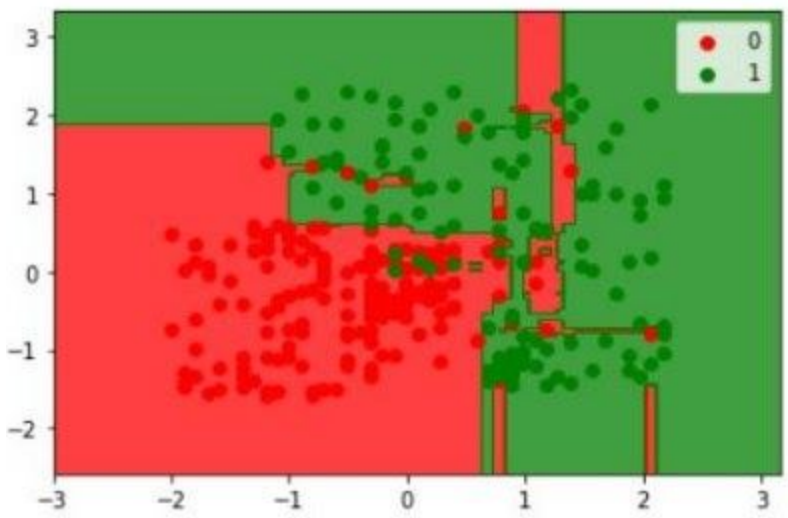
<u>Problem Statement- 1 :</u> Social_Network_Ads

AIM:

Try to understand the dataset of Social_Network_Ads.csv and try to find the best suitable ML algorithm and write the code in python for algorithm from scratch and try to achieve the below output plot

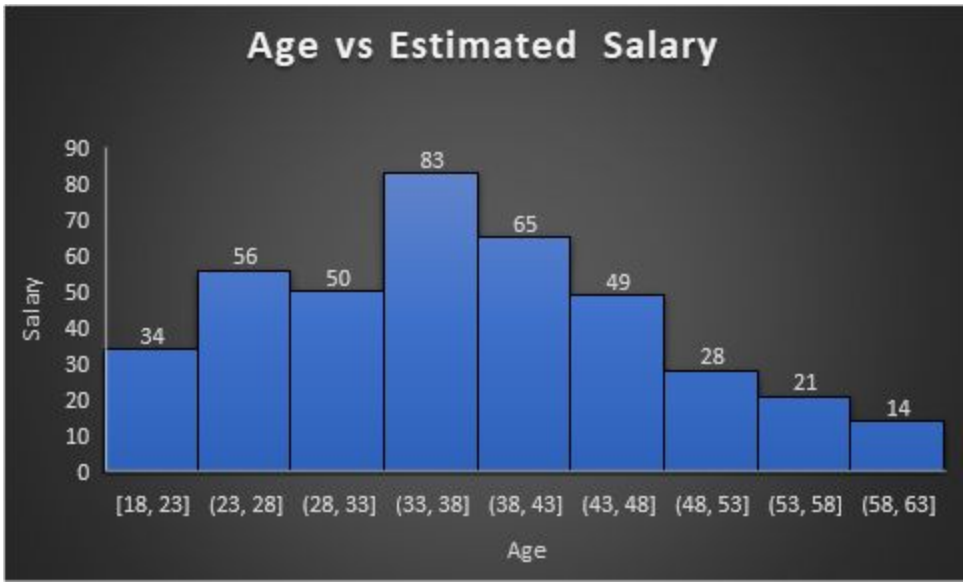


DATASET:

The Social_Network_Ads.csv is a Categorical dataset that contains the data about profiles of the users either he/she purchased the product or not.

