Predicting Heart Disease using Machine Learning

Importing Libraries

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

# Libraries for visualisation
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')

# Libraries for Preprocessing
from sklearn.preprocessing import StandardScaler, OneHotEncoder, OrdinalEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.compose import ColumnTransformer

# Machine Learning Algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBo
from sklearn.neighbors import KNeighborsClassifier
# Libraries for Metrics
from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import classification_report,roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
```

Reading the data set

In [2]:
```
# Reading the data set
data = pd.read_csv(r'C:\Users\ASUS\Downloads\heart (1).csv')
print("Number of rows in the dataset: {}".format(data.shape[0]))
print("Number of cols in the dataset: {}".format(data.shape[1]))
```

```
Number of rows in the dataset: 918
Number of cols in the dataset: 12
```

In [3]:
```
data.head()
```

Out[3]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina
0	40	M	ATA	140	289	0	Normal	172	N
1	49	F	NAP	160	180	0	Normal	156	N
2	37	M	ATA	130	283	0	ST	98	N
3	48	F	ASY	138	214	0	Normal	108	Y
4	54	M	NAP	150	195	0	Normal	122	N

Exploratory Data Analysis

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [ ]:
```

```

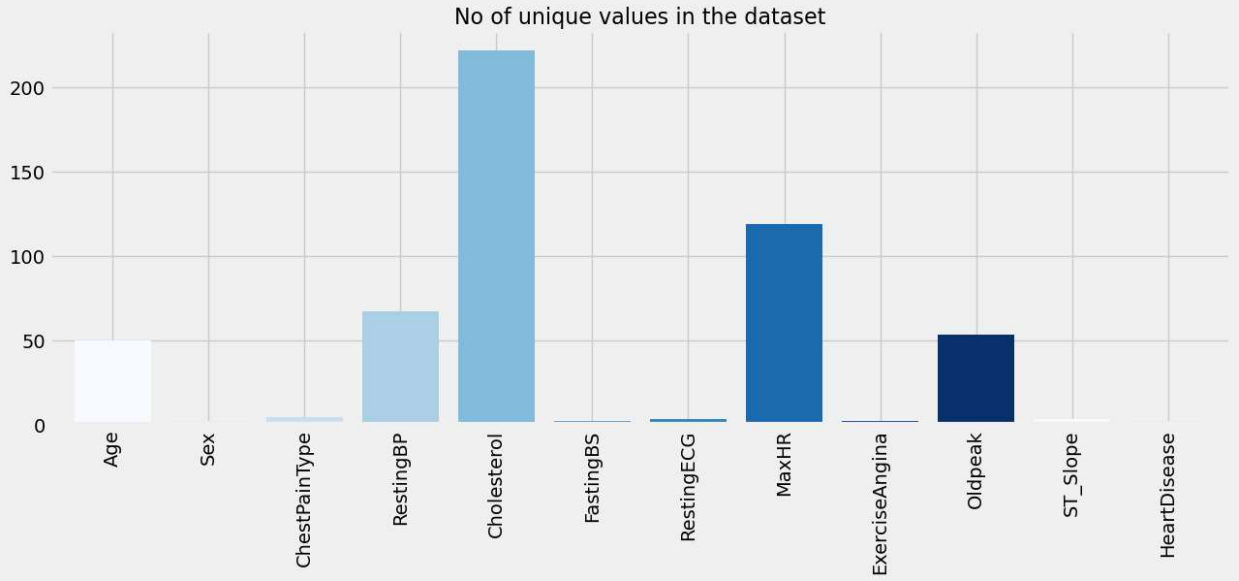
In [5]:
```
data.describe().T.style.background_gradient(cmap='Blues')
```

Out[5]:

	count	mean	std	min	25%	50%	75%	
Age	918.000000	53.510893	9.432617	28.000000	47.000000	54.000000	60.000000	77.
RestingBP	918.000000	132.396514	18.514154	0.000000	120.000000	130.000000	140.000000	200.
Cholesterol	918.000000	198.799564	109.384145	0.000000	173.250000	223.000000	267.000000	603.
FastingBS	918.000000	0.233115	0.423046	0.000000	0.000000	0.000000	0.000000	1.
MaxHR	918.000000	136.809368	25.460334	60.000000	120.000000	138.000000	156.000000	202.
Oldpeak	918.000000	0.887364	1.066570	-2.600000	0.000000	0.600000	1.500000	6.
HeartDisease	918.000000	0.553377	0.497414	0.000000	0.000000	1.000000	1.000000	1.

Cardinality of the data

In [6]:
```
color = plt.cm.Blues(np.linspace(0, 1, 10))
plt.figure(figsize=(14,5))
data.nunique().plot(kind='bar', width=.8, color=color)
plt.title('No of unique values in the dataset', size=16)
plt.show()
print(data.nunique())
```




```
Age               50
Sex                2
ChestPainType      4
RestingBP         67
Cholesterol      222
FastingBS          2
RestingECG         3
MaxHR            119
ExerciseAngina     2
Oldpeak           53
ST_Slope           3
HeartDisease       2
dtype: int64
```

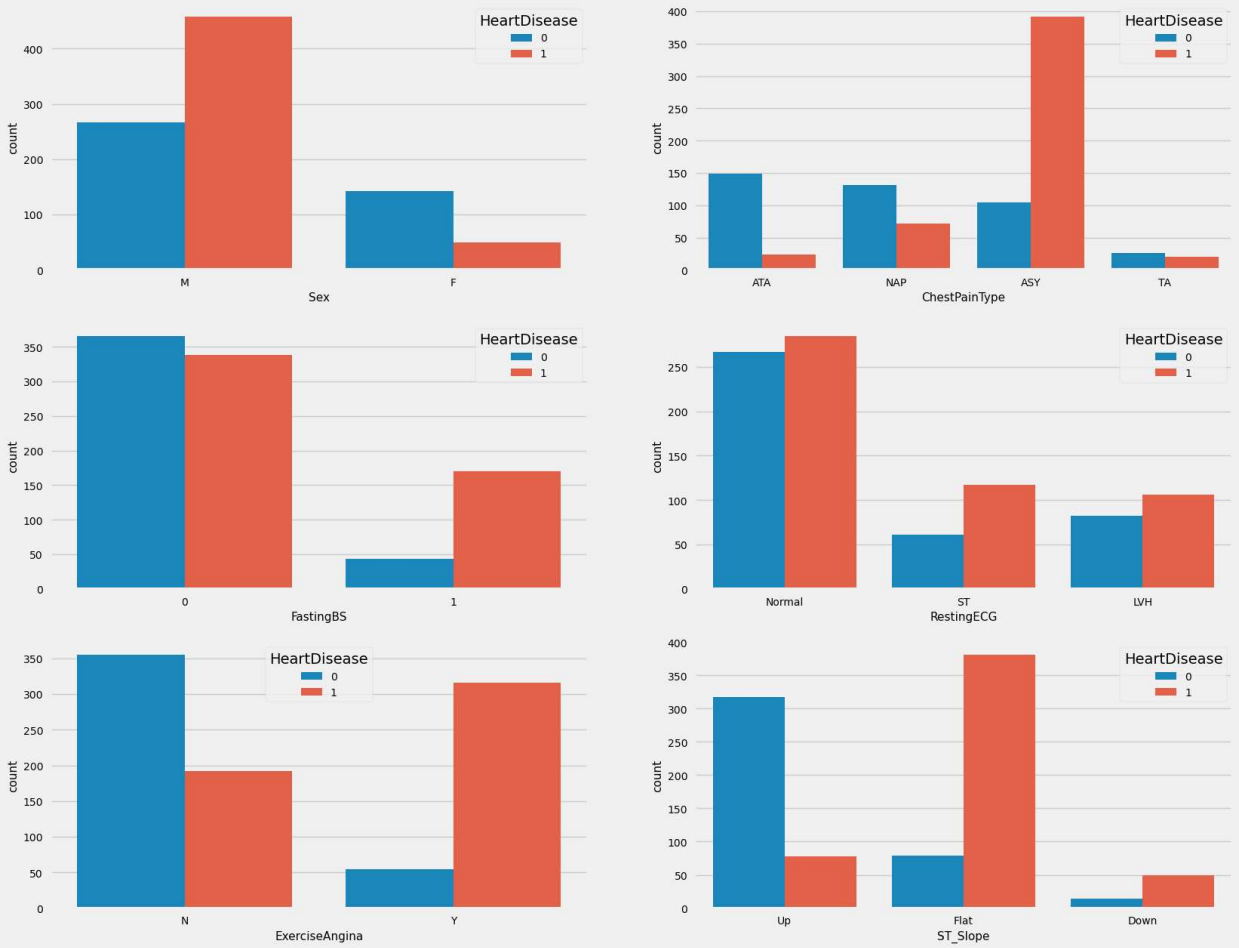
Seperating the data into discrette features and continuous feat

In [7]:
```
discrette_feature = [i for i in data.columns if data[i].nunique() < 10]
continuous_feature = [i for i in data.columns if data[i].nunique() > 10]
print(f'discrette feature: {discrette_feature}')
print(f'continuous feature: {continuous_feature}')
```

