Project By : PRASAD JADHAV

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from imblearn.over_sampling import SMOTE

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import svm

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
dataset = pd.read_csv('heart.csv')
dataset.shape
```

Out[3]:
```
(1025, 14)
```

In [4]:
```
dataset.head()
```

Out[4]:

	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 [5]:
```
dataset.tail()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

In [6]:
```
dataset.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
```

In [7]:
```
dataset.describe()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000

In [8]:
```
dataset.corr()
```

Out[8]:

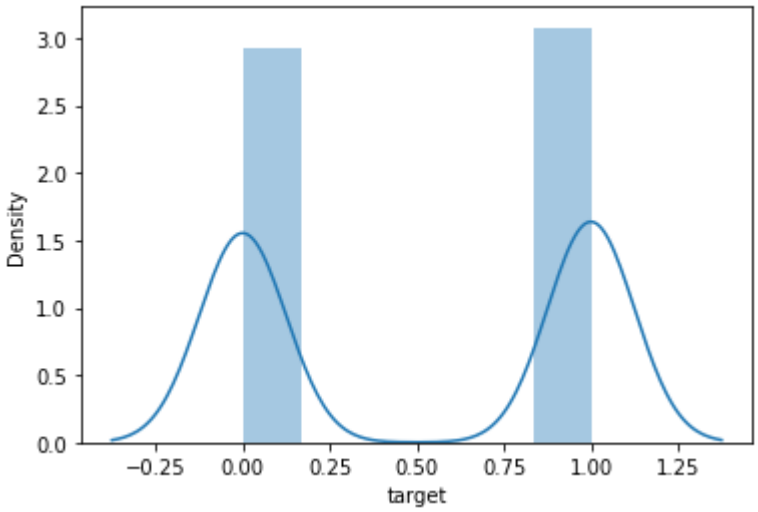
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	ex
age	1.000000	-0.103240	-0.071966	0.271121	0.219823	0.121243	-0.132696	-0.390227	0.08
sex	-0.103240	1.000000	-0.041119	-0.078974	-0.198258	0.027200	-0.055117	-0.049365	0.13
cp	-0.071966	-0.041119	1.000000	0.038177	-0.081641	0.079294	0.043581	0.306839	-0.40
trestbps	0.271121	-0.078974	0.038177	1.000000	0.127977	0.181767	-0.123794	-0.039264	0.06
chol	0.219823	-0.198258	-0.081641	0.127977	1.000000	0.026917	-0.147410	-0.021772	0.06
