In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import KFold
```

In [2]:
```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

In [3]:
```
head = dataset.head() # First 5 line
print(head,"\n")
tail = dataset.tail() # Last 5 line
print(tail,"\n")
describe = dataset.describe() #summary statistics for numerical columns
print(describe,"\n")
info = dataset.info() #index, datatype and memory information
print(info,"\n")
min = dataset.min() # Returns the lowest value in each column
print(min,"\n")
median = dataset.median() # Returns the median value in each column
print(median,"\n")
max = dataset.max() # Returns the highest value in each column
print(max,"\n")
```

```
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0

      User ID  Gender  Age  EstimatedSalary  Purchased
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
```

```
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

            User ID         Age  EstimatedSalary   Purchased
count  4.000000e+02  400.000000       400.000000  400.000000
mean   1.569154e+07   37.655000     69742.500000    0.357500
std    7.165832e+04   10.482877     34096.960282    0.479864
min    1.556669e+07   18.000000     15000.000000    0.000000
25%    1.562676e+07   29.750000     43000.000000    0.000000
50%    1.569434e+07   37.000000     70000.000000    0.000000
75%    1.575036e+07   46.000000     88000.000000    1.000000
max    1.581524e+07   60.000000    150000.000000    1.000000

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
User ID            400 non-null int64
Gender             400 non-null object
Age                400 non-null int64
EstimatedSalary    400 non-null int64
Purchased          400 non-null int64
dtypes: int64(4), object(1)
memory usage: 15.7+ KB
None

User ID            15566689
Gender               Female
Age                      18
EstimatedSalary       15000
Purchased                 0
dtype: object

User ID            15694341.5
Age                      37.0
EstimatedSalary       70000.0
Purchased                0.0
dtype: float64
```

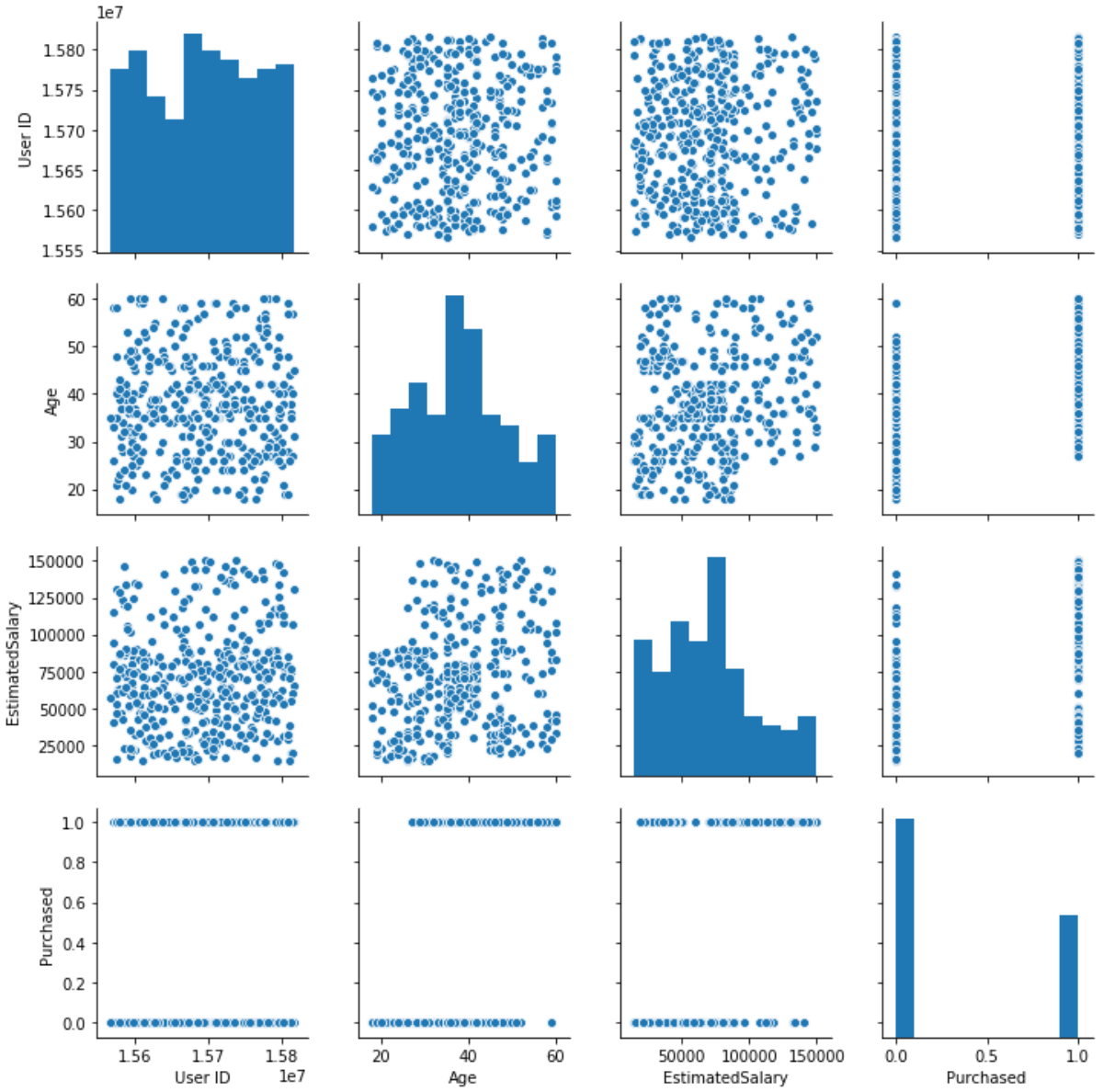
```
User ID            15815236
Gender                 Male
Age                      60
EstimatedSalary      150000
Purchased                 1
dtype: object
```