```
discrette feature: ['Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngin
a', 'ST_Slope', 'HeartDisease']
continuous feature: ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
```

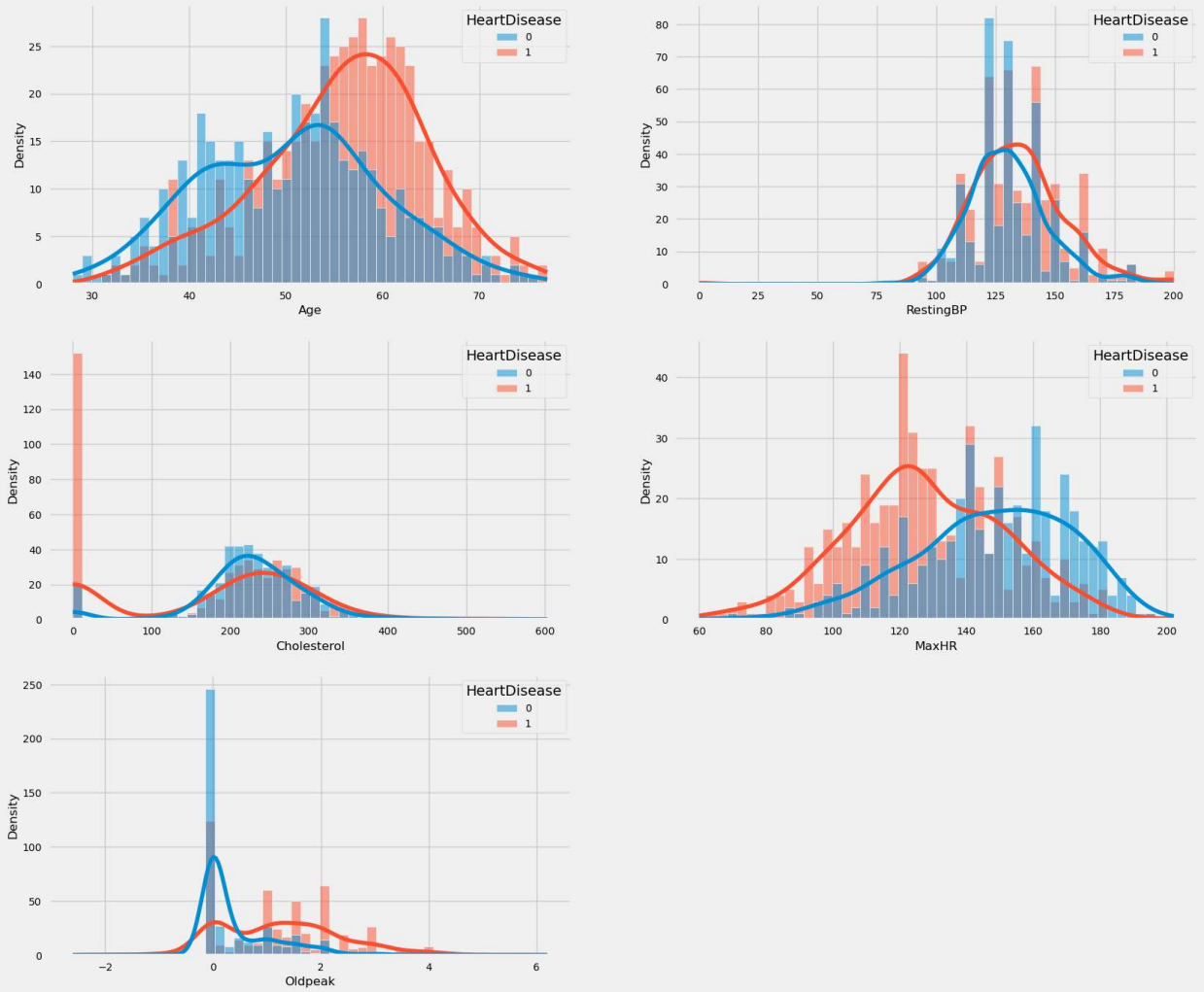
Visualising discrette feature with output variable count plot

In [8]:
```
plt.style.use(plt.style.available[19])
i = 1
plt.figure(figsize=(18,20))
for feature in discrette_feature[:-1]:
    plt.subplot(4, 2, i)
    sns.countplot(x=data[feature], hue=data.HeartDisease)
    i += 1

plt.show()
```



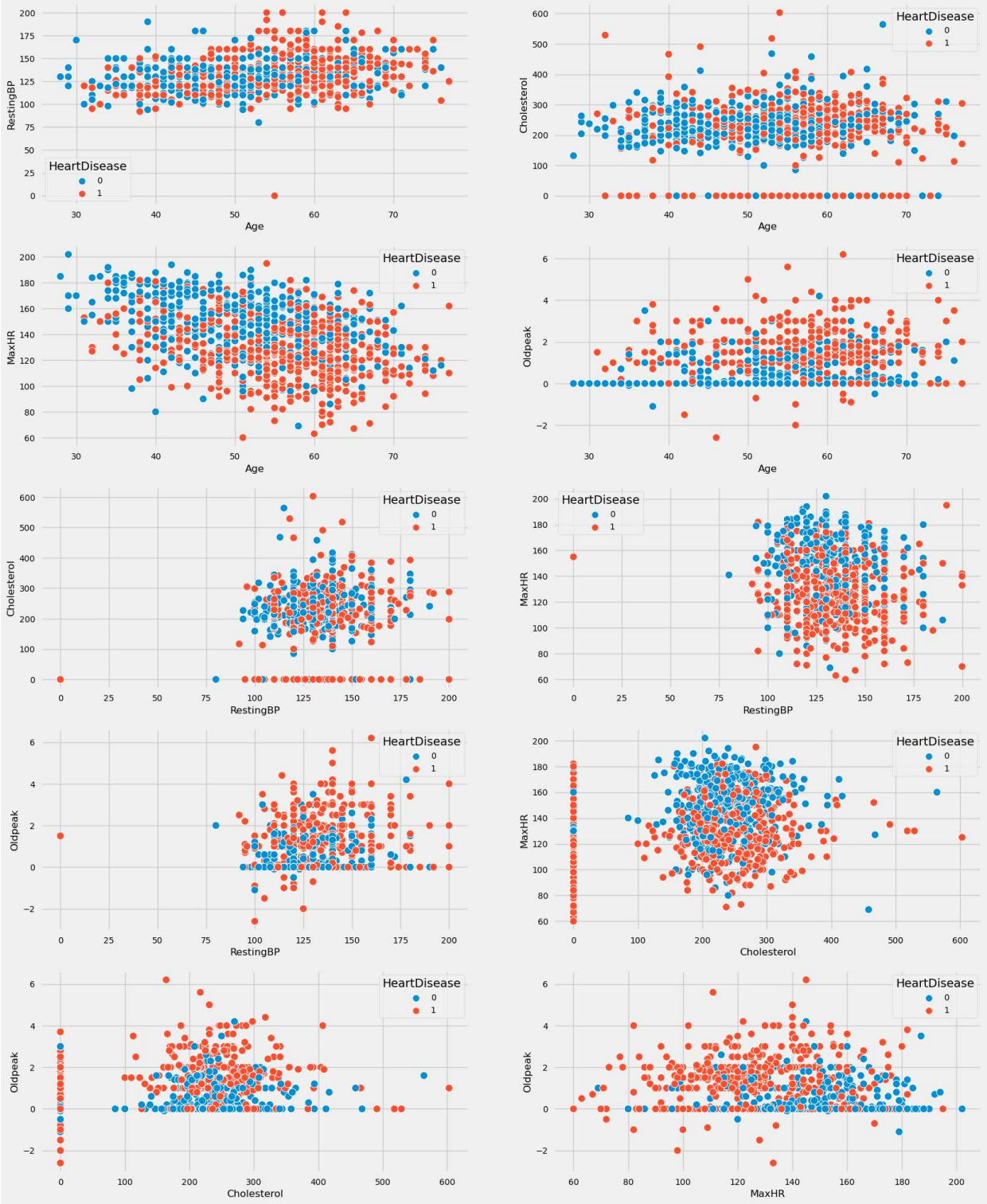
visualising continuous features with output variable using histogram and kde

In [9]:
```
plt.style.use('fivethirtyeight')
i = 1
plt.figure(figsize=(18,16))
for feature in continuous_feature:
    plt.subplot(3, 2, i)
    sns.histplot(x=data[feature],kde=True,bins=50, hue=data.HeartDisease)
    plt.xlabel(feature,size=12)
    plt.ylabel("Density",size=12)
    i += 1