fbs	0.121243	0.027200	0.079294	0.181767	0.026917	1.000000	-0.104051	-0.008866	0.04
restecg	-0.132696	-0.055117	0.043581	-0.123794	-0.147410	-0.104051	1.000000	0.048411	-0.06
thalach	-0.390227	-0.049365	0.306839	-0.039264	-0.021772	-0.008866	0.048411	1.000000	-0.38
exang	0.088163	0.139157	-0.401513	0.061197	0.067382	0.049261	-0.065606	-0.380281	1.00
oldpeak	0.208137	0.084687	-0.174733	0.187434	0.064880	0.010859	-0.050114	-0.349796	0.31
slope	-0.169105	-0.026666	0.131633	-0.120445	-0.014248	-0.061902	0.086086	0.395308	-0.26
ca	0.271551	0.111729	-0.176206	0.104554	0.074259	0.137156	-0.078072	-0.207888	0.10
thal	0.072297	0.198424	-0.163341	0.059276	0.100244	-0.042177	-0.020504	-0.098068	0.19
target	-0.229324	-0.279501	0.434854	-0.138772	-0.099966	-0.041164	0.134468	0.422895	-0.43

In [9]:
```
dataset.cov()
```

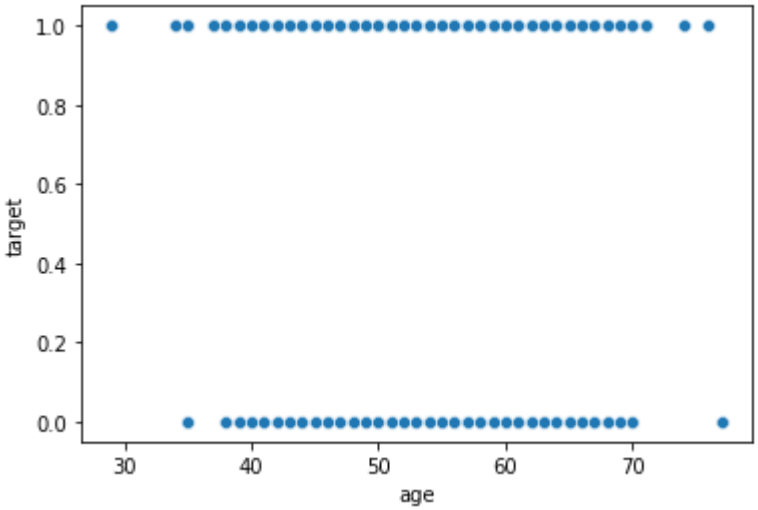
Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
age	82.306450	-0.431198	-0.672251	43.085733	102.890625	0.392164	-0.635490	-81.44608
sex	-0.431198	0.211944	-0.019491	-0.636863	-4.708984	0.004465	-0.013395	-0.52283
cp	-0.672251	-0.019491	1.060160	0.688565	-4.336914	0.029108	0.023687	7.26829
trestbps	43.085733	-0.636863	0.688565	306.835410	115.657227	1.135165	-1.144685	-15.82282
chol	102.890625	-4.708984	-4.336914	115.657227	2661.787109	0.495117	-4.014648	-25.84179
fbs	0.392164	0.004465	0.029108	1.135165	0.495117	0.127111	-0.019583	-0.07271
restecg	-0.635490	-0.013395	0.023687	-1.144685	-4.014648	-0.019583	0.278655	0.58790
thalach	-81.446089	-0.522838	7.268296	-15.822822	-25.841797	-0.072719	0.587909	529.26332
exang	0.378144	0.030288	-0.195451	0.506798	1.643555	0.008303	-0.016373	-4.13611
oldpeak	2.218825	0.045812	-0.211407	3.857971	3.933301	0.004549	-0.031085	-9.45602
slope	-0.947742	-0.007584	0.083727	-1.303344	-0.454102	-0.013634	0.028073	5.61807
ca	2.539458	0.053021	-0.187017	1.887842	3.949219	0.050406	-0.042482	-4.92991
thal	0.407093	0.056697	-0.104385	0.644446	3.209961	-0.009333	-0.006718	-1.40029
target	-1.040392	-0.064346	0.223903	-1.215584	-2.579102	-0.007339	0.035496	4.86519

In [10]:
```
sns.distplot(dataset['target'])
plt.show()
```



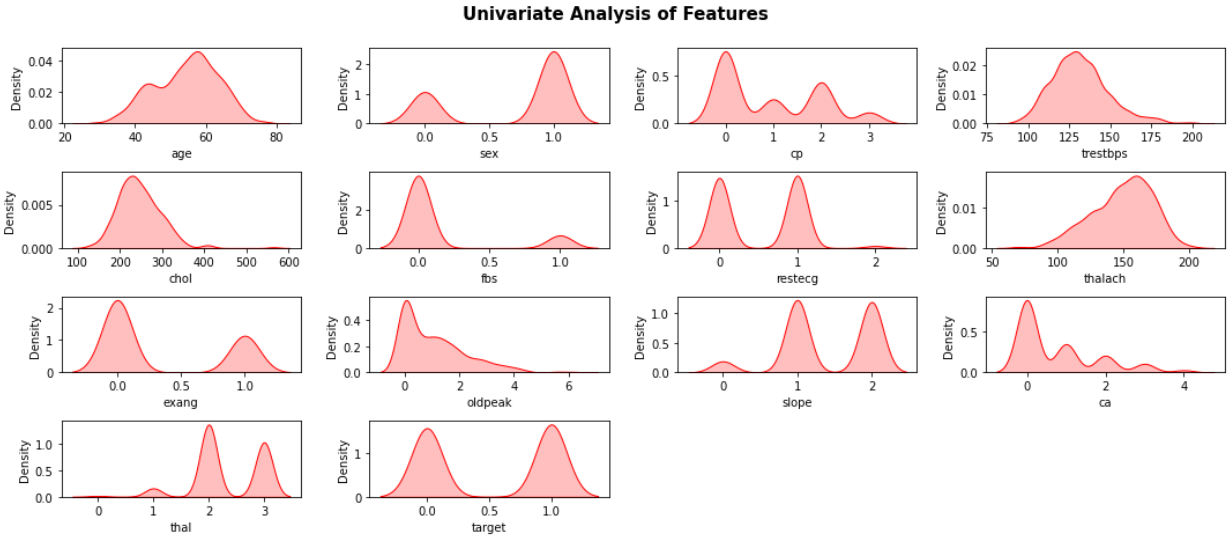
In [11]:
```
sns.scatterplot(dataset['age'],dataset['target'])
plt.show()
```



In [12]:
```
features = [feature for feature in dataset.columns]
```

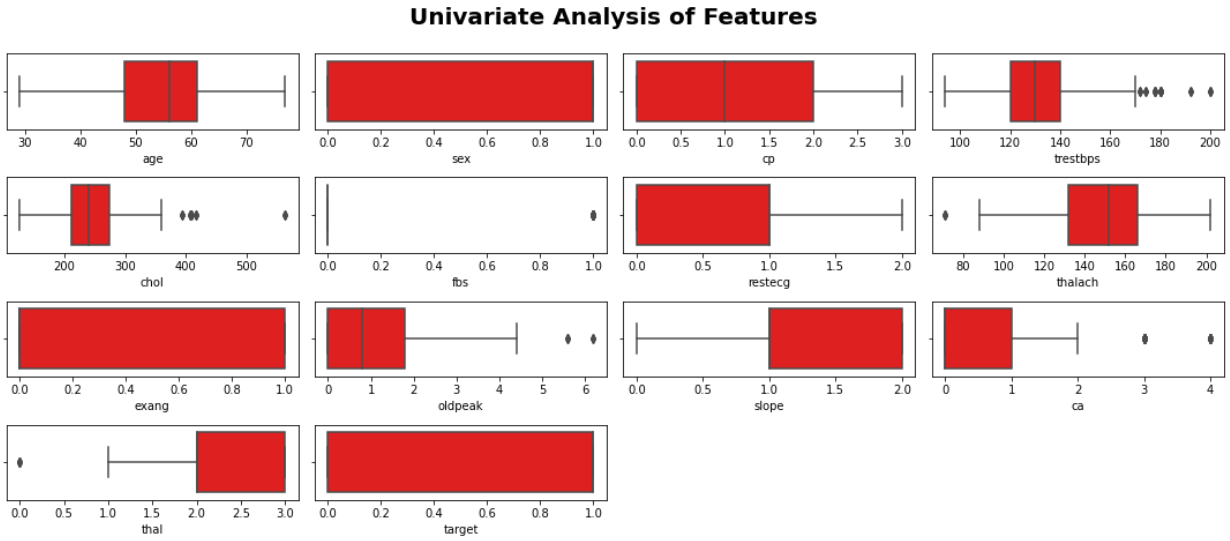
In [13]:
```
plt.figure(figsize=(15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=15,y=

for i in range(0,len(features)):
    plt.subplot(10,4,i+1)
    sns.kdeplot(x=dataset[features[i]],shade=True,color='red')
    plt.tight_layout()
```



In [14]:
```
plt.figure(figsize = (15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=20,y=

for i in range(0,len(features)):
    plt.subplot(10,4,i+1)
    sns.boxplot(data=dataset,x=features[i],color='red')
    plt.xlabel(features[i])
    plt.tight_layout()
```



In [ ]:
```
def remove_outliers(in_dataset, in_cols):

    first_quartile = in_dataset[in_cols].quantile(0.25)
    third_quartile = in_dataset[in_cols].quantile(0.75)
    iqr = third_quartile - first_quartile
    upper_limit = third_quartile + 1.5 * iqr
    lower_limit = first_quartile - 1.5 * iqr
    in_dataset.loc[(in_dataset[in_cols] > upper_limit), in_cols] = upper_limit
    in_dataset.loc[(in_dataset[in_cols] < lower_limit), in_cols] = lower_limit
    return in_dataset
```

In [ ]:
```
for features in features:
    dataset = remove_outliers(dataset,features)
```

In [15]:
```
dataset['target'].value_counts()
```

Out[15]:
```
1    526
0    499
Name: target, dtype: int64
```

In [16]:
```
X = dataset.drop('target',axis=1)
y = dataset['target']
```

In [17]:
```
X_res,y_res = SMOTE().fit_resample(X,y)
```

In [18]:
```
X_train,X_test,y_train,y_test = train_test_split(X_res,y_res,test_size=0.20,ra
```

In [19]:
```
st = StandardScaler()
X_train = st.fit_transform(X_train)
X_test = st.fit_transform(X_test)
```

In [20]:
```
model_df = {}

def model_val(model,X,y):
    X_train,X_test,y_train,y_test = train_test_split(X_res,y_res,test_size=0.2(
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    print(f'{model} Accuracy is {accuracy_score(y_test,y_pred)}')

    score = cross_val_score(model,X,y,cv=5,n_jobs=-1)
```

```
    print(f'{model} Average cross val score is {np.mean(score)}')
    model_df[model] = round(np.mean(score)*100,2)
```

In [21]:
```
model = LogisticRegression()
model_val(model,X,y)
```

```
LogisticRegression() Accuracy is 0.8388625592417062
LogisticRegression() Average cross val score is 0.8429268292682928
```

In [22]:
```
model = DecisionTreeClassifier()
model_val(model,X,y)
```

```
DecisionTreeClassifier() Accuracy is 1.0
DecisionTreeClassifier() Average cross val score is 0.9931707317073171
```

In [23]:
```
model = RandomForestClassifier()
model_val(model,X,y)
```

```
RandomForestClassifier() Accuracy is 1.0
RandomForestClassifier() Average cross val score is 0.9970731707317073
```

In [24]:
```
model = GradientBoostingClassifier()
model_val(model,X,y)
```

```
GradientBoostingClassifier() Accuracy is 0.957345971563981
GradientBoostingClassifier() Average cross val score is 0.9639024390243902
```

In [25]:
```
model_df
```

Out[25]:
```
{LogisticRegression(): 84.29,
 DecisionTreeClassifier(): 99.32,
 RandomForestClassifier(): 99.71,
 GradientBoostingClassifier(): 96.39}
```

In [26]:
```
import pickle
import joblib
```

In [27]:
```
pickle.dump(model,open('heart_disease_prediction.pkl','wb'))
```

In [28]:
```
model = pickle.load(open('heart_disease_prediction.pkl','rb'))
```

In [30]:
```
new_df = pd.DataFrame({
    'age':52,
    'sex':1,
    'cp':0,
    'trestbps':125,
    'chol':212,
    'fbs':0,
    'restecg':1,
    'thalach':168,
    'exang': 0,
    'oldpeak':1.0,
    'slope':2,
    'ca':2,
    'thal':3
},index=[0])
```

In [31]:
```
model.predict(new_df)
```

Out[31]:
```
array([0], dtype=int64)
```

In [32]:
```
p = model.predict(new_df)

prob = model.predict_proba(new_df)
if p == 1:
    print('Heart Disease!')
    print(f'You will be Heart Disease! with Probability of {prob[0][1]:.2f}')
else:
    print('Not-Heart Disease!')
```

```
Not-Heart Disease!
```