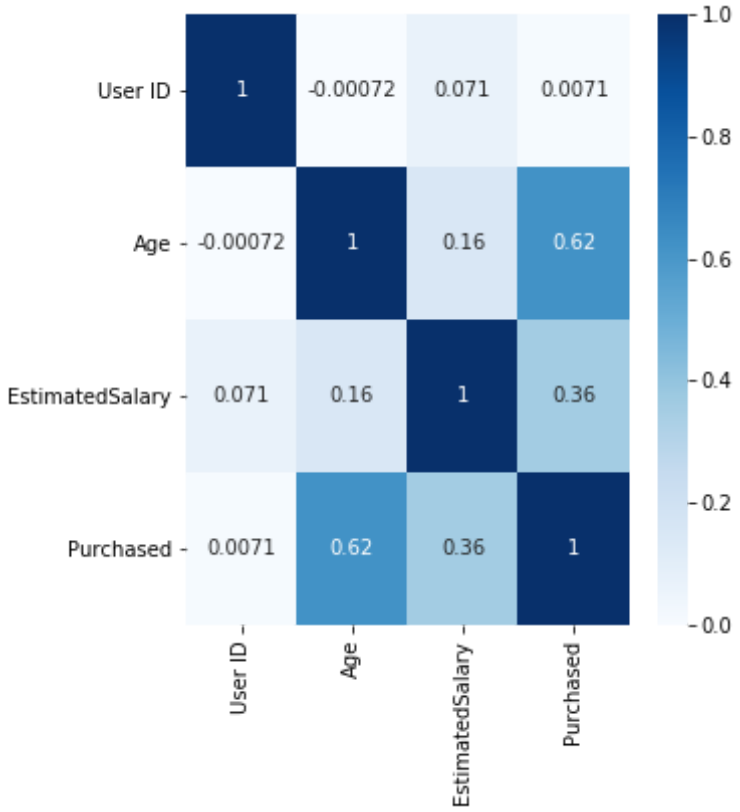
In [4]:
```
#seaborn vis.

sns.pairplot(dataset)

sns.catplot(x="Age", y="EstimatedSalary", data=dataset);
sns.heatmap(dataset.corr(),cmap = 'Blues', annot=True)
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1efec7d0e48>
```





In [5]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.33, random_state = 0)
```

In [6]:
```
# feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\utils\validation.py:59
5: DataConversionWarning: Data with input dtype int64 was converted to
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

```
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\utils\validation.py:59
5: DataConversionWarning: Data with input dtype int64 was converted to
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\utils\validation.py:59
5: DataConversionWarning: Data with input dtype int64 was converted to
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [7]:
```
#Logistic Regression

from sklearn.linear_model import LogisticRegression
logr = LogisticRegression(random_state=0)
logr.fit(X_train,y_train)

y_pred = logr.predict(X_test)
```

```
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.p
y:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)
```

In [8]:
```
from sklearn import metrics
#mean absolute error
mae = metrics.mean_absolute_error(y_test,y_pred)
print("mean absolute error",mae)
#mean squared error
mse = metrics.mean_squared_error(y_test, y_pred)
print("mean squared error: ",mse)
#root mean square error
rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print("root mean square error: ",rmse)


print("coefficient: ",logr.coef_)
```

```
mean absolute error 0.12878787878787878
mean squared error:  0.12878787878787878

root mean square error:  0.3588702812826367
```

```
Root Mean Square Error:  0.3588702812826307
coefficient:  [[2.02677084 0.98516956]]
```

In [9]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[78  6]
 [11 37]]
```

