2440016804 - Rio Pramana - LA01 - Assignment 7

Import libraries and dataset

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

In [2]:
```
# Importing the dataset, downloaded file is in the same folder
csv_path = "Social_Network_Ads.csv"
ads_df = pd.read_csv(csv_path)
```

Run a quick check on the dataset

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

Out[3]:
```
(400, 3)
```

In [4]:
```
ads_df.head(5)
```

Out[4]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [5]:
```
ads_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              400 non-null    int64
 1   EstimatedSalary  400 non-null    int64
 2   Purchased        400 non-null    int64
dtypes: int64(3)
memory usage: 9.5 KB
```

The independent variables are all numeric and there's no missing data

In [6]:
```
ads_df.describe()
```

Out[6]:

	Age	EstimatedSalary	Purchased
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	88000.000000	1.000000
max	60.000000	150000.000000	1.000000

In [7]:
```
#Extracting independent variables:
x = ads_df.iloc[:,:-1].values #Extract semua kolom kecuali kolom terakhir
print(x)
```

```
[[    19  19000]
 [    35  20000]
 [    26  43000]
 [    27  57000]
 [    19  76000]
 [    27  58000]
 [    27  84000]
 [    32 150000]
 [    25  33000]
 [    35  65000]
 [    26  80000]
 [    26  52000]
 [    20  86000]
 [    32  18000]
 [    18  82000]
 [    29  80000]
 [    47  25000]
 [    45  26000]
 [    46  28000]
 [    48  29000]
 [    45  22000]
 [    47  49000]
 [    48  41000]
 [    45  22000]
 [    46  23000]
 [    47  20000]
 [    49  28000]
 [    47  30000]
 [    29  43000]
 [    31  18000]
 [    31  74000]
 [    27 137000]
 [    21  16000]
 [    28  44000]
 [    27  90000]
 [    35  27000]
 [    33  28000]
 [    30  49000]
 [    26  72000]
 [    27  31000]
 [    27  17000]
 [    33  51000]
 [    35 108000]
 [    30  15000]
 [    28  84000]
 [    23  20000]
 [    25  79000]
 [    27  54000]
 [    30 135000]
 [    31  89000]
 [    24  32000]
 [    18  44000]
 [    29  83000]
 [    35  23000]
 [    27  58000]
 [    24  55000]
 [    23  48000]
 [    28  79000]
 [    22  18000]
 [    32 117000]
 [    27  20000]
 [    25  87000]
 [    23  66000]
 [    32 120000]
 [    59  83000]
 [    24  58000]
 [    24  19000]
 [    23  82000]
 [    22  63000]
 [    31  68000]
 [    25  80000]
 [    24  27000]
 [    20  23000]
 [    33 113000]
 [    32  18000]
 [    34 112000]
 [    18  52000]
 [    22  27000]
 [    28  87000]
 [    26  17000]
 [    30  80000]
 [    39  42000]
 [    20  49000]
 [    35  88000]
 [    30  62000]
 [    31 118000]
 [    24  55000]
 [    28  85000]
 [    26  81000]
 [    35  50000]
 [    22  81000]
 [    30 116000]
 [    26  15000]
 [    29  28000]
 [    29  83000]
 [    35  44000]
 [    35  25000]
 [    28 123000]
 [    35  73000]
 [    28  37000]
 [    27  88000]
 [    28  59000]
 [    32  86000]
 [    33 149000]
 [    19  21000]
 [    21  72000]
 [    26  35000]
 [    27  89000]
 [    26  86000]
 [    38  80000]
 [    39  71000]
 [    37  71000]
 [    38  61000]
 [    37  55000]
 [    42  80000]
 [    40  57000]
 [    35  75000]
 [    36  52000]
 [    40  59000]
 [    41  59000]
 [    36  75000]
 [    37  72000]
 [    40  75000]
 [    35  53000]
 [    41  51000]
 [    39  61000]
 [    42  65000]
 [    26  32000]
 [    30  17000]
 [    26  84000]
 [    31  58000]
 [    33  31000]
 [    30  87000]
 [    21  68000]
 [    28  55000]
 [    23  63000]
 [    20  82000]
 [    30 107000]
 [    28  59000]
 [    19  25000]
 [    19  85000]
 [    18  68000]
 [    35  59000]
 [    30  89000]
 [    34  25000]
 [    24  89000]
 [    27  96000]
 [    41  30000]
 [    29  61000]
 [    20  74000]
 [    26  15000]
 [    41  45000]
 [    31  76000]
 [    36  50000]
 [    40  47000]
 [    31  15000]
 [    46  59000]
 [    29  75000]
 [    26  30000]
 [    32 135000]
 [    32 100000]
 [    25  90000]
 [    37  33000]
 [    35  38000]
 [    33  69000]
 [    18  86000]
 [    22  55000]
