In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.cm import rainbow
import seaborn as sns
import cufflinks as cf
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
heart_data=pd.read_csv('heart.csv')
heart_data.head()
```

Out[2]:

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

In [3]:
```
heart_data.shape
```

Out[3]: (1025, 14)

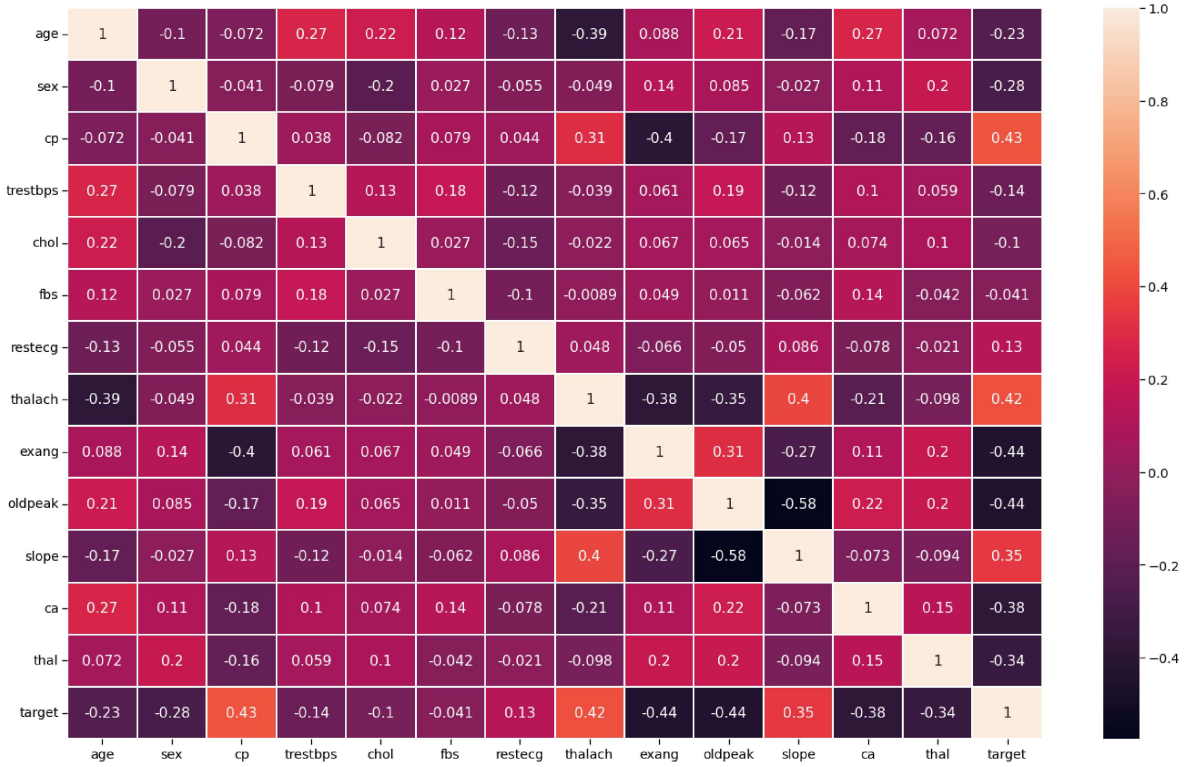
In [4]:
```
heart_data.describe()
```

Out[4]:

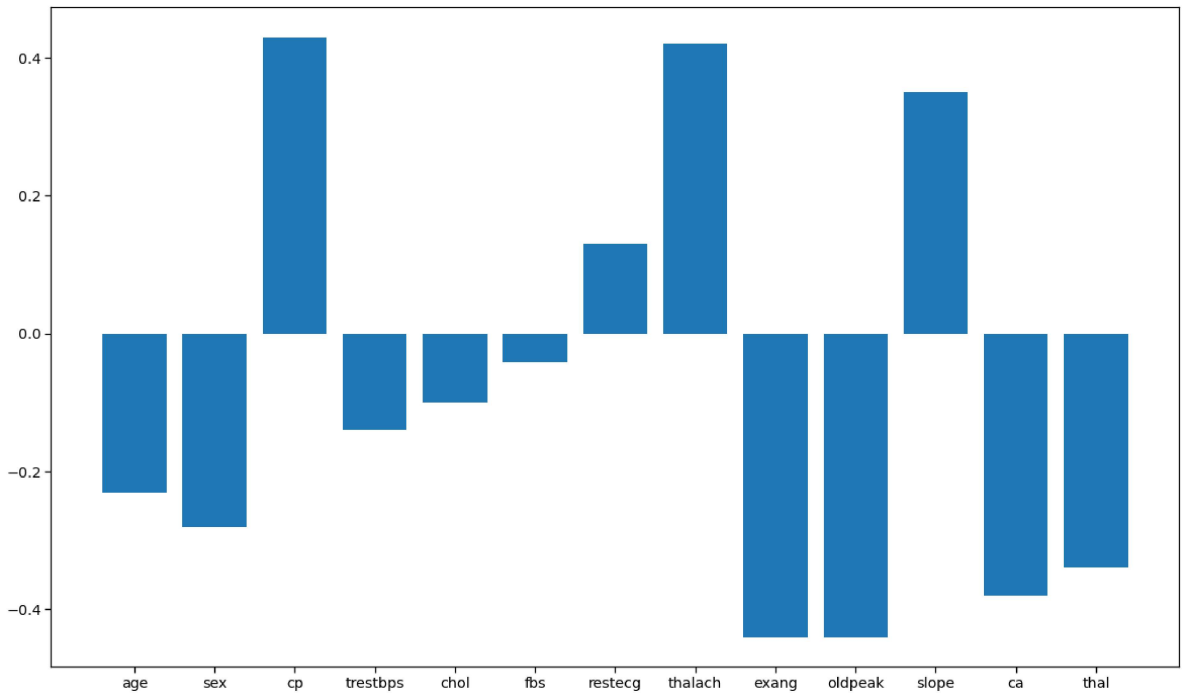
	age	sex	cp	trestbps	chol	fbs	restec
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.00000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.52975
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.52787