In [10]:
```
# calculate accuracy
from sklearn import metrics
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.8712121212121212
```

In [11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.33, random_state = 0)

# feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\utils\validation.py:59
5: DataConversionWarning: Data with input dtype int64 was converted to
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\utils\validation.py:59
5: DataConversionWarning: Data with input dtype int64 was converted to
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\utils\validation.py:59
5: DataConversionWarning: Data with input dtype int64 was converted to
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [12]:
```
# fitting K-NN to the training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski'
, p = 2)
classifier.fit(X_train, y_train)
```

Out[12]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [13]:
```
# 10-Fold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y
_train, cv = 10)
accuracies.mean()
accuracies.std()
```

Out[13]:
```
0.039898768298312896
```

In [14]:
```
# Predict the test result
y_pred = classifier.predict(X_test)
```

In [15]:
```
# confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

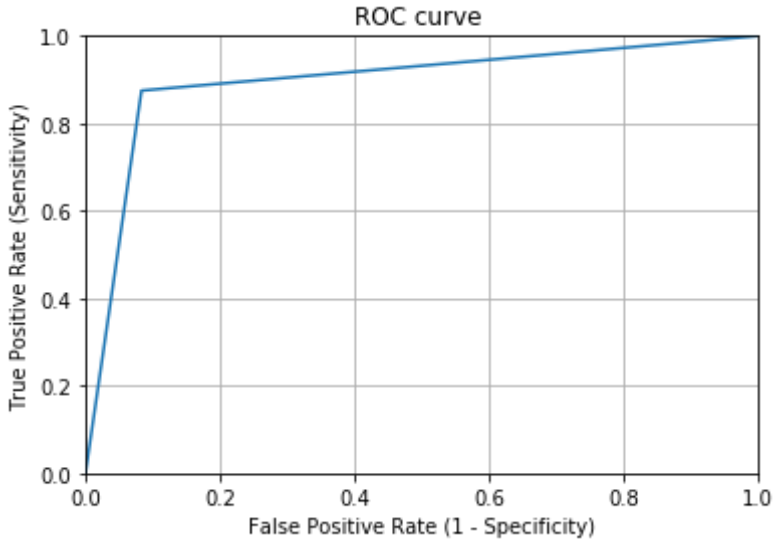
```
[[77  7]
 [ 6 42]]
```

In [16]:
```
from sklearn import metrics

roc = metrics.roc_auc_score(y_test,y_pred)
print("roc: ",roc)
```

```
roc:  0.8958333333333334
```

In [17]:
```
#ROC curve
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred)
plt.plot(fpr, tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve ')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
```



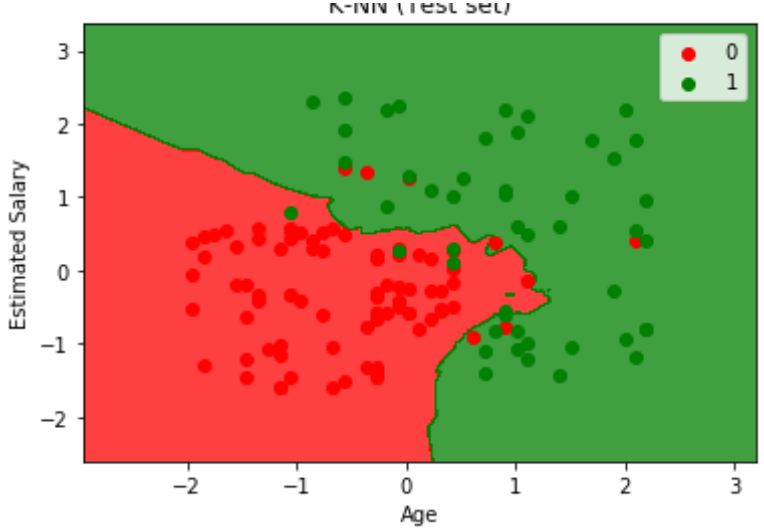
In [18]:
```
#calculate acc. score
from sklearn.metrics import accuracy_score
print("Accuracy score: ", accuracy_score(y_test,y_pred))
```

```
Accuracy score:  0.9015151515151515
```

In [19]:
```
# TEST SETS //KNN
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_
set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_
```