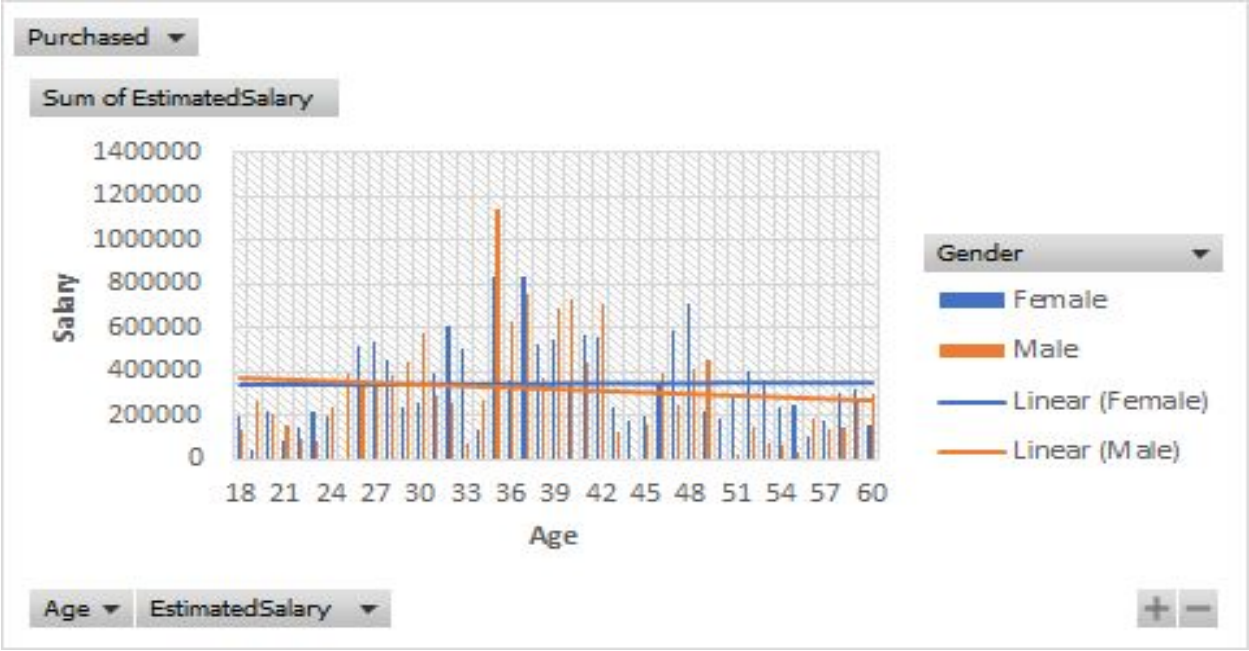
<u>Understanding the Dataset:</u>

For the purpose of understanding the Data, I have explored the data in excel *using charts and a Pivot table,* also I have inserted the screenshot of them below.

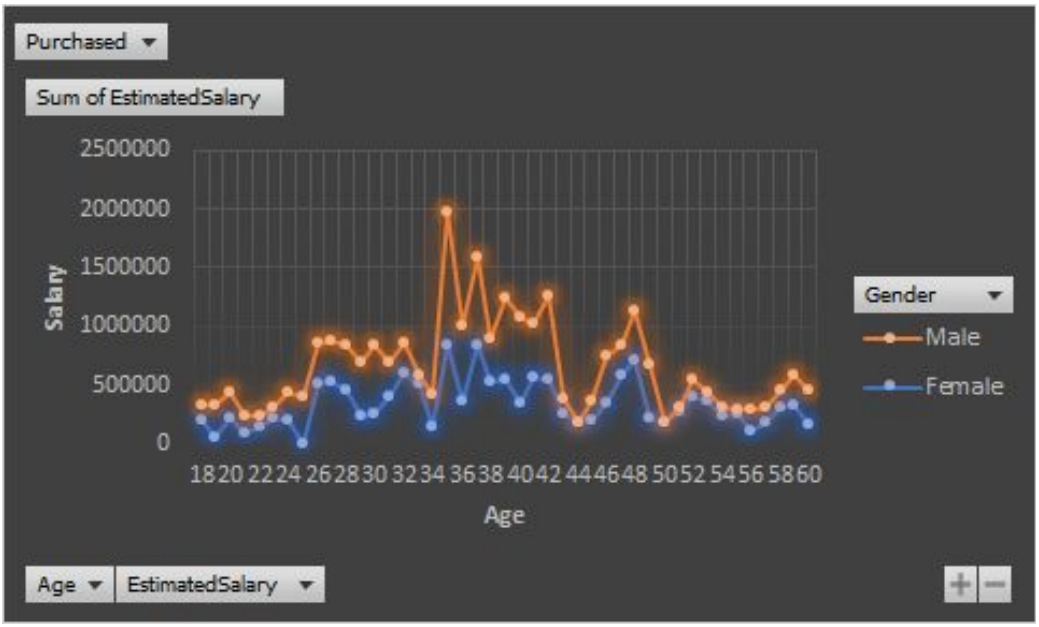


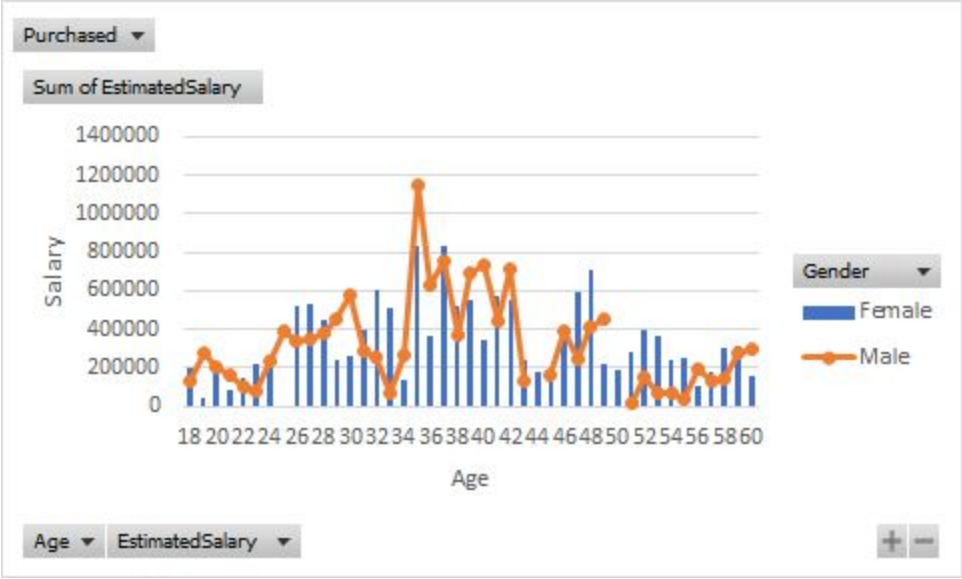
- The above Histogram shows that people of age between 33 to 38, get more salary than others.
- The Salary package increases from age 18, to a certain period of time upto age of 38 and then drops down slowly or gradually once the person gets older and older.
- Also many purchases are made by females rather than male people.

A Bar chart, Line Chart, Combo Chart of the Data from Pivot Table data respectively,



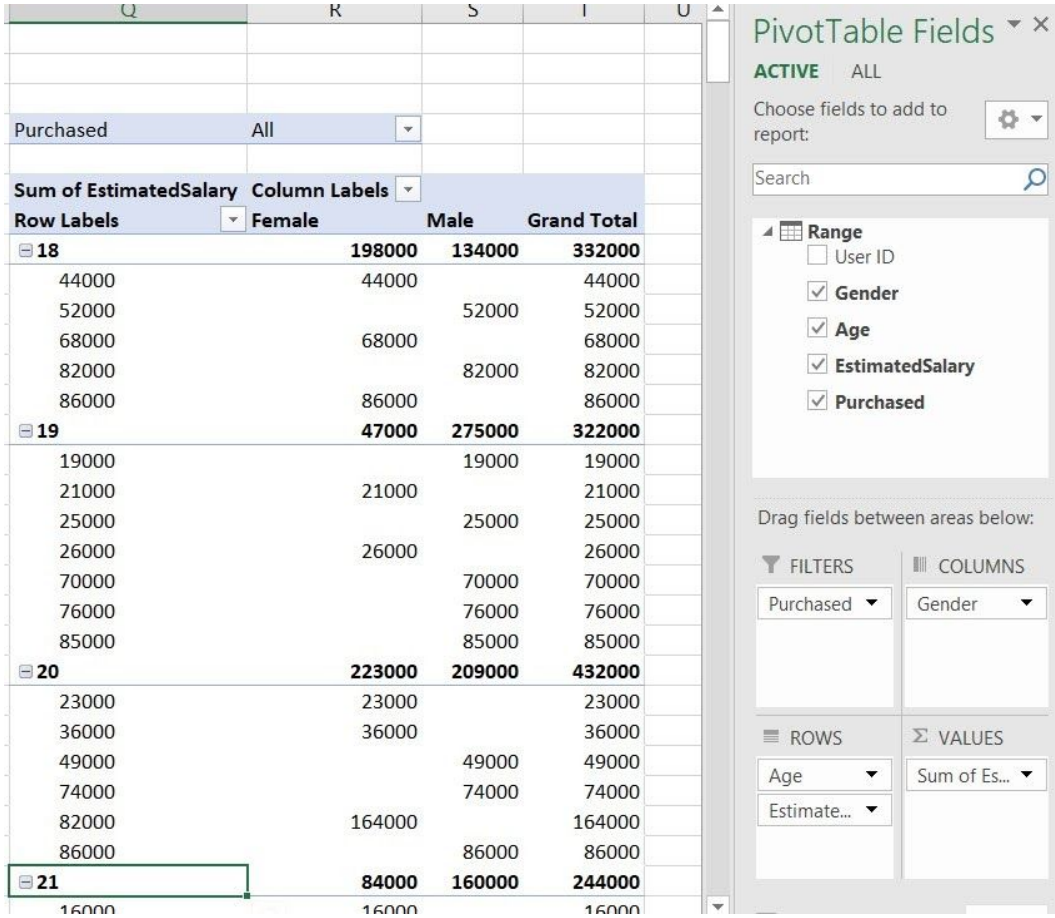
plot of age vs Salary along with Trendline





PIVOT TABLE:

An image of the pivot table that I have used is inserted along with its table fields



Purchased	All

Sum of EstimatedSalary	Column Labels		
Row Labels	Female	Male	Grand Total
18	198000	134000	332000
44000	44000		44000
52000		52000	52000
68000	68000		68000
82000		82000	82000
86000	86000		86000
19	47000	275000	322000
19000		19000	19000
21000	21000		21000
25000		25000	25000
26000	26000		26000
70000		70000	70000
76000		76000	76000
85000		85000	85000
20	223000	209000	432000
23000	23000		23000
36000	36000		36000
49000		49000	49000
74000		74000	74000
82000	164000		164000
86000		86000	86000
21	84000	160000	244000
16000	16000		16000

<u>SOLUTION:</u>

Here for this data I have used ***Random Forest Classifier***, instead of Logistic Regression and Naive Bayes Classifier, beacuse it is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and *uses averaging to improve the predictive accuracy and control over-fitting*.

<u>Note:</u> Using Scratch, though I tried my Best, I was only able to calculate the accuracy but not the Output Plot; But I have plotted the output using Built-in Libraries.

The explanation is inserted along with the code and I have inserted the screenshot of them below,

The <u>Accuracy</u> of the model is: ***0.863***

jupyter Social Network ads - Curneu Last Checkpoint: a minute ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

From Scratch

In [18]:
```
import numpy as np
from collections import Counter


def entropy(y):
    hist = np.bincount(y)
    ps = hist / len(y)
    return -np.sum([p * np.log2(p) for p in ps if p > 0])


class Node:

    def __init__(self, feature=None, threshold=None, left=None, right=None, *, value=None):
        self.feature = feature
        self.threshold = threshold
        self.left = left
        self.right = right
        self.value = value

    def is_leaf_node(self):
        return self.value is not None



#Decision Tree

class DecisionTree:

    def __init__(self, min_samples_split=2, max_depth=100, n_feats=None):
        self.min_samples_split = min_samples_split
        self.max_depth = max_depth
        self.n_feats = n_feats
        self.root = None

    def fit(self, X, y):
```

jupyter Social Network ads - Curneu Last Checkpoint: a minute ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
    def _information_gain(self, y, X_column, split_thresh):
        # parent loss
        parent_entropy = entropy(y)

        # generate split
        left_idxs, right_idxs = self._split(X_column, split_thresh)

        if len(left_idxs) == 0 or len(right_idxs) == 0:
            return 0

        # compute the weighted avg. of the loss for the children
        n = len(y)
        n_l, n_r = len(left_idxs), len(right_idxs)
        e_l, e_r = entropy(y[left_idxs]), entropy(y[right_idxs])
        child_entropy = (n_l / n) * e_l + (n_r / n) * e_r

        # information gain is difference in loss before vs. after split
        ig = parent_entropy - child_entropy
        return ig


    def _split(self, X_column, split_thresh):
        left_idxs = np.argwhere(X_column <= split_thresh).flatten()
        right_idxs = np.argwhere(X_column > split_thresh).flatten()
        return left_idxs, right_idxs


    def _traverse_tree(self, x, node):
        if node.is_leaf_node():
            return node.value

        if x[node.feature] <= node.threshold:
            return self._traverse_tree(x, node.left)
        return self._traverse_tree(x, node.right)

    def _most_common_label(self, y):
        counter = Counter(y)
        most_common = counter.most_common(1)[0][0]
```

jupyter Social Network ads - Curneu Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
    counter = Counter(y)
    most_common = counter.most_common(1)[0][0]
    return most_common



#Random Forest:

class RandomForest:

    def __init__(self, n_trees=10, min_samples_split=2,
                 max_depth=100, n_feats=None):
        self.n_trees = n_trees
        self.min_samples_split = min_samples_split
        self.max_depth = max_depth
        self.n_feats = n_feats
        self.trees = []

    def fit(self, X, y):
        self.trees = []
        for _ in range(self.n_trees):
            tree = DecisionTree(min_samples_split=self.min_samples_split,
                max_depth=self.max_depth, n_feats=self.n_feats)
            X_samp, y_samp = bootstrap_sample(X, y)
            tree.fit(X_samp, y_samp)
            self.trees.append(tree)

    def predict(self, X):
        tree_preds = np.array([tree.predict(X) for tree in self.trees])
        tree_preds = np.swapaxes(tree_preds, 0, 1)
        y_pred = [most_common_label(tree_pred) for tree_pred in tree_preds]
        return np.array(y_pred)

    def accuracy(y_true, y_pred):
        accuracy = np.sum(y_true == y_pred) / len(y_true)
        return accuracy
```

jupyter Social Network ads - Curneu Last Checkpoint: a minute ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
        tree_preds = np.swapaxes(tree_preds, 0, 1)
        y_pred = [most_common_label(tree_pred) for tree_pred in tree_preds]
        return np.array(y_pred)

    def accuracy(y_true, y_pred):
        accuracy = np.sum(y_true == y_pred) / len(y_true)
        return accuracy
```

In [19]:

```
#Load the Dataset:
data = pd.read_csv('Social_Network_Ads.csv')
X = data.iloc[:, [2, 3]].values
y = data.iloc[:, 4].values

#Train Test Split:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)

clf = RandomForest(n_trees=3, max_depth=10)

#fit the model:
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

#Finding Accuracy:
acc = accuracy(y_test, y_pred)
print ("Accuracy:", acc)
```

```
Accuracy: 0.8625
```

Note: Using scratch, though i tried my best, i was only able to print the 'accuracy' of the model; but not the required output plot.

I have brought the outplot plot using built-in Libraries, which is given below

jupyter Social Network ads - Curneu Last Checkpoint: 3 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [ ]:

Using In-built Libraries

In [20]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

from matplotlib.colors import ListedColormap
from matplotlib import pyplot as plt
```

In [21]:
```
# Random forest Classifier:

classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

In [22]:
```
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')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-22-e2dbafe34fa3> in <module>
      1 X_set, y_set = X_test, y_test
      2 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
----> 3                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
      4 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
      5              alpha = 0.75, cmap = ListedColormap(('red', 'green')))
```

jupyter Social Network ads - Curneu Last Checkpoint: 4 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

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

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-22-e2dbafe34fa3> in <module>
      1 X_set, y_set = X_test, y_test
      2 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
----> 3                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
      4 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
      5             alpha = 0.75, cmap = ListedColormap(('red', 'green')))

~\Anaconda3\lib\site-packages\numpy\lib\function_base.py in meshgrid(*xi, **kwargs)
   4058
   4059     if copy_:
-> 4060         output = [x.copy() for x in output]
   4061
   4062     return output

~\Anaconda3\lib\site-packages\numpy\lib\function_base.py in <listcomp>(.0)
   4058
   4059     if copy_:
-> 4060         output = [x.copy() for x in output]
   4061
   4062     return output

MemoryError:
```

Note: Since i run the above snippet more than once, it lead to a memory error & I'm unable to solve it. Sorry !!

But, I have attached the screenshot of the output plot along with the snippet when it occured for the first time.

In [ ]:

jupyter social network ads - Curneu Last Checkpoint: 2 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

In [35]:

```
from matplotlib.colors import ListedColormap
from matplotlib import pyplot as plt
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 should be avoided as value-mapping will have precedence in case its length 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 should be avoided as value-mapping will have precedence in case its length 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.
```