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.00000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.00000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.00000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.00000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.00000

Correlation

In [5]:
```
plt.figure(figsize=(20,12))
sns.set_context('notebook',font_scale = 1.3)
sns.heatmap(heart_data.corr(),annot=True,linewidth =2)
plt.tight_layout()
```

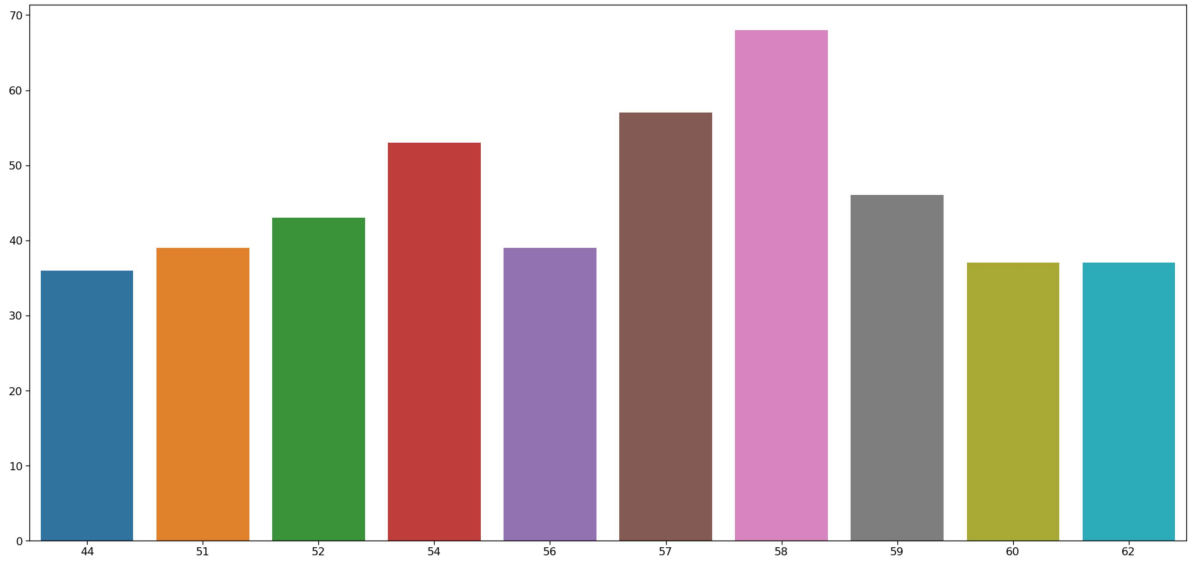


In [6]:
```
corr_target=[-0.23,-0.28,0.43,-0.14,-0.1,-0.041,0.13,0.42,-0.44,-0.44,0.3
features=['age','sex','cp','trestbps','chol','fbs','restecg','thalach','e
plt.figure(figsize=(20,12))
plt.bar(features,corr_target)