```
set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel
()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which sh
ould be avoided as value-mapping will have precedence in case its lengt
h matches with 'x' & 'y'.  Please use a 2-D array with a single row if
you really want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which sh
ould be avoided as value-mapping will have precedence in case its lengt
h matches with 'x' & 'y'.  Please use a 2-D array with a single row if
you really want to specify the same RGB or RGBA value for all points.
```



In [20]:
```
# fitting random forest classification to the training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'ent
ropy', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[20]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='en
tropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                       oob_score=False, random_state=0, verbose=0, warm_start=Fals
e)
```

In [21]:
```
# 10-Fold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y
_train, cv = 10)
accuracies.mean()
accuracies.std()
```

Out[21]: 0.05902707833760258

In [22]:
```
y_pred = classifier.predict(X_test)
```

In [23]:
```
# confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

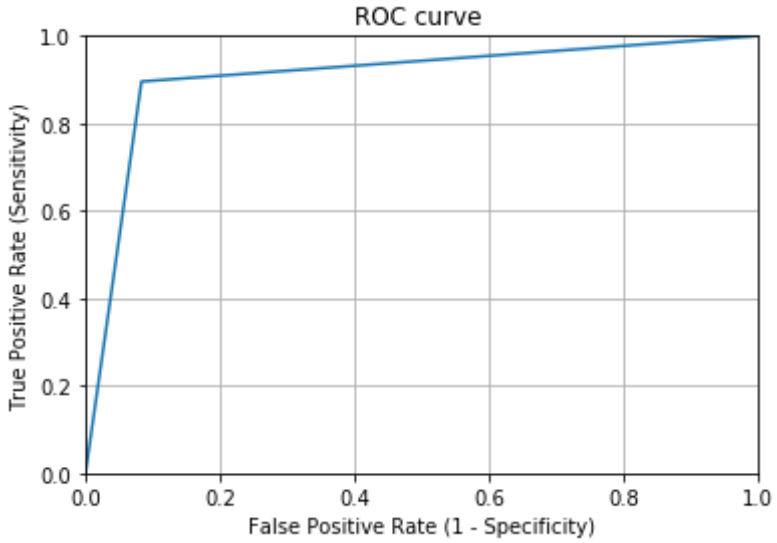
```
[[77  7]
 [ 5 43]]
```

In [24]:
```
from sklearn import metrics

roc = metrics.roc_auc_score(y_test,y_pred)
print("roc: ",roc)
```

```
roc:  0.90625
```

In [25]:
```
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred)
plt.plot(fpr, tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve ')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
```



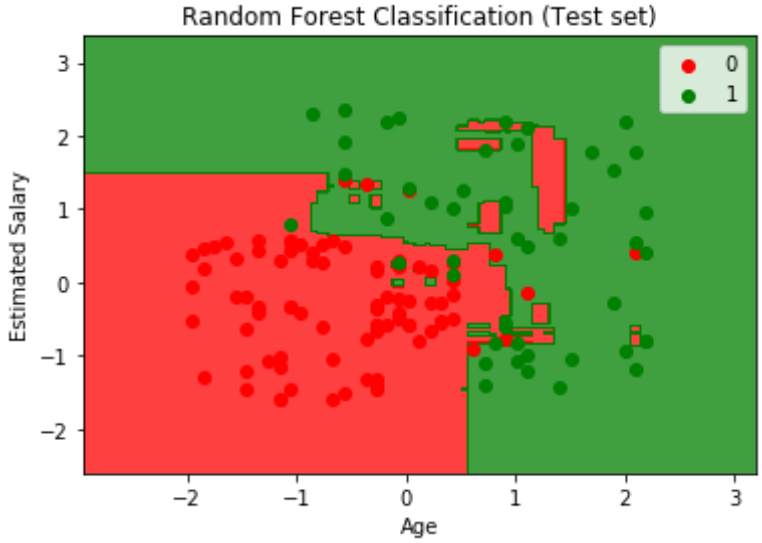
In [26]:
```
#calculate acc. score
from sklearn.metrics import accuracy_score
print("Accuracy score: ", accuracy_score(y_test,y_pred))
```

```
Accuracy score:  0.9090909090909091
```

In [27]:
```
# TEST SETS //Random Forest
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_
set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_
set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel
()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest Classification (Test set)')
```

```
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which sh
ould be avoided as value-mapping will have precedence in case its lengt
h matches with 'x' & 'y'.  Please use a 2-D array with a single row if
you really want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which sh
ould be avoided as value-mapping will have precedence in case its lengt
h matches with 'x' & 'y'.  Please use a 2-D array with a single row if
you really want to specify the same RGB or RGBA value for all points.
```