plt.show()
```



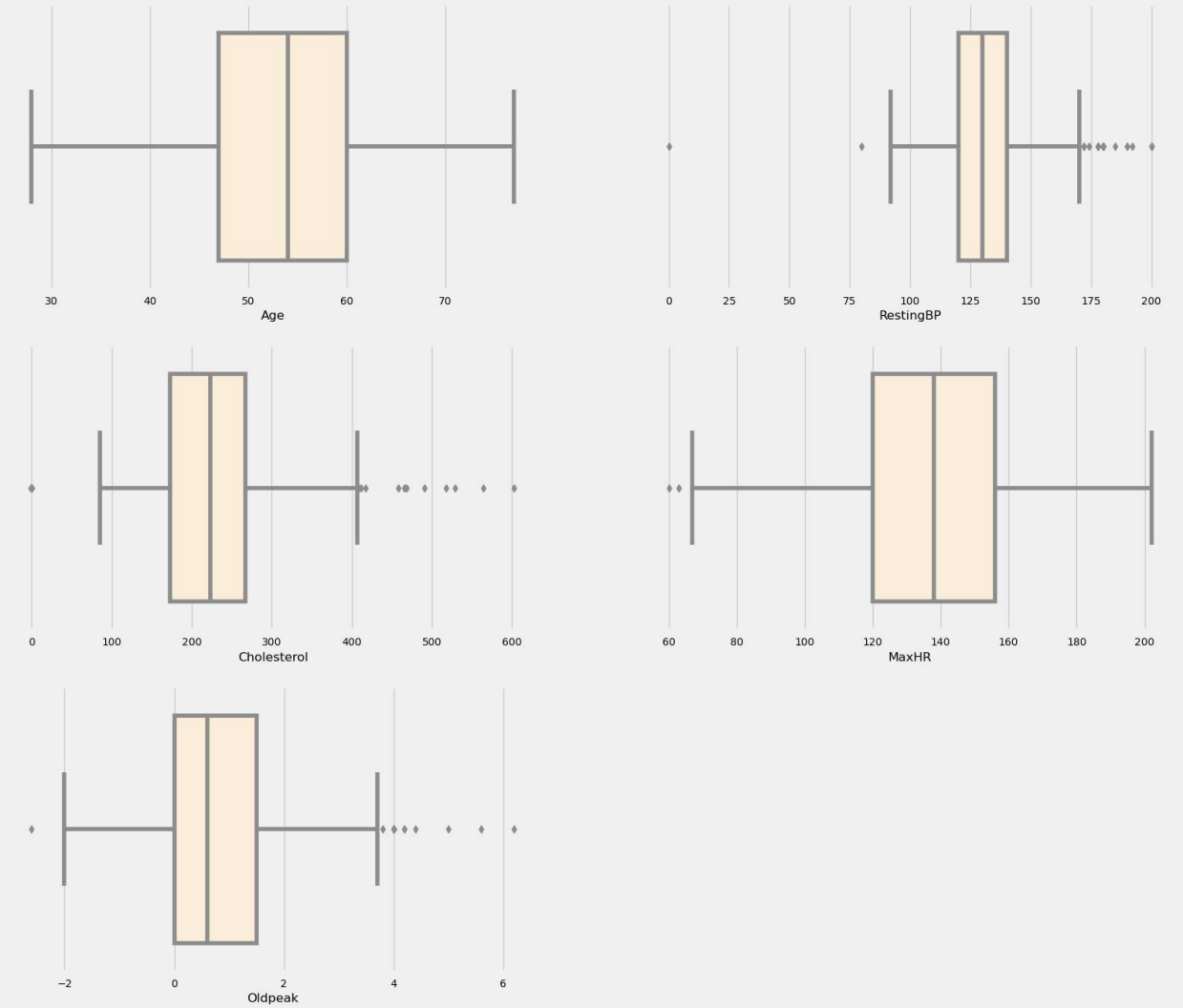
visualising continuous features with each other using scatter plot

In [10]:
```
i = 1
n = 1
plt.figure(figsize=(18,24))
for feature in continuous_feature:
    for i_ in continuous_feature[n:]:
        plt.subplot(5, 2, i)
        sns.scatterplot(x=data[feature],y=data[i_], s=80, hue=data.HeartDisease)
        plt.xlabel(feature,size=12)
        plt.ylabel(i_,size=12)
        i += 1
    n+=1
plt.show()
```

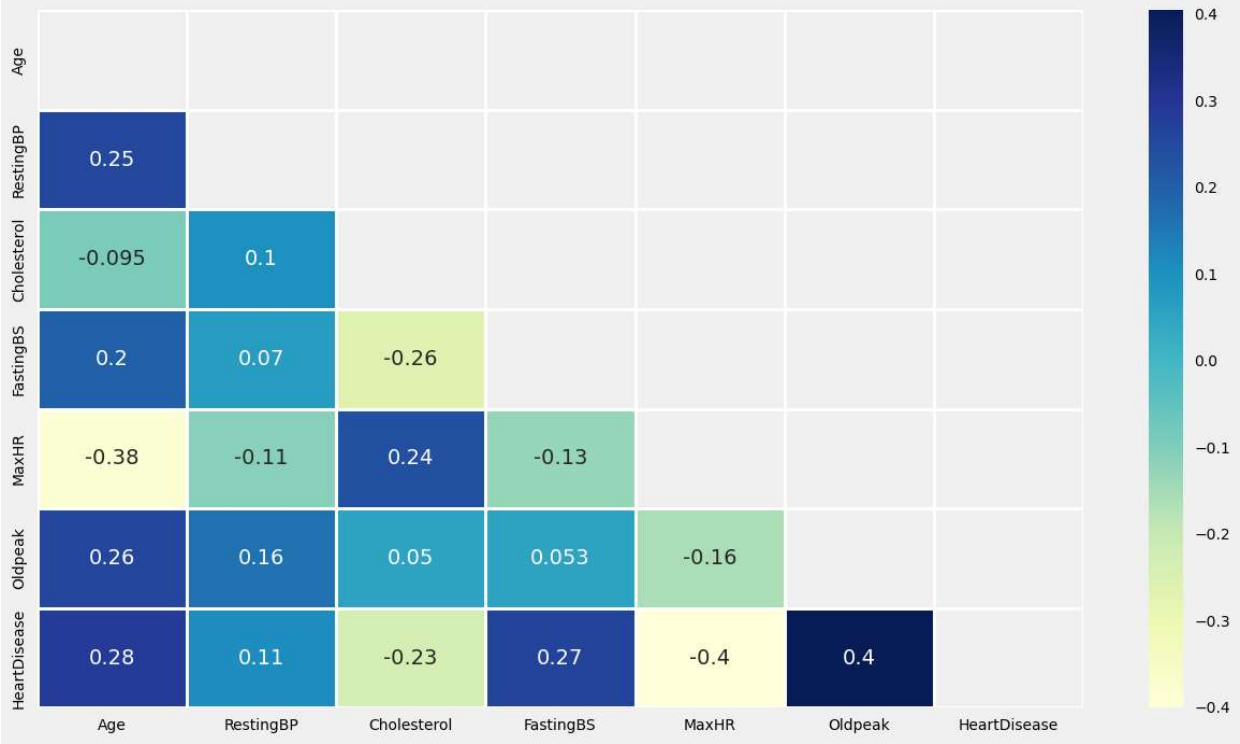


In [11]:

```
plt.style.use('fivethirtyeight')
i = 1
plt.figure(figsize=(18,16))
for feature in continuous_feature:
    plt.subplot(3, 2, i)
    sns.boxplot(data=data, x=feature, hue='HeartDisease', color='papayawhip')
    plt.xlabel(feature,size=12)
#     plt.ylabel("Density",size=12)
    i += 1

plt.show()
```



In [12]:
```
plt.figure(figsize=(14,8))
sns.heatmap(data.corr(), mask=np.triu(data.corr()), annot=True, cmap='YlGnBu',linewidt
plt.show()
```



Data Preprocessing

In [13]:
```
data.head()
```

Out[13]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina
0	40	M	ATA	140	289	0	Normal	172	N
1	49	F	NAP	160	180	0	Normal	156	N
2	37	M	ATA	130	283	0	ST	98	N
3	48	F	ASY	138	214	0	Normal	108	Y
4	54	M	NAP	150	195	0	Normal	122	N

seperating input features and output features

In [14]:
```
X = data.drop(columns=['HeartDisease'])
y = data['HeartDisease']
```

splitting the data for training and testing

In [15]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state
print(f'x_train shape: {x_train.shape}')
print(f'y_train shape: {y_train.shape}')
print(f'x_test shape : {x_test.shape}')
print(f'y_test shape : {y_test.shape}')
```

```
x_train shape: (734, 11)
y_train shape: (734,)
x_test shape : (184, 11)
y_test shape : (184,)
```

Applying One Hot Encoding and standardization using column transformer

In [16]:
```
transformer = ColumnTransformer(transformers = [('onehot', OneHotEncoder(sparse=False,
                                                 ('scaler', StandardScaler(), [0,3,4,5,
                                        remainder = 'passthrough')
x_train = transformer.fit_transform(x_train)
x_test = transformer.transform(x_test)
```

Model Building

In [17]:
```
def do_prediction(classifier):

    # training the classifier on the dataset
    classifier.fit(x_train, y_train)

    #Do prediction and evaluting the prediction
    prediction = classifier.predict(x_test)
    cross_validation_score = cross_val(x_train,y_train, classifier)
    accuracy = accuracy_score(y_test, prediction)
    con_metrix = confusion_matrix(y_test, prediction)
    tn,fp,fn,tp= confusion_matrix(y_test,prediction).ravel()
    print("TN:",tn)
    print("FP:",fp)
    print("FN:",fn)
    print("TP:",tp)
    PR=precision_score(y_test, prediction)
    RC=recall_score(y_test,prediction)
    F1=2*PR*RC/(PR+RC)

    return prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1

def cross_val(x_train, y_train, classifier):

    # Applying k-Fold Cross Validation
    accuracies = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv
    return accuracies.mean()
```

Logistic Regression

In [18]:
```
logistic_reg = LogisticRegression()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(logist

plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for logistic regression',size=14)
plt.show()
print('Logistic Regression Performace on the training data have an accuracy score is -
print('Logistic Regression Performace on the testing data have an accuracy score is -
print('Precission of logistic regression is - {}'.format((PR).round(2)))
print('Recall of logistic regression is - {}'.format((RC).round(2)))
print('F1 score  of logistic regression is - {}'.format((F1).round(2)))
```

```
TN: 73
FP: 14
FN: 10
TP: 87
```

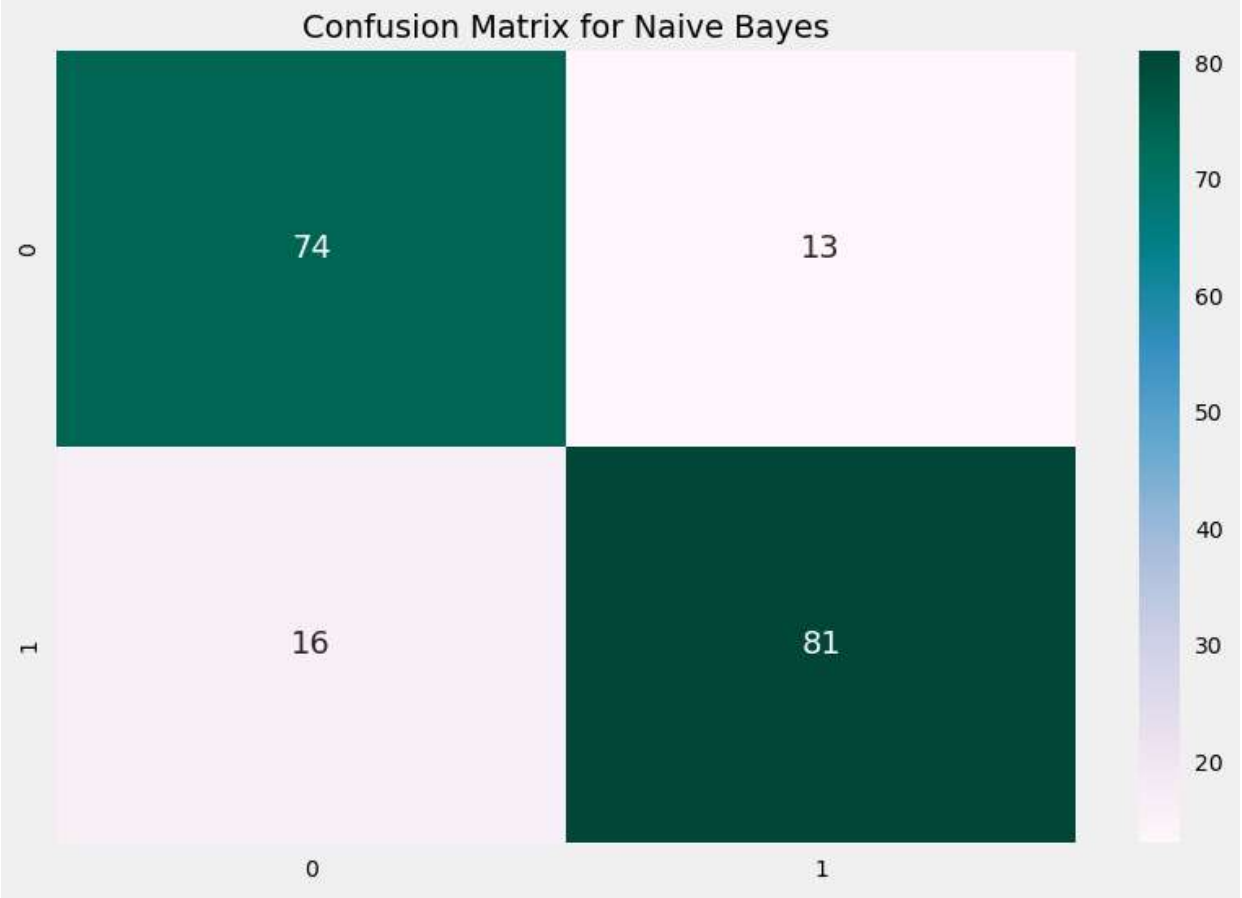

```
Logistic Regression Performace on the training data have an accuracy score is - 86.0
Logistic Regression Performace on the testing data have an accuracy score is - 87.0
Precission of logistic regression is - 0.86
Recall of logistic regression is - 0.9
F1 score  of logistic regression is - 0.88
```