plt.show()
```



Age Analysis

In [8]:
```
plt.figure(figsize=(25,12))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=heart_data.age.value_counts()[:10].index,y=heart_data.age.v
plt.tight_layout()
```

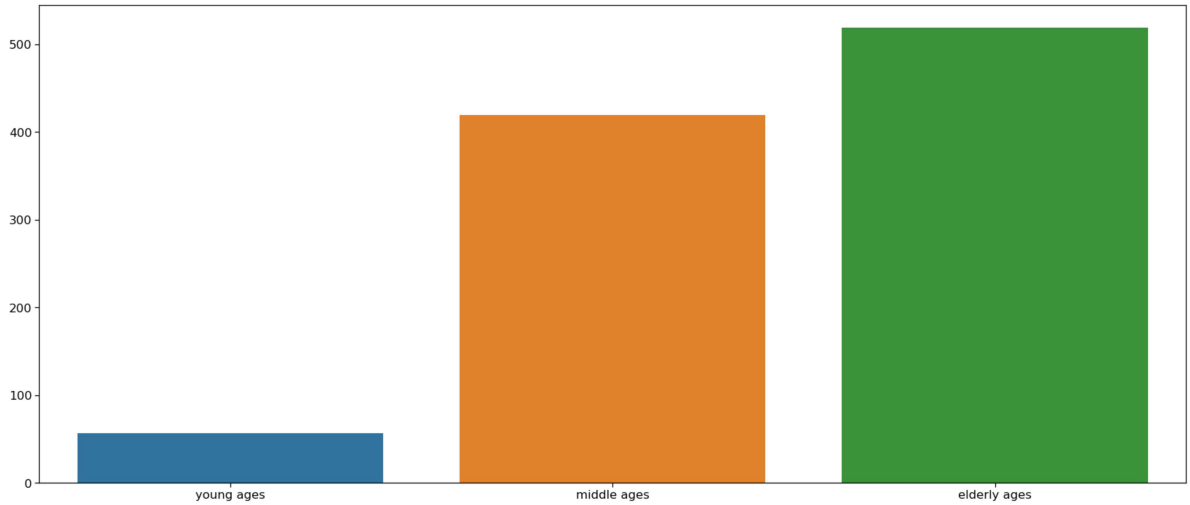


In [9]:
```
minAge=min(heart_data.age)
maxAge=max(heart_data.age)
meanAge=heart_data.age.mean()
print('Min Age :',minAge)
print('Max Age :',maxAge)
print('Mean Age :',meanAge)
```

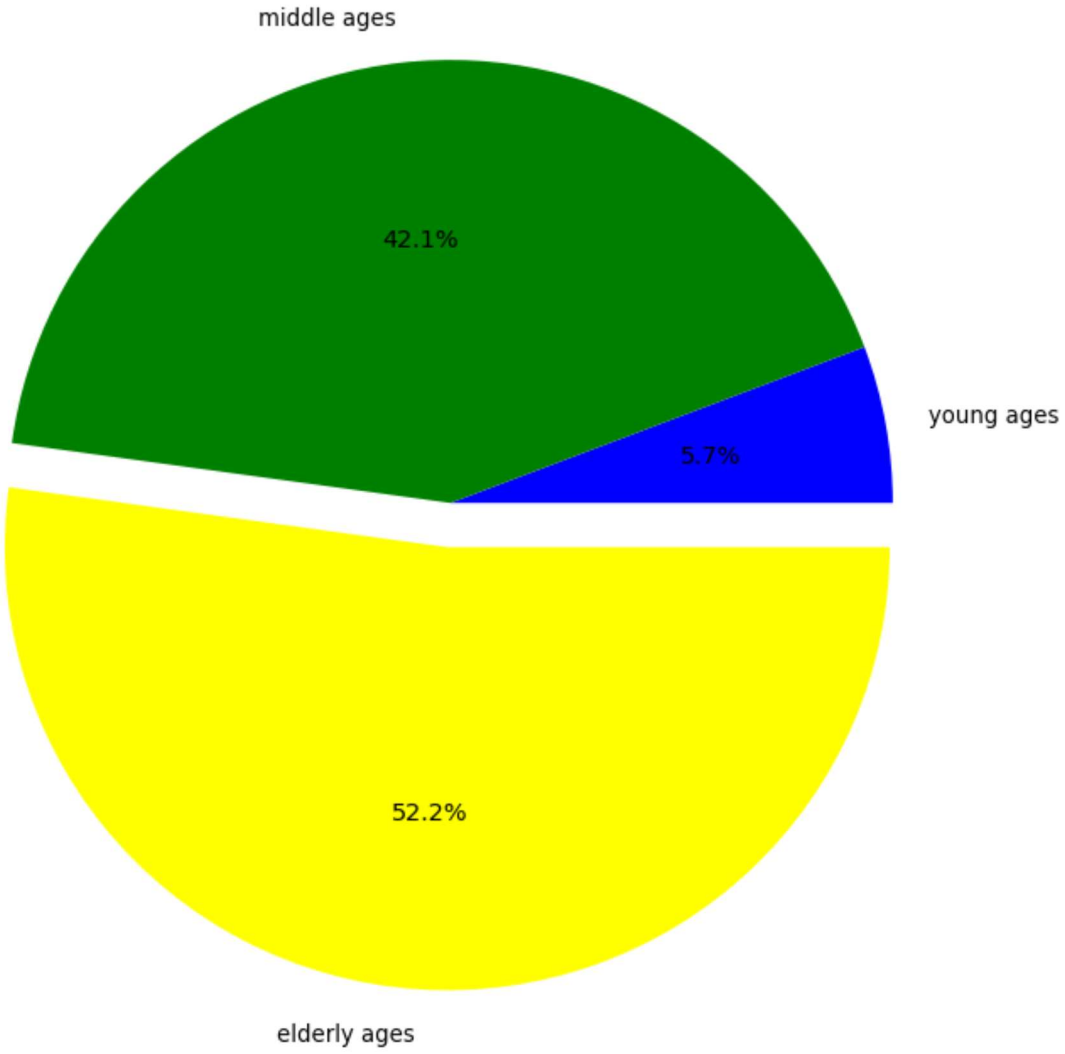
```
Min Age : 29
Max Age : 77
Mean Age : 54.43414634146342
```

In [10]:
```
Young = heart_data[(heart_data.age>=29)&(heart_data.age<40)]
Middle = heart_data[(heart_data.age>=40)&(heart_data.age<55)]
Elder = heart_data[(heart_data.age>55)]