In [28]:
```
# fitting decision tree classification to the training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state
 = 0)
classifier.fit(X_train, y_train)
```

Out[28]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_dept
h=None,
                       max_features=None, max_leaf_nodes=None,
```

```
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False, random_state=
0,
                       splitter='best')
```

In [29]:
```
# 10-Fold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y
_train, cv = 10)
accuracies.mean()
accuracies.std()
```

Out[29]:
```
0.06268189553345331
```

In [30]:
```
y_pred = classifier.predict(X_test)
```

In [31]:
```
# confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[74 10]
 [ 6 42]]
```

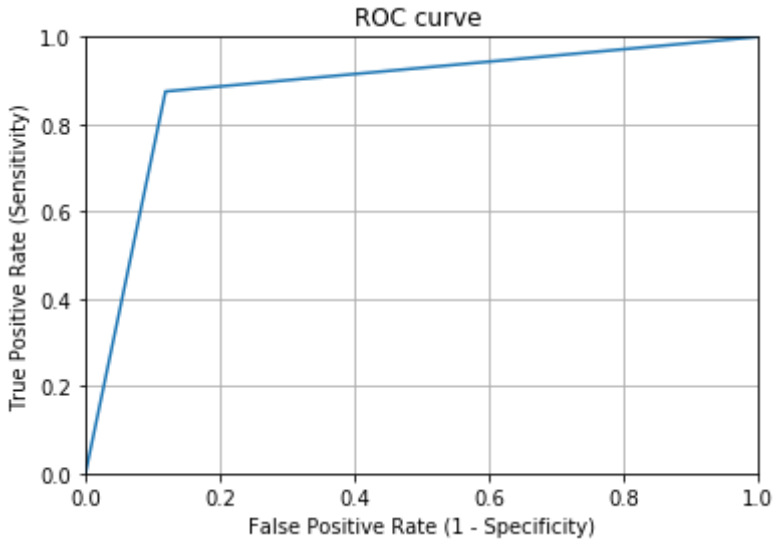
In [32]:
```
from sklearn import metrics

roc = metrics.roc_auc_score(y_test,y_pred)
print("roc: ",roc)
```

```
roc:  0.8779761904761905
```

In [33]:
```
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred)
plt.plot(fpr, tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve ')
```

```
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
```



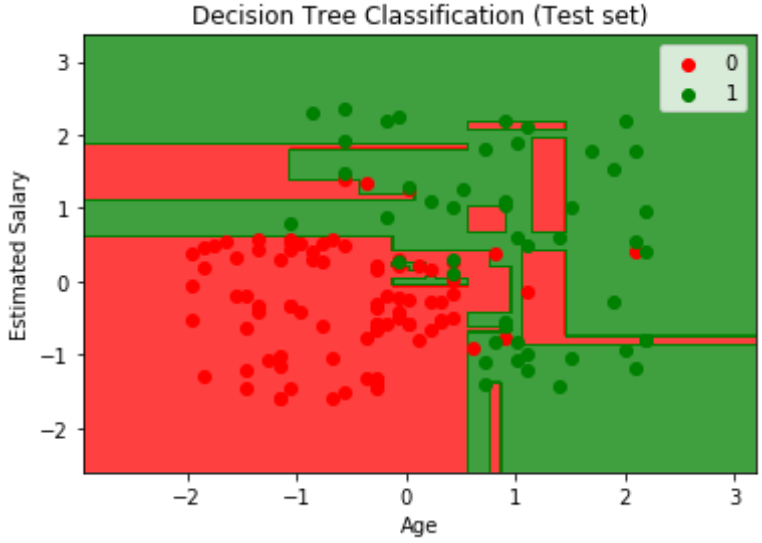
In [34]:
```
#calculate acc. score
from sklearn.metrics import accuracy_score
print("Accuracy score: ", accuracy_score(y_test,y_pred))
```

```
Accuracy score:  0.8787878787878788
```

In [35]:
```
# TEST SETS //Decision Tree
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_
set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_
set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel
()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
```

```
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which sh
ould be avoided as value-mapping will have precedence in case its lengt
h matches with 'x' & 'y'.  Please use a 2-D array with a single row if
you really want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which sh
ould be avoided as value-mapping will have precedence in case its lengt
h matches with 'x' & 'y'.  Please use a 2-D array with a single row if
you really want to specify the same RGB or RGBA value for all points.
```



In [ ]: