alpha=alpha)
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("scatterplot with colormap")
plt.colorbar(label="value")
```

```
<matplotlib.colorbar.Colorbar at 0x7f413d90be80>
```



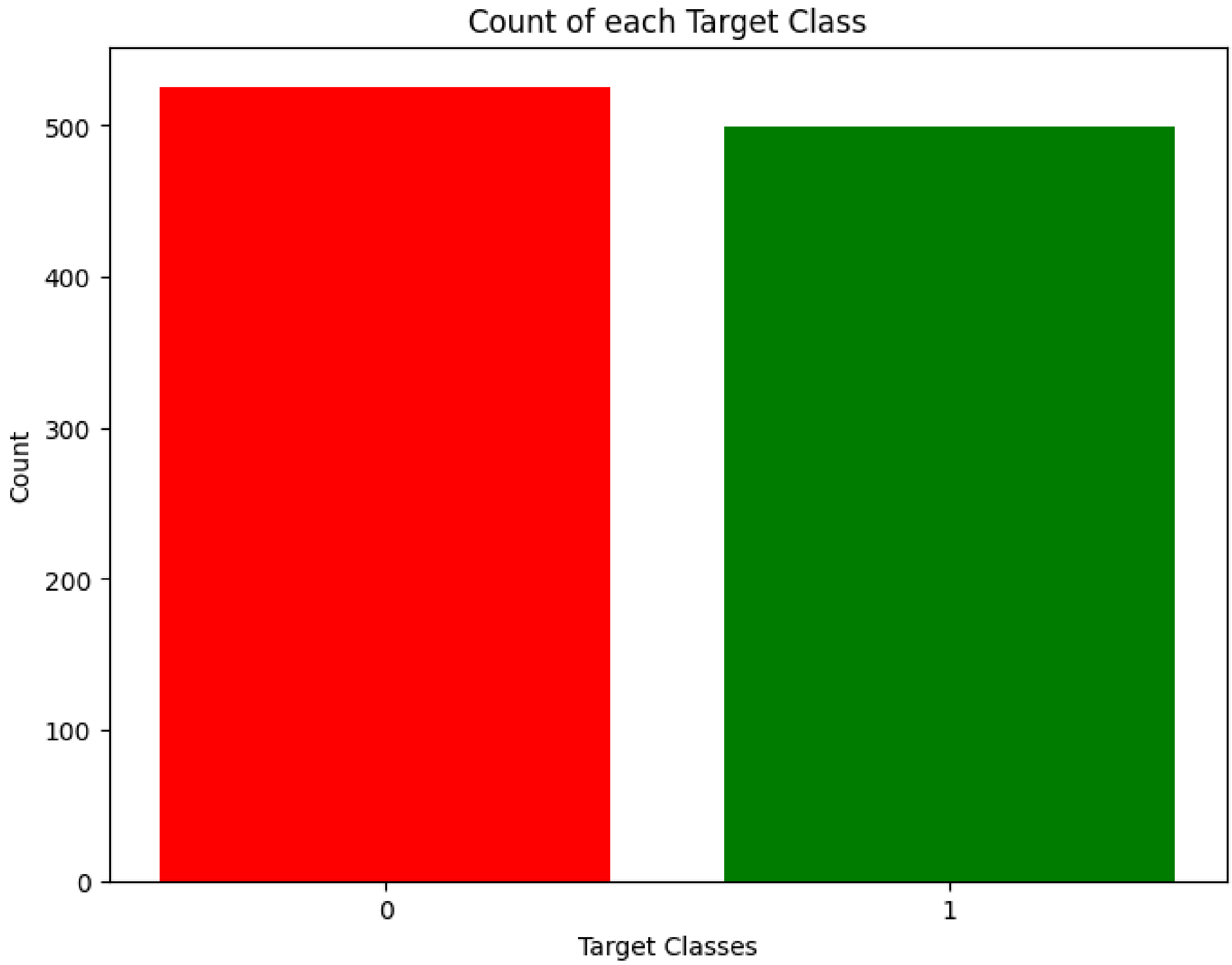
```
df.hist()
```

```
array([[<Axes: title={'center': 'age'}>, <Axes: title={'center': 'sex'}>,
        <Axes: title={'center': 'cp'}>,
        <Axes: title={'center': 'trestbps'}>],
       [<Axes: title={'center': 'chol'}>,
        <Axes: title={'center': 'fbs'}>,
        <Axes: title={'center': 'restecg'}>,
        <Axes: title={'center': 'thalach'}>],
       [<Axes: title={'center': 'exang'}>,
        <Axes: title={'center': 'oldpeak'}>,
        <Axes: title={'center': 'slope'}>,
        <Axes: title={'center': 'ca'}>],
       [<Axes: title={'center': 'thal'}>,
        <Axes: title={'center': 'target'}>, <Axes: >, <Axes: >]],
      dtype=object)
```



```
rcParams['figure.figsize'] = 8,6
plt.bar(df['target'].unique(), df['target'].value_counts(), color = ['red', 'green'])
plt.xticks([0, 1])
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class')
```

```
Text(0.5, 1.0, 'Count of each Target Class')
```



Data Processing

```
df = pd.get_dummies(df, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'c
```

```
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
df[columns_to_scale] = standardScaler.fit_transform(df[columns_to_scale])
```

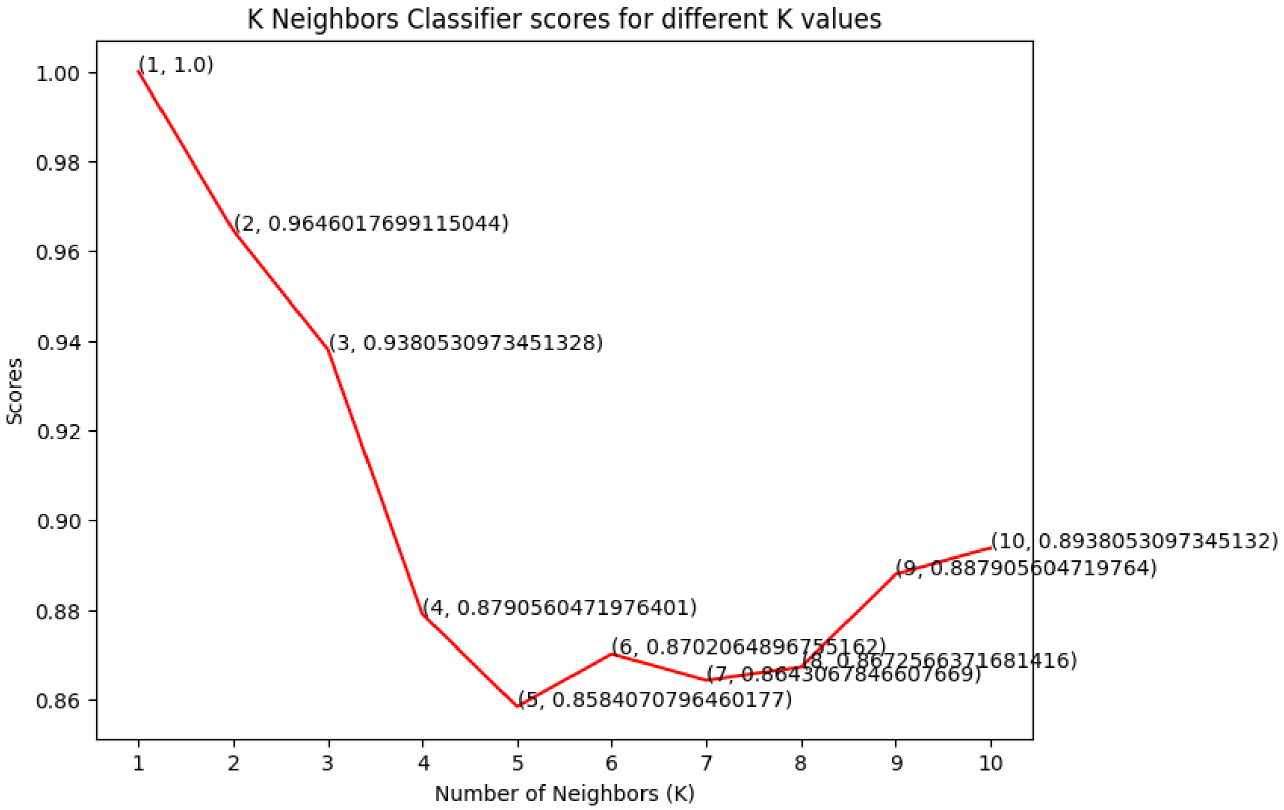
testing and training the data

```
y = df['target']
X = df.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_sta
```

```
knn_scores = []
for k in range(1,11):
    knn_classifier = KNeighborsClassifier(n_neighbors = k)
    knn_classifier.fit(X_train, y_train)
    knn_scores.append(knn_classifier.score(X_test, y_test))
```

```
plt.plot([k for k in range(1, 11)], knn_scores, color = 'red')
for i in range(1,11):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 11)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K values')
```

```
Text(0.5, 1.0, 'K Neighbors Classifier scores for different K values')
```



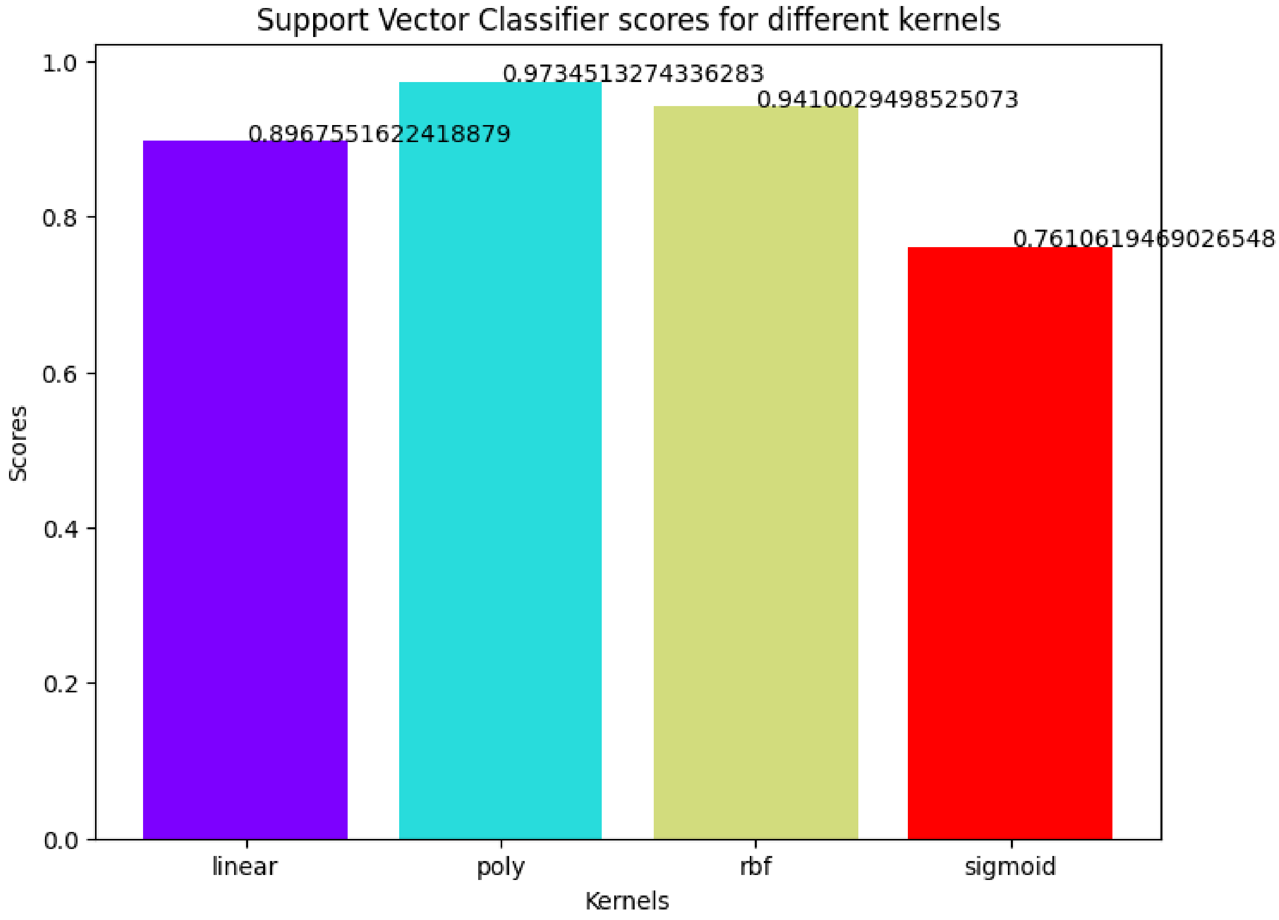
```
print("The score for K Neighbors Classifier is {}% with {} nieghbors.".format(knn_scor
```

```
The score for K Neighbors Classifier is 86.72566371681415% with 8 nieghbors.
```

```
svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    svc_classifier = SVC(kernel = kernels[i])
    svc_classifier.fit(X_train, y_train)
    svc_scores.append(svc_classifier.score(X_test, y_test))
```

```
colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
    plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')
plt.ylabel('Scores')
plt.title('Support Vector Classifier scores for different kernels')
```

```
Text(0.5, 1.0, 'Support Vector Classifier scores for different kernels')
```



```
print("The score for Support Vector Classifier is {}% with {} kernel.".format(svc_scor
```

```
The score for Support Vector Classifier is 89.67551622418878% with linear kernel.
```

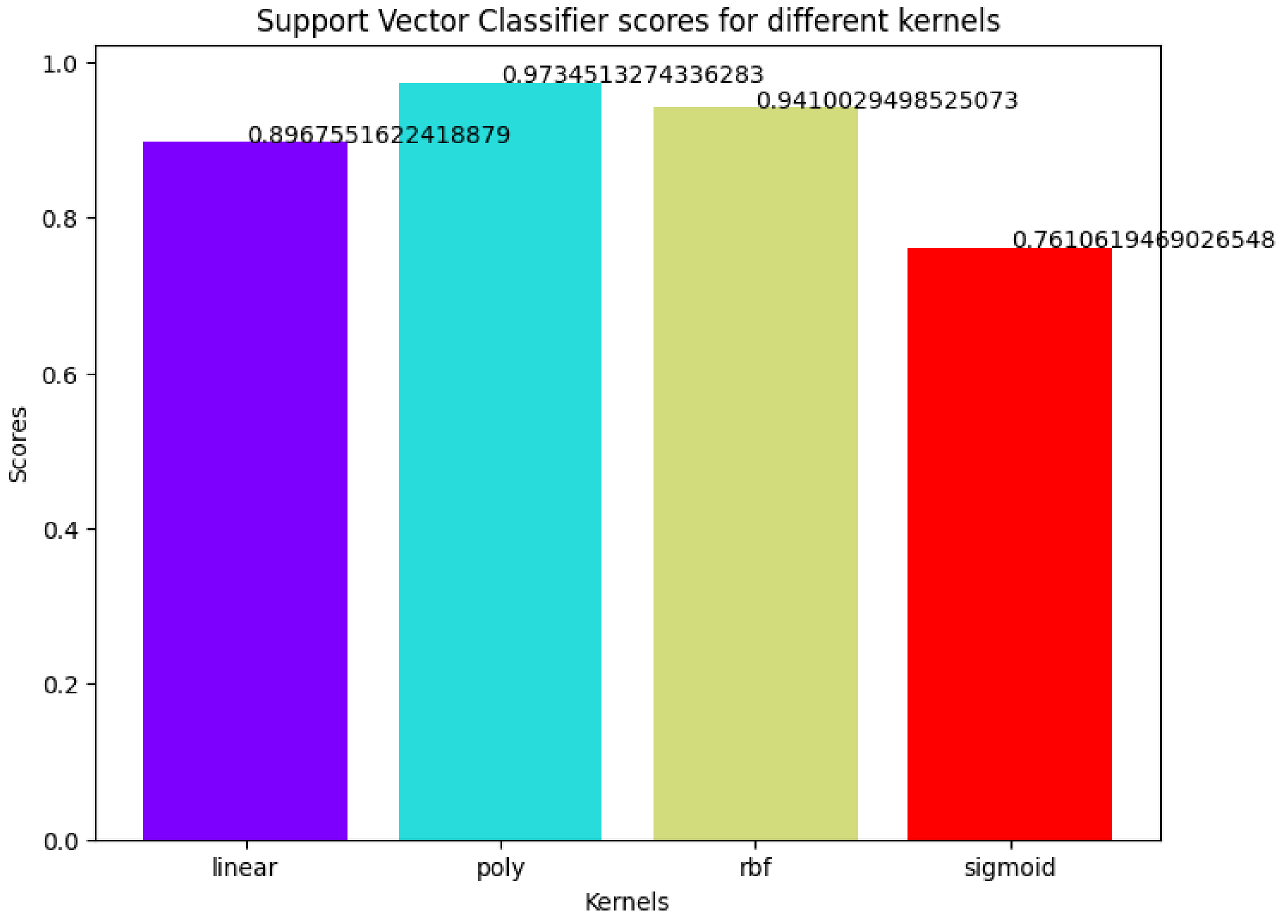
Support Vector Classifier

There are several kernels for Support Vector Classifier. I'll test some of them and check which has the best score.

```
from sklearn.svm import SVC
svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    svc_classifier = SVC(kernel = kernels[i])
    svc_classifier.fit(X_train, y_train)
    svc_scores.append(svc_classifier.score(X_test, y_test))
```

```
colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
    plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')
plt.ylabel('Scores')
plt.title('Support Vector Classifier scores for different kernels')
```

```
Text(0.5, 1.0, 'Support Vector Classifier scores for different kernels')
```



```
print("The score for Support Vector Classifier is {}% with {} kernel.".format(svc_scor
```

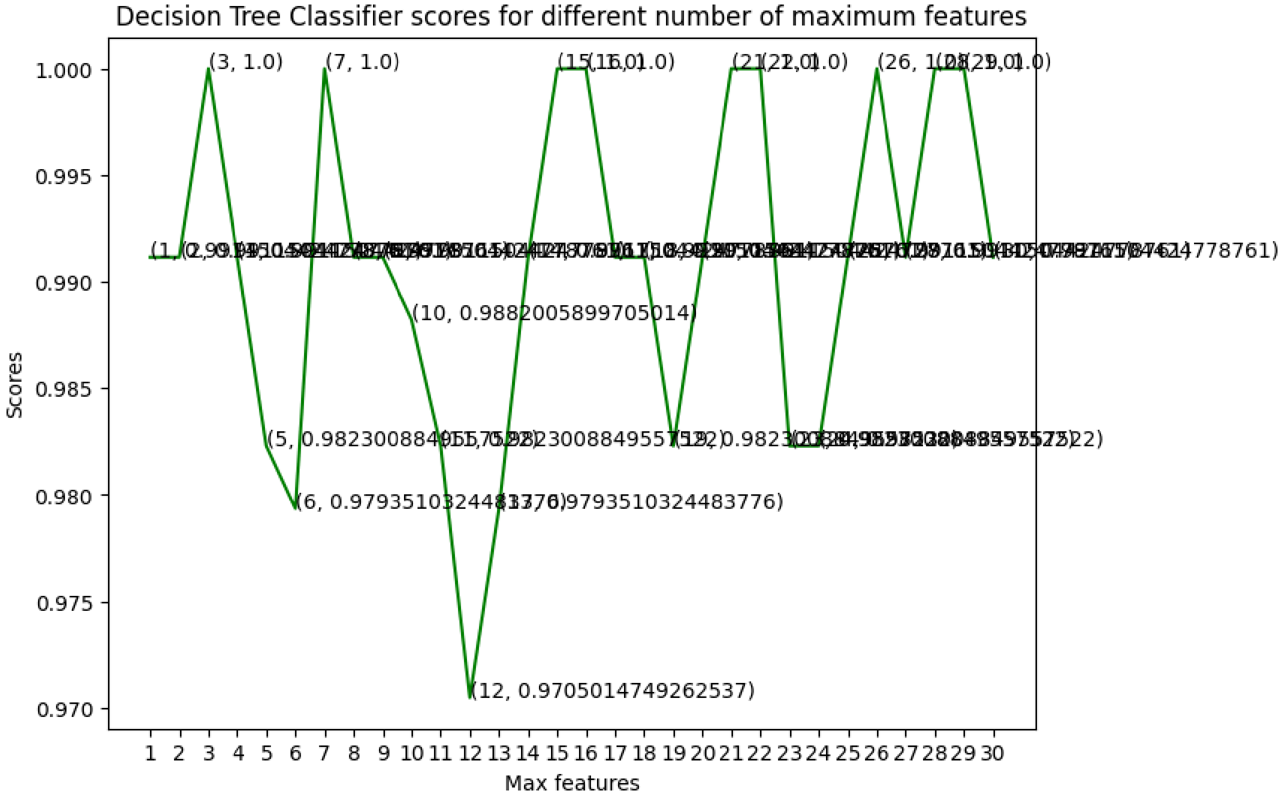
```
The score for Support Vector Classifier is 89.67551622418878% with linear kernel.
```

Decision Tree Classifier

```
dt_scores = []
for i in range(1, len(X.columns) + 1):
    dt_classifier = DecisionTreeClassifier(max_features = i, random_state = 0)
    dt_classifier.fit(X_train, y_train)
    dt_scores.append(dt_classifier.score(X_test, y_test))
```

```
plt.plot([i for i in range(1, len(X.columns) + 1)], dt_scores, color = 'green')
for i in range(1, len(X.columns) + 1):
    plt.text(i, dt_scores[i-1], (i, dt_scores[i-1]))
plt.xticks([i for i in range(1, len(X.columns) + 1)])
plt.xlabel('Max features')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different number of maximum features')
```

```
Text(0.5, 1.0, 'Decision Tree Classifier scores for different number of maximum
features')
```



```
print("The score for Decision Tree Classifier is {}% with {} maximum features.".format
```

```
The score for Decision Tree Classifier is 99.11504424778761% with [2, 4, 18] maximum
```

Random Forest Classifier

```
rf_scores = []
estimators = [10, 100, 200, 500, 1000]
for i in estimators:
    rf_classifier = RandomForestClassifier(n_estimators = i, random_state = 0)
    rf_classifier.fit(X_train, y_train)
```