plt.figure(figsize=(23,10))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=['young ages','middle ages','elderly ages'],y=[len(Young),l
plt.tight_layout()
```

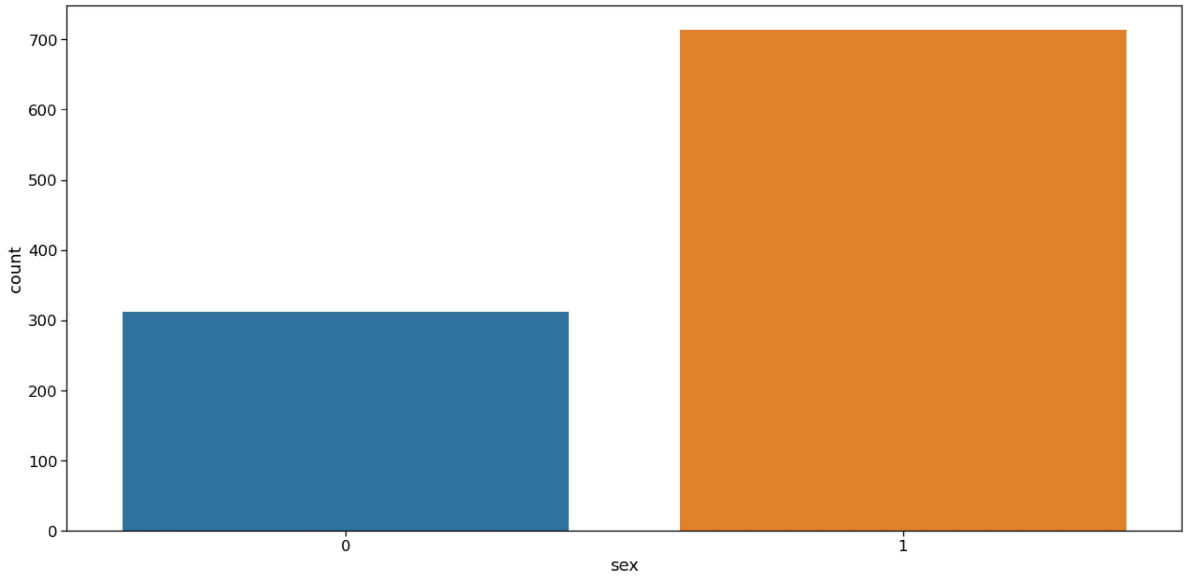


In [11]:
```
colors = ['blue','green','yellow']
explode = [0,0,0.1]
plt.figure(figsize=(10,10))
sns.set_context('notebook',font_scale = 1.2)
plt.pie([len(Young),len(Middle),len(Elder)],labels=['young ages','middle a
plt.tight_layout()
```

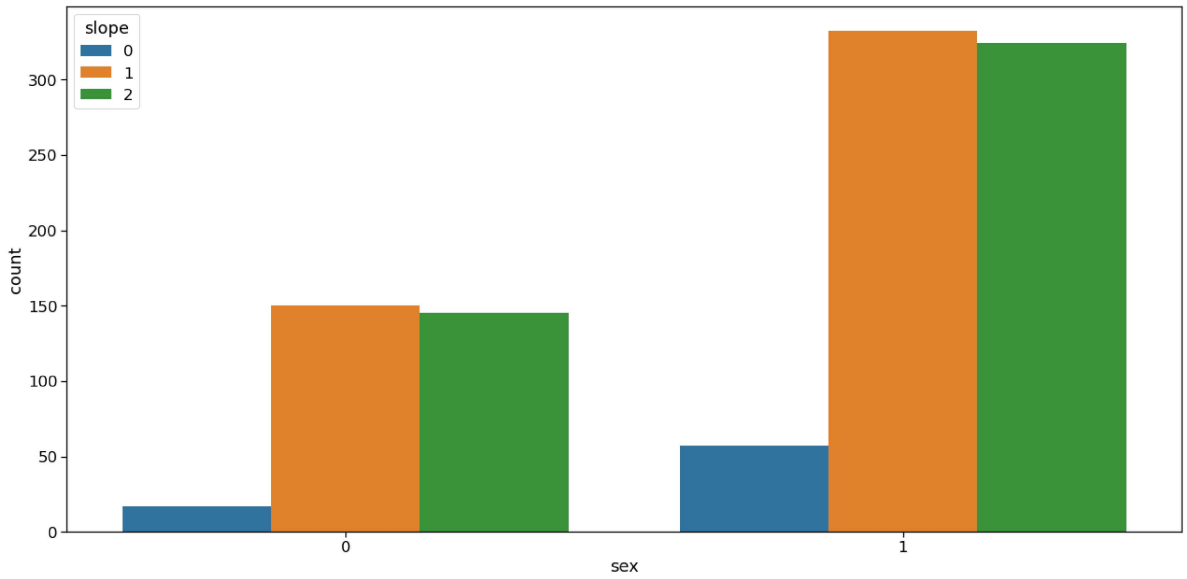


Sex Feature Analysis

In [12]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_data['sex'])
plt.tight_layout()
```

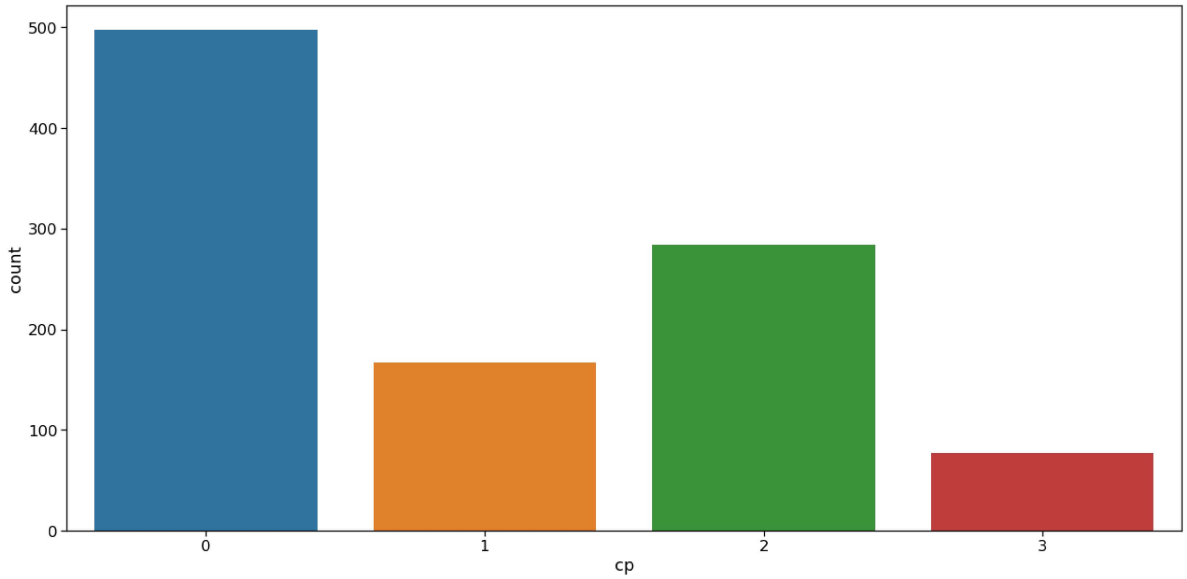


In [13]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_data['sex'],hue=heart_data["slope"])
plt.tight_layout()
```

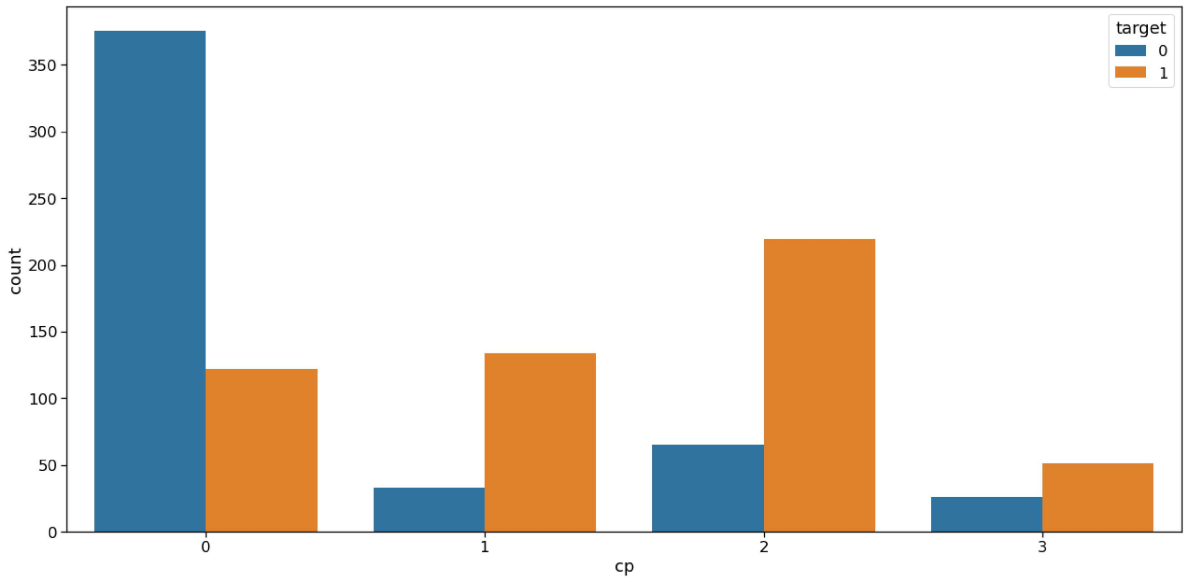


Chest Pain Type Analysis

In [14]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_data['cp'])
plt.tight_layout()
```

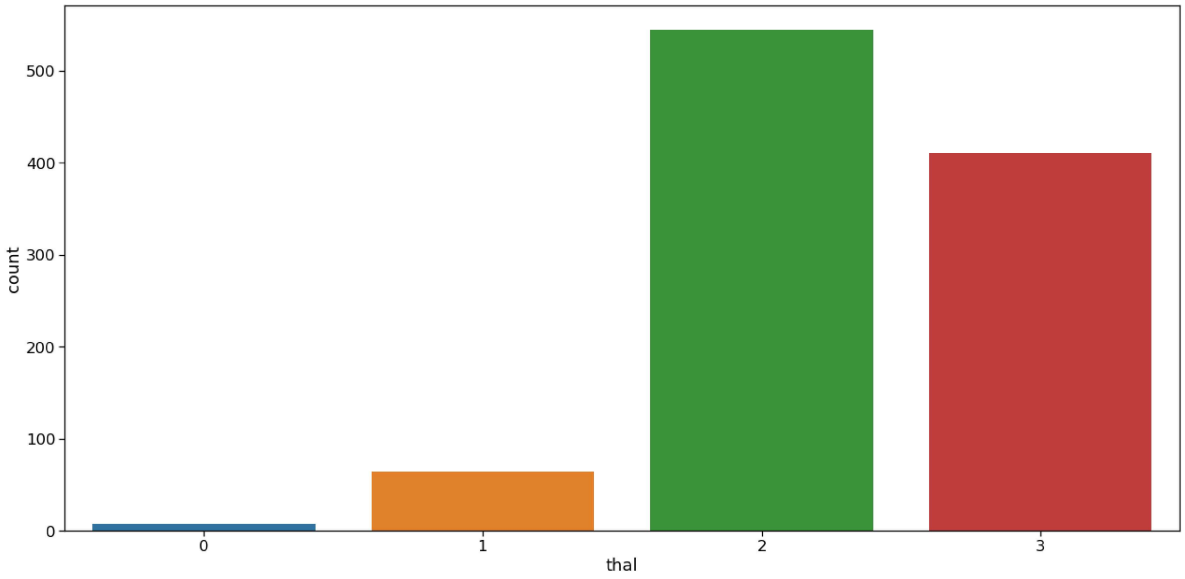


In [15]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_data['cp'],hue=heart_data["target"])
plt.tight_layout()
```



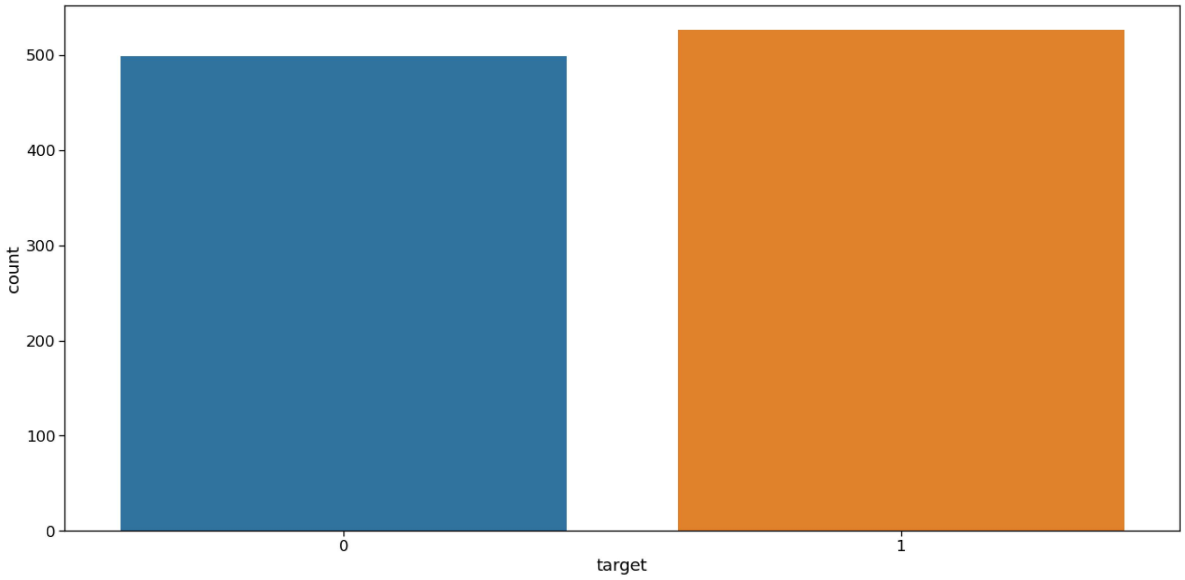
Thal Analysis

In [16]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_data['thal'])
plt.tight_layout()
```



Target

In [17]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_data['target'])
plt.tight_layout()
```



Preprocessing

In [18]:
```
from sklearn.preprocessing import StandardScaler
heart_data = pd.get_dummies(heart_data, columns = ['sex', 'cp', 'fbs', 'r
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
heart_data[columns_to_scale] = standardScaler.fit_transform(heart_data[co
```

Splting Dataset

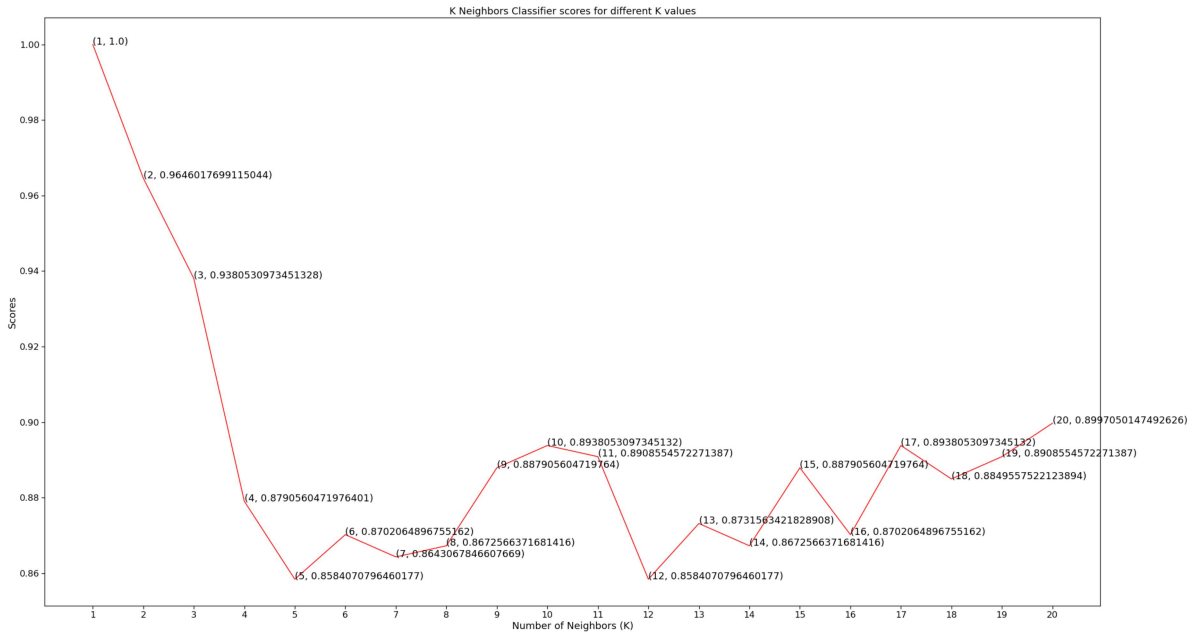
In [20]:
```
from sklearn.model_selection import train_test_split
y = heart_data['target']
X = heart_data.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3
```

KNN Classifier

In [21]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_scores = []
for k in range(1,21):
    knn_classifier = KNeighborsClassifier(n_neighbors = k)
    knn_classifier.fit(X_train, y_train)
    knn_scores.append(knn_classifier.score(X_test, y_test))
```