In [ ]:

Naive Bayes

In [85]:
```
gaussian = GaussianNB()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(gaussi

plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for Naive Bayes',size=14)
plt.show()
print('Naive Bayes Performace on the training data have an accuracy score of {}'.forma
print('Naive Bayes Performace on the testing data have an accuracy score of {}'.format
print('Precission of Naive Bayes is - {}'.format((PR).round(2)))
print('Recall of Naive Bayes is - {}'.format((RC).round(2)))
print('F1 score  of Naive Bayes is - {}'.format((F1).round(2)))
```

```
TN: 74
FP: 13
FN: 16
TP: 81
```




```
Naive Bayes Performace on the training data have an accuracy score of 86.0
Naive Bayes Performace on the testing data have an accuracy score of 84.0
Precission of Naive Bayes is - 0.86
Recall of Naive Bayes is - 0.84
F1 score  of Naive Bayes is - 0.85
```

In [ ]:

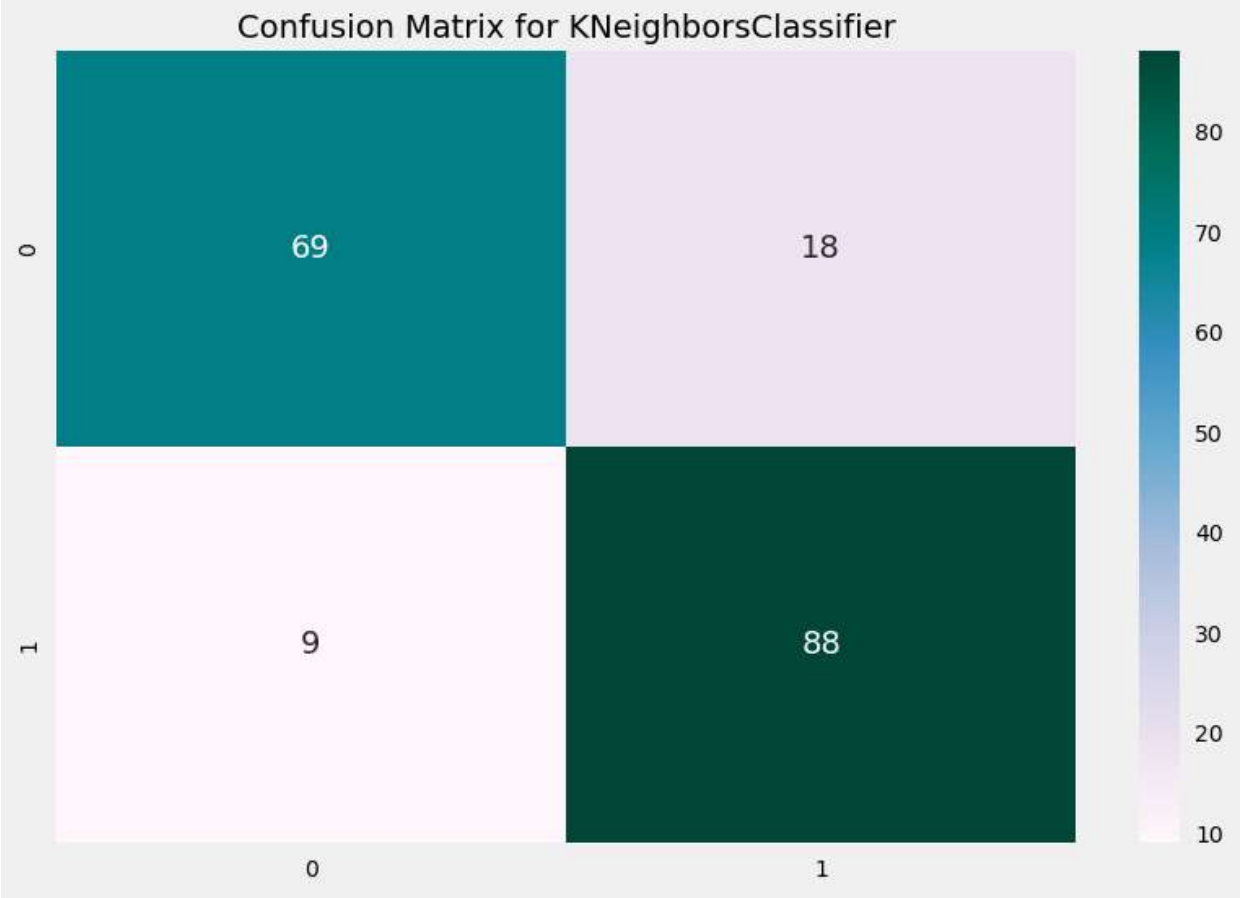
In [ ]:

K-Nearest Neighbour

In [87]:
```
KNN =  KNeighborsClassifier()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(KNN)
plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for KNeighborsClassifier',size=14)
plt.show()
print(' KNeighborsClassifier Performace on the training data have an accuracy score of
print(' KNeighborsClassifierPerformace on the testing data have an accuracy score of {
print('Precission of KNeighborsClassifier is - {}'.format((PR).round(2)))
print('Recall of KNeighborsClassifier is - {}'.format((RC).round(2)))
print('F1 score  of KNeighborsClassifier is - {}'.format((F1).round(2)))
```

```
TN: 69
FP: 18
FN: 9
TP: 88
```

```
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: F
utureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default
behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this
behavior will change: the default value of `keepdims` will become False, the `axis` o
ver which the statistic is taken will be eliminated, and the value None will no longe
r be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: F
utureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default
behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this
behavior will change: the default value of `keepdims` will become False, the `axis` o
ver which the statistic is taken will be eliminated, and the value None will no longe
r be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: F
utureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default
behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this
behavior will change: the default value of `keepdims` will become False, the `axis` o
ver which the statistic is taken will be eliminated, and the value None will no longe
r be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: F
utureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default
behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this
behavior will change: the default value of `keepdims` will become False, the `axis` o
ver which the statistic is taken will be eliminated, and the value None will no longe
r be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: F
utureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default
behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this
behavior will change: the default value of `keepdims` will become False, the `axis` o
ver which the statistic is taken will be eliminated, and the value None will no longe
r be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: F
utureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default
behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this
behavior will change: the default value of `keepdims` will become False, the `axis` o
ver which the statistic is taken will be eliminated, and the value None will no longe
r be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```




```
 KNeighborsClassifier Performace on the training data have an accuracy score of 84.0
 KNeighborsClassifierPerformace on the testing data have an accuracy score of 85.0
Precission of KNeighborsClassifier is - 0.83
Recall of KNeighborsClassifier is - 0.91
F1 score  of KNeighborsClassifier is - 0.87
```

In [ ]:

Support Vector Machine

In [89]:
```
svm = SVC()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(svm)

plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for SVM',size=14)
plt.show()
print('Support Vector Machine Performace on the training data have an accuracy score o
print('Support Vector Machine Performace on the testing data have an accuracy score of
print('Precission of Support Vector Machine is - {}'.format((PR).round(2)))
print('Recall of Support Vector Machine is - {}'.format((RC).round(2)))
print('F1 score  of Support Vector Machine is - {}'.format((F1).round(2)))
```

```
TN: 74
FP: 13
FN: 10
TP: 87
```

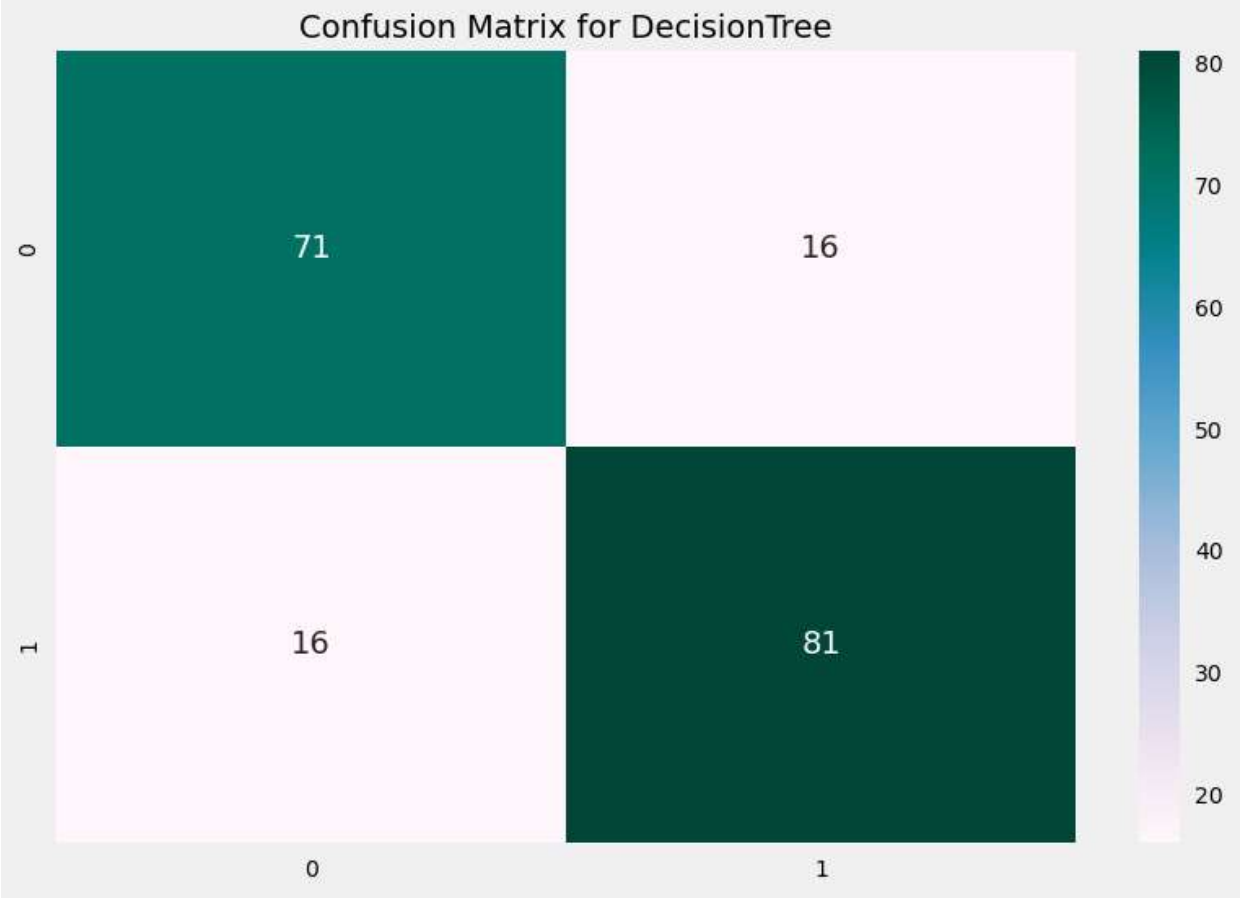

```
Support Vector Machine Performace on the training data have an accuracy score of 86.0
Support Vector Machine Performace on the testing data have an accuracy score of 88.0
Precission of Support Vector Machine is - 0.87
Recall of Support Vector Machine is - 0.9
F1 score  of Support Vector Machine is - 0.88
```

In [ ]:

DecisionTreeClassifier

In [92]:
```
DTCclassifier = DecisionTreeClassifier()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(DTCcla
plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for DecisionTree',size=14)
plt.show()
print('DecisionTree Performace on the training data have an accuracy score of {}'.form
print('DecisionTree Performace on the testing data have an accuracy score of {}'.forma
print('Precission of DecisionTree is - {}'.format((PR).round(2)))
print('Recall of DecisionTree is - {}'.format((RC).round(2)))
print('F1 score  of DecisionTree is - {}'.format((F1).round(2)))
```

```
TN: 71
FP: 16
FN: 16
TP: 81
```



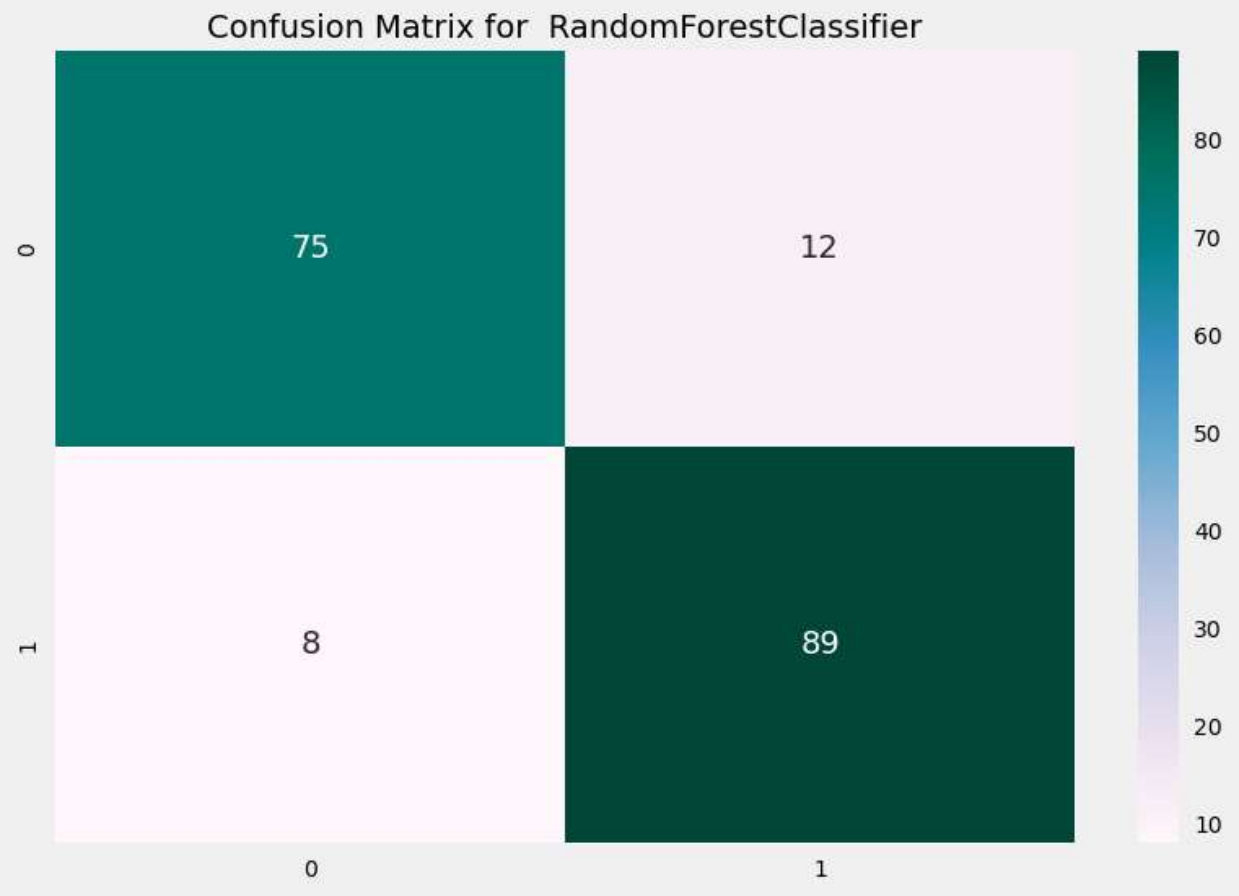

```
DecisionTree Performace on the training data have an accuracy score of 75.0
DecisionTree Performace on the testing data have an accuracy score of 83.0
Precission of DecisionTree is - 0.84
Recall of DecisionTree is - 0.84
F1 score  of DecisionTree is - 0.84
```

In [ ]:

Random Forest

In [93]:
```
Rndforestclassifier =  RandomForestClassifier()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(Rndfor
plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for  RandomForestClassifier',size=14)
plt.show()
print(' RandomForestClassifier Performace on the training data have an accuracy score
print(' RandomForestClassifierPerformace on the testing data have an accuracy score of
print('Precission of RandomForestClassifier is - {}'.format((PR).round(2)))
print('Recall of RandomForestClassifier is - {}'.format((RC).round(2)))
print('F1 score  of RandomForestClassifier is - {}'.format((F1).round(2)))
```