 [    35  71000]
 [    29 148000]
 [    29  47000]
 [    21  88000]
 [    34 115000]
 [    26 118000]
 [    34  43000]
 [    34  72000]
 [    23  28000]
 [    35  47000]
 [    25  22000]
 [    24  23000]
 [    31  34000]
 [    26  16000]
 [    31  71000]
 [    32 117000]
 [    33  43000]
 [    33  60000]
 [    31  66000]
 [    20  82000]
 [    33  41000]
 [    35  72000]
 [    28  32000]
 [    24  84000]
 [    19  26000]
 [    29  43000]
 [    19  70000]
 [    28  89000]
 [    34  43000]
 [    30  79000]
 [    20  36000]
 [    26  80000]
 [    35  22000]
 [    35  39000]
 [    49  74000]
 [    39 134000]
 [    41  71000]
 [    58 101000]
 [    47  47000]
 [    55 130000]
 [    52 114000]
 [    40 142000]
 [    46  22000]
 [    48  96000]
 [    52 150000]
 [    59  42000]
 [    35  58000]
 [    47  43000]
 [    60 108000]
 [    49  65000]
 [    40  78000]
 [    46  96000]
 [    59 143000]
 [    41  80000]
 [    35  91000]
 [    37 144000]
 [    60 102000]
 [    35  60000]
 [    37  53000]
 [    36 126000]
 [    56 133000]
 [    40  72000]
 [    42  80000]
 [    35 147000]
 [    39  42000]
 [    40 107000]
 [    49  86000]
 [    38 112000]
 [    46  79000]
 [    40  57000]
 [    37  80000]
 [    46  82000]
 [    53 143000]
 [    42 149000]
 [    38  59000]
 [    50  88000]
 [    56 104000]
 [    41  72000]
 [    51 146000]
 [    35  50000]
 [    57 122000]
 [    41  52000]
 [    35  97000]
 [    44  39000]
 [    37  52000]
 [    48 134000]
 [    37 146000]
 [    50  44000]
 [    52  90000]
 [    41  72000]
 [    40  57000]
 [    58  95000]
 [    45 131000]
 [    35  77000]
 [    36 144000]
 [    55 125000]
 [    35  72000]
 [    48  90000]
 [    42 108000]
 [    40  75000]
 [    37  74000]
 [    47 144000]
 [    40  61000]
 [    43 133000]
 [    59  76000]
 [    60  42000]
 [    39 106000]
 [    57  26000]
 [    57  74000]
 [    38  71000]
 [    49  88000]
 [    52  38000]
 [    50  36000]
 [    59  88000]
 [    35  61000]
 [    37  70000]
 [    52  21000]
 [    48 141000]
 [    37  93000]
 [    37  62000]
 [    48 138000]
 [    41  79000]
 [    37  78000]
 [    39 134000]
 [    49  89000]
 [    55  39000]
 [    37  77000]
 [    35  57000]
 [    36  63000]
 [    42  73000]
 [    43 112000]
 [    45  79000]
 [    46 117000]
 [    58  38000]
 [    48  74000]
 [    37 137000]
 [    37  79000]
 [    40  60000]
 [    42  54000]
 [    51 134000]
 [    47 113000]
 [    36 125000]
 [    38  50000]
 [    42  70000]
 [    39  96000]
 [    38  50000]
 [    49 141000]
 [    39  79000]
 [    39  75000]
 [    54 104000]
 [    35  55000]
 [    45  32000]
 [    36  60000]
 [    52 138000]
 [    53  82000]
 [    41  52000]
 [    48  30000]
 [    48 131000]
 [    41  60000]
 [    41  72000]
 [    42  75000]
 [    36 118000]
 [    47 107000]
 [    38  51000]
 [    48 119000]
 [    42  65000]
 [    40  65000]
 [    57  60000]
 [    36  54000]
 [    58 144000]
 [    35  79000]
 [    38  55000]
 [    39 122000]
 [    53 104000]
 [    35  75000]
 [    38  65000]
 [    47  51000]
 [    47 105000]
 [    41  63000]
 [    53  72000]
 [    54 108000]
 [    39  77000]
 [    38  61000]
 [    38 113000]
 [    37  75000]
 [    42  90000]
 [    37  57000]
 [    36  99000]
 [    60  34000]
 [    54  70000]
 [    41  72000]
 [    40  71000]
 [    42  54000]
 [    43 129000]
 [    53  34000]
 [    47  50000]
 [    42  79000]
 [    42 104000]
 [    59  29000]
 [    58  47000]
 [    46  88000]
 [    38  71000]
 [    54  26000]
 [    60  46000]
 [    60  83000]
 [    39  73000]
 [    59 130000]
 [    37  80000]
 [    46  32000]
 [    46  74000]
 [    42  53000]
 [    41  87000]
 [    58  23000]
 [    42  64000]
 [    48  33000]
 [    44 139000]
 [    49  28000]
 [    57  33000]
 [    56  60000]
 [    49  39000]
 [    39  71000]
 [    47  34000]
 [    48  35000]
 [    48  33000]
 [    47  23000]
 [    45  45000]
 [    60  42000]
 [    39  59000]
 [    46  41000]
 [    51  23000]
 [    50  20000]
 [    36  33000]
 [    49  36000]]
```

In [8]:
```
#Extracting dependent variable:
y = ads_df.iloc[:,2].values #Extract kolom terakhir
print(y)
```

```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1
 1 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1
 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0
 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1
 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1]
```

Splitting training and test data

Before we do feature scaling, we must split our data first

With 400 data, I decided to split the dataset into 3:1 ratio (Training set contains 300 data, Test set contains 100 data)

In [9]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
print('Train set: ', x_train.shape, y_train.shape)
print('Test set: ', x_test.shape, y_test.shape)
```

```
Train set:  (300, 2) (300,)
Test set:  (100, 2) (100,)
```

Feature Scaling

In [10]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [11]:
```
x_train[0:5]
```

Out[11]:
```
array([[ 0.58164944, -0.88670699],
       [-0.60673761,  1.46173768],
       [-0.01254409, -0.5677824 ],
       [-0.60673761,  1.89663484],
       [ 1.37390747, -1.40858358]])
```

In [12]:
```
x_test[0:5]
```

Out[12]:
```
array([[-0.54748976,  0.5130727 ],
       [ 0.15442019, -0.61825566],
       [-0.10879604,  0.14615539],
       [-0.54748976,  0.26846116],
       [-0.10879604, -0.61825566]])
```

Predicting using K-NN Classifier (Minkowski)

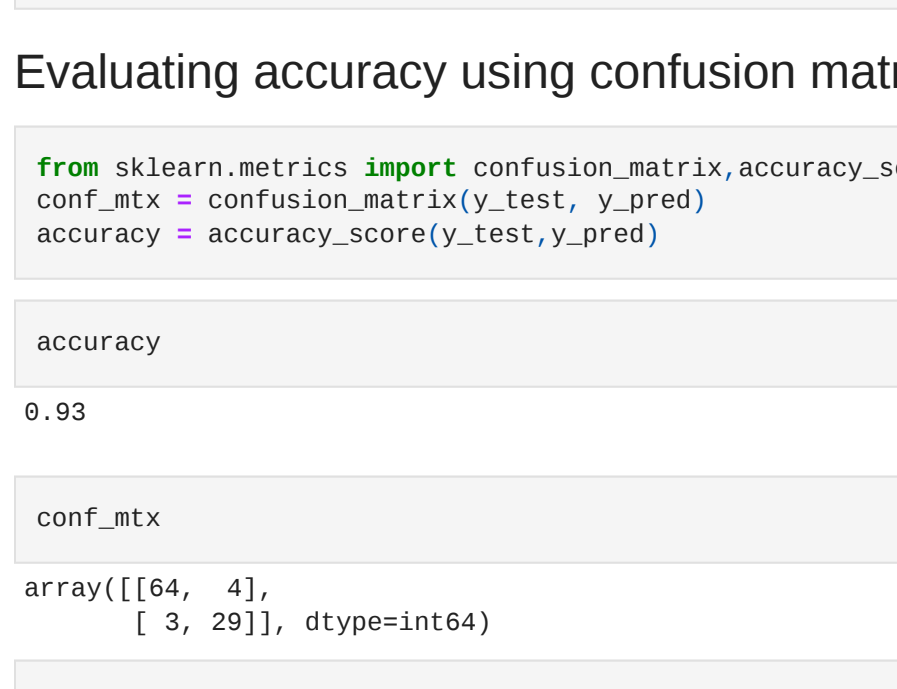
In [13]:
```
#Import KNN classifier
from sklearn.neighbors import KNeighborsClassifier
```

Finding the best number of K's to use for best accuracy

In [14]:
```
from sklearn import metrics
K = 10 #Finding best number of K's between 1-9 (inclusive)
mean_acc = np.zeros((K-1))
std_acc = np.zeros((K-1))
ConfusionMatrix = []
for n in range(1,K):
    #Training the model and predict
    neighb = KNeighborsClassifier(n_neighbors = n, metric = 'minkowski').fit(x_train, y_train) #using Minkowski distance metric
    yhat = neighb.predict(x_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)

    std_acc[n-1] = np.std(yhat == y_test) / np.sqrt(yhat.shape[0])

mean_acc
```

Out[14]:
```
array([0.88, 0.88, 0.93, 0.93, 0.93, 0.93, 0.93, 0.93, 0.93])
```

In [15]:
```
plt.plot(range(1,K), mean_acc, 'g')
plt.fill_between(range(1,K), mean_acc - 1 * std_acc, mean_acc + 1 * std_acc, alpha = 0.1)
plt.legend(('Accuracy', '+/- 3xstd'))
plt.ylabel('Accuracy')
plt.xlabel('Number of Neighbors (K)')
plt.tight_layout()
plt.show()
```



In [16]:
```
print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.93 with k= 3
```

We found that k = 3 resulted in best accuracy (even though the same thing could be said for k = 4 to k = 9)

So, for the classifier, we'll use