In [22]:
```
plt.figure(figsize=(35,20))
sns.set_context('notebook',font_scale = 1.5)
plt.plot([k for k in range(1, 21)], knn_scores, color = 'red')
for i in range(1,21):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K values')
```

Out[22]: Text(0.5, 1.0, 'K Neighbors Classifier scores for different K values')

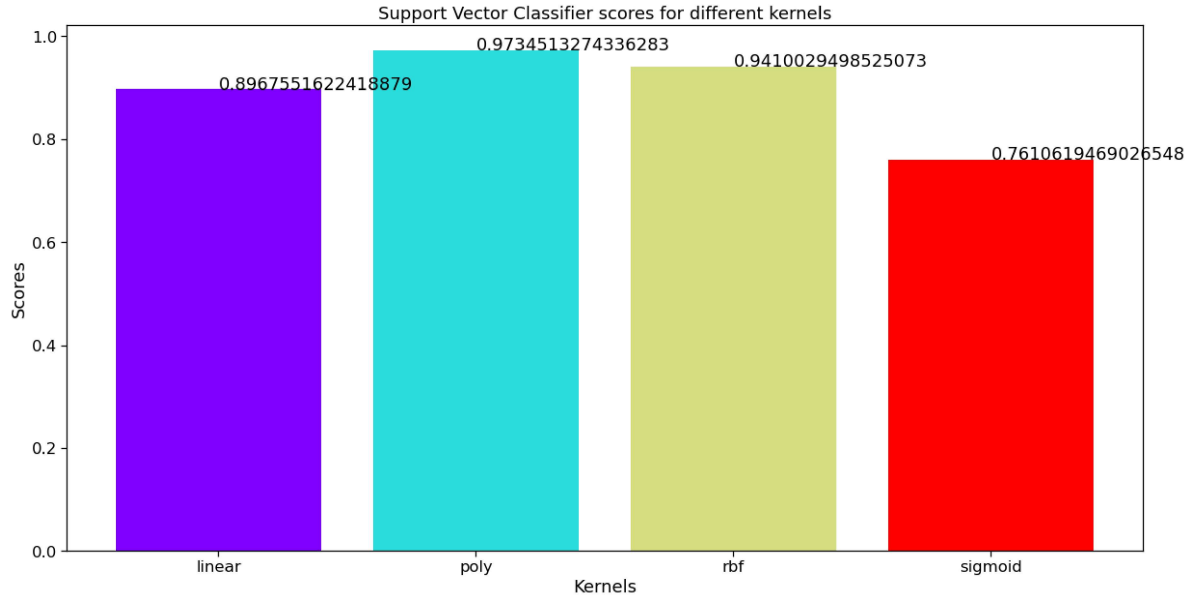


Support Vector Machine

In [23]:
```
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))
```

In [24]:
```
plt.figure(figsize=(20,10))
sns.set_context('notebook',font_scale = 1.5)
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')
```

Out[24]: Text(0.5, 1.0, 'Support Vector Classifier scores for different kernels')

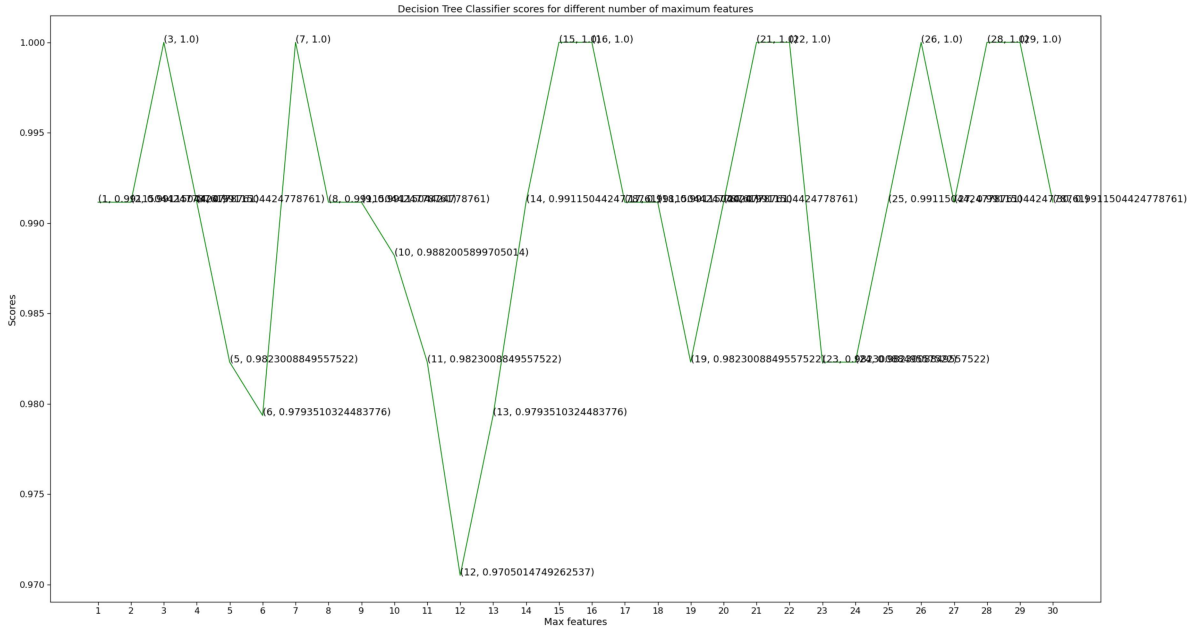


Decision Tree Classifier

In [25]:
```
from sklearn.tree import DecisionTreeClassifier
dt_scores = []
for i in range(1, len(X.columns) + 1):
    dt_classifier = DecisionTreeClassifier(max_features = i, random_state
    dt_classifier.fit(X_train, y_train)
    dt_scores.append(dt_classifier.score(X_test, y_test))
```

In [26]:
```
plt.figure(figsize=(35,20))
sns.set_context('notebook',font_scale = 1.5)
plt.plot([i for i in range(1, len(X.columns) + 1)], dt_scores, color = 'g
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 maximur
```

Out[26]:
```
Text(0.5, 1.0, 'Decision Tree Classifier scores for different number of maxim
um features')
```

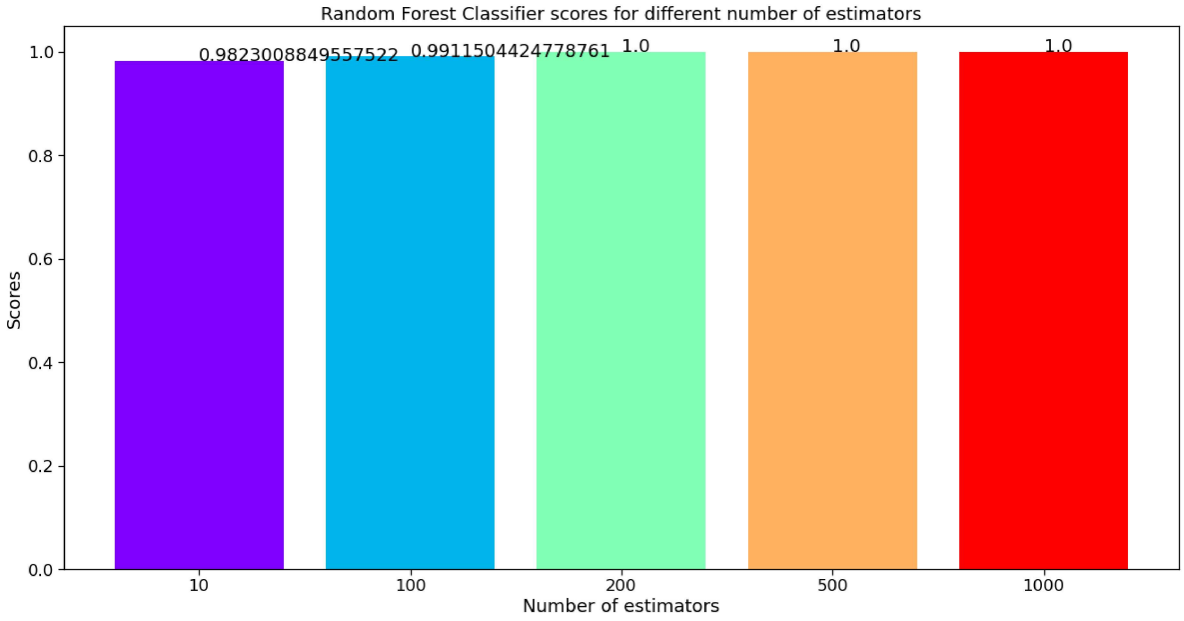


RandomForest Classifier

In [27]:
```
from sklearn.ensemble import RandomForestClassifier
rf_scores = []
estimators = [10, 100, 200, 500, 1000]
for i in estimators:
    rf_classifier = RandomForestClassifier(n_estimators = i, random_state
    rf_classifier.fit(X_train, y_train)
    rf_scores.append(rf_classifier.score(X_test, y_test))
```

In [28]:
```
plt.figure(figsize=(20,10))
sns.set_context('notebook',font_scale = 1.5)
colors = rainbow(np.linspace(0, 1, len(estimators)))
plt.bar([i for i in range(len(estimators))], rf_scores, color = colors, w
for i in range(len(estimators)):
    plt.text(i, rf_scores[i], rf_scores[i])
plt.xticks(ticks = [i for i in range(len(estimators))], labels = [str(est
plt.xlabel('Number of estimators')
plt.ylabel('Scores')
plt.title('Random Forest Classifier scores for different number of estimat
```

Out[28]:
```
Text(0.5, 1.0, 'Random Forest Classifier scores for different number of estim
ators')
```



Naive Bayes Classifier

In [30]:
```
from sklearn.naive_bayes import GaussianNB
nb_classifier=GaussianNB()
nb_classifier.fit(X_train,y_train)
```

Out[30]:
```
GaussianNB()
```

In [32]:
```
y_pred=nb_classifier.predict(X_test)
```

In [34]:
```
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test,y_pred)
print("Accuracy Score by Naive Bayes classifier is:",ac)
```

```
Accuracy Score by Naive Bayes classifier is: 0.8761061946902655
```