```
TN: 75
FP: 12
FN: 8
TP: 89
```




```
 RandomForestClassifier Performace on the training data have an accuracy score of 86.
0
 RandomForestClassifierPerformace on the testing data have an accuracy score of 89.0
Precission of RandomForestClassifier is - 0.88
Recall of RandomForestClassifier is - 0.92
F1 score  of RandomForestClassifier is - 0.9
```

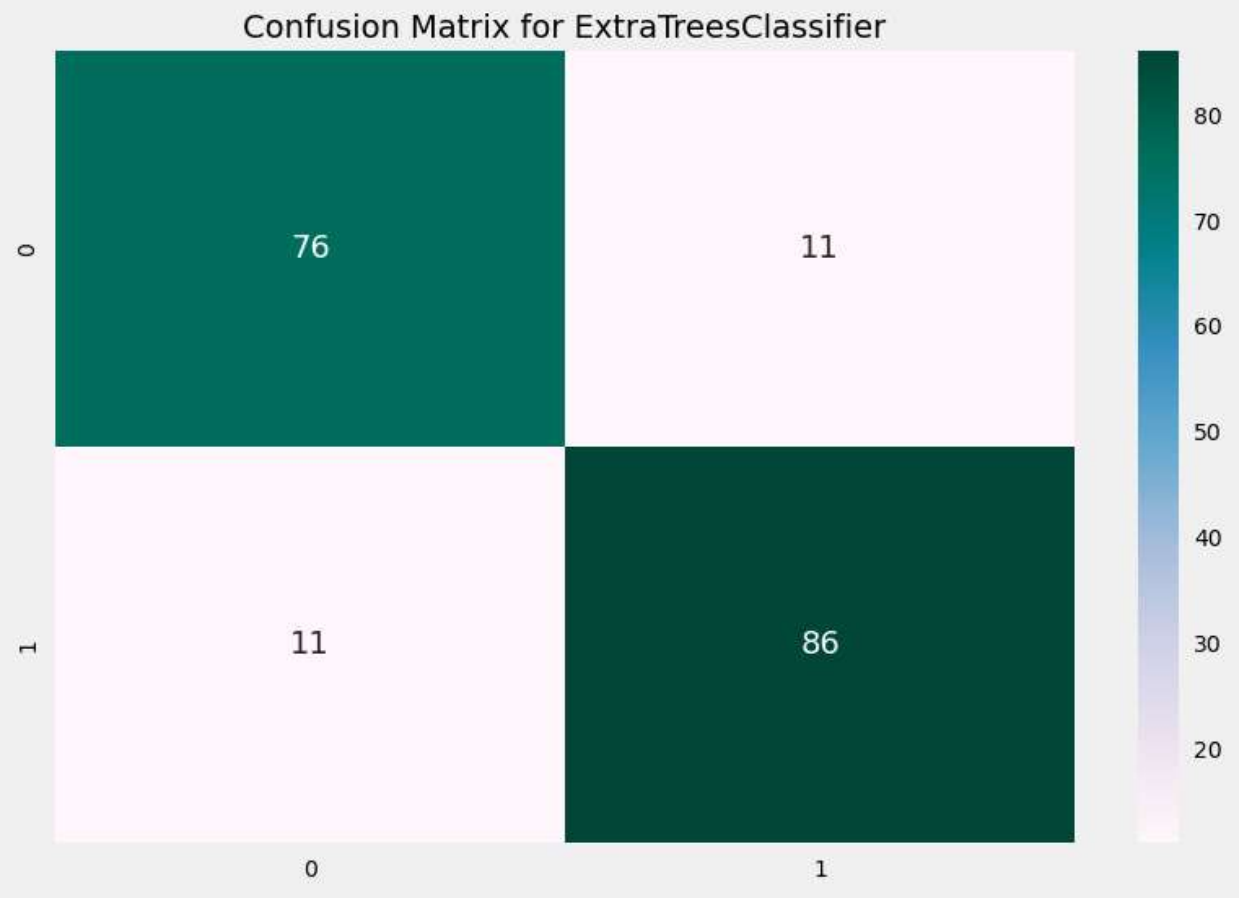
In [ ]:

ExtraTreesClassifier

In [96]:

```
ExtraTrees=  ExtraTreesClassifier()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(Extra
plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for ExtraTreesClassifier',size=14)
plt.show()
print(' ExtraTreesClassifier Performace on the training data have an accuracy score of
print(' ExtraTreesClassifier Performace on the testing data have an accuracy score of
print('Precission of ExtraTreesClassifier is - {}'.format((PR).round(2)))
print('Recall of ExtraTreesClassifier is - {}'.format((RC).round(2)))
print('F1 score  of ExtraTreesClassifier is - {}'.format((F1).round(2)))
```

```
TN: 76
FP: 11
FN: 11
TP: 86
```



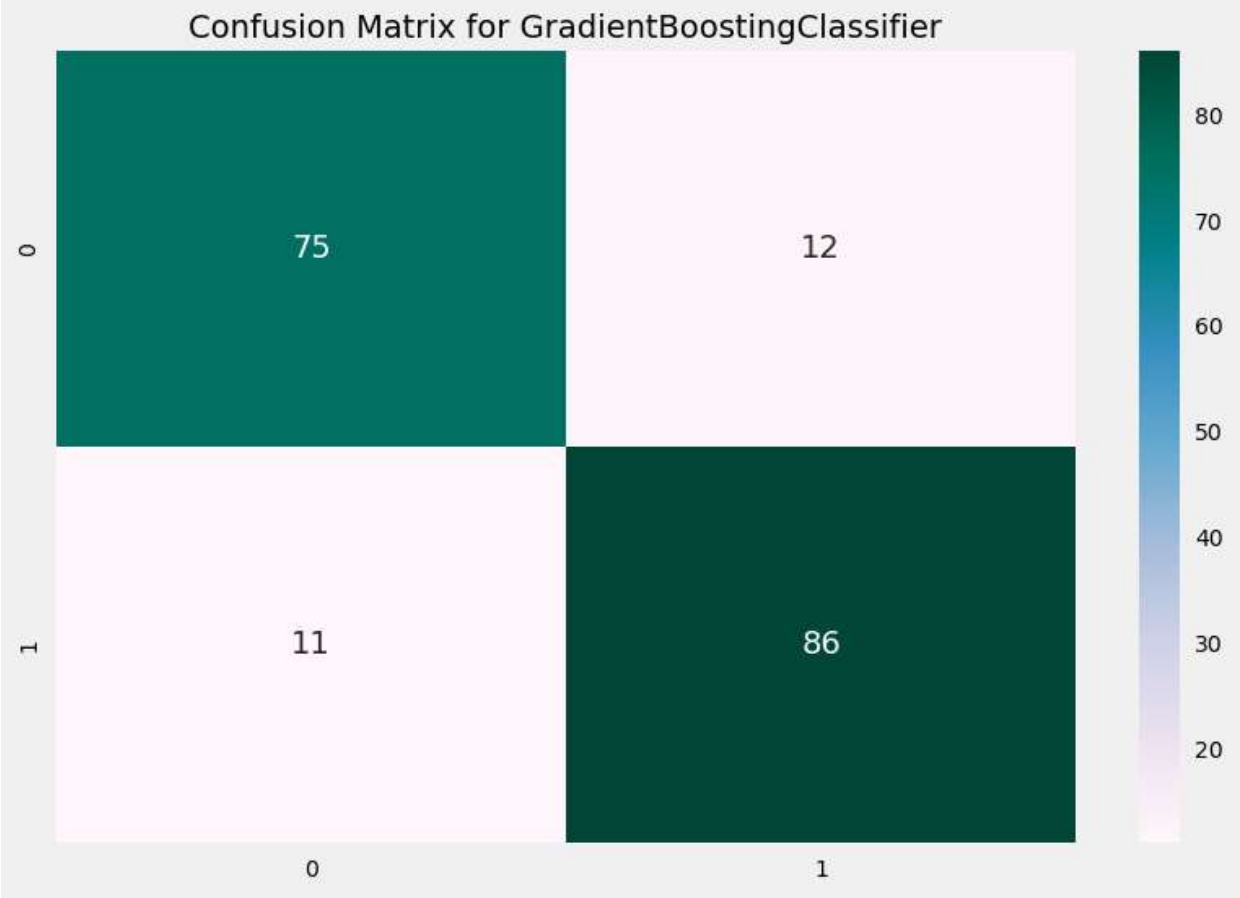

```
 ExtraTreesClassifier Performace on the training data have an accuracy score of 86.0
 ExtraTreesClassifier Performace on the testing data have an accuracy score of 88.0
Precission of ExtraTreesClassifier is - 0.89
Recall of ExtraTreesClassifier is - 0.89
F1 score  of ExtraTreesClassifier is - 0.89
```

GradientBoostingClassifier

In [98]:

```
GradientBoost =  GradientBoostingClassifier()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(Gradie
plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for GradientBoostingClassifier',size=14)
plt.show()
print(' GradientBoostingClassifier Performace on the training data have an accuracy sc
print(' GradientBoostingClassifierPerformace on the testing data have an accuracy scor
print('Precission of GradientBoostingClassifier is - {}'.format((PR).round(2)))
print('Recall of GradientBoostingClassifier is - {}'.format((RC).round(2)))
print('F1 score  of GradientBoostingClassifier is - {}'.format((F1).round(2)))
```

```
TN: 75
FP: 12
FN: 11
TP: 86
```



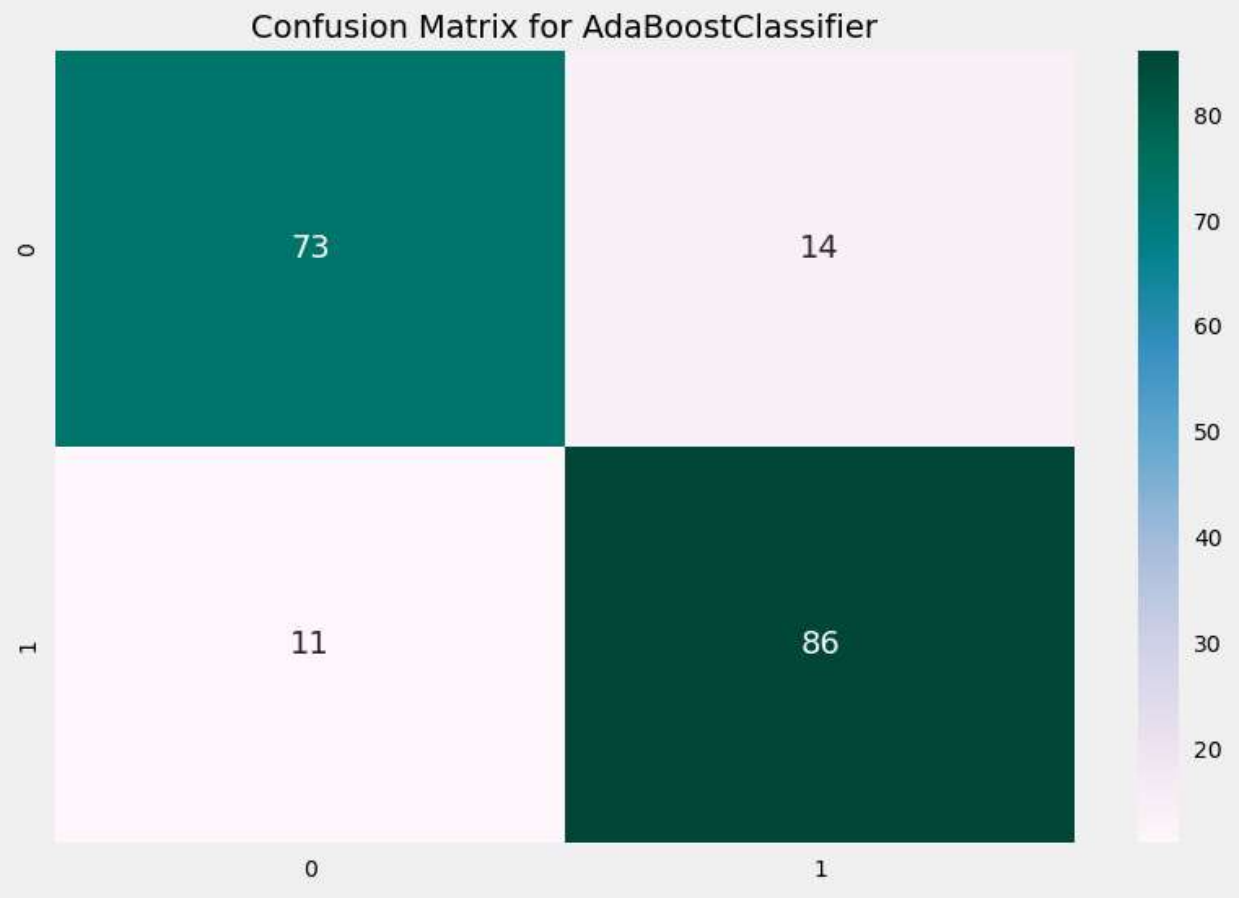

```
 GradientBoostingClassifier Performace on the training data have an accuracy score of
86.0
 GradientBoostingClassifierPerformace on the testing data have an accuracy score of 8
8.0
Precission of GradientBoostingClassifier is - 0.88
Recall of GradientBoostingClassifier is - 0.89
F1 score  of GradientBoostingClassifier is - 0.88
```

AdaBoostClassifier

In [99]:

```
AdaBoost=  AdaBoostClassifier()
prediction,accuracy,con_metrix, cross_validation_score,PR,RC,F1 = do_prediction(AdaBoo
plt.figure(figsize=(9,6))
sns.heatmap(con_metrix, annot=True, cmap='PuBuGn')
plt.title('Confusion Matrix for AdaBoostClassifier',size=14)
plt.show()
print(' AdaBoostClassifier Performace on the training data have an accuracy score of {
print(' AdaBoostClassifier Performace on the testing data have an accuracy score of {}
print('Precission of AdaBoostClassifier is - {}'.format((PR).round(2)))
print('Recall of AdaBoostClassifier is - {}'.format((RC).round(2)))
print('F1 score  of AdaBoostClassifier is - {}'.format((F1).round(2)))
```

```
TN: 73
FP: 14
FN: 11
TP: 86
```




```
 AdaBoostClassifier Performace on the training data have an accuracy score of 83.0
 AdaBoostClassifier Performace on the testing data have an accuracy score of 86.0
Precission of AdaBoostClassifier is - 0.86
Recall of AdaBoostClassifier is - 0.89
F1 score  of AdaBoostClassifier is - 0.87
```

In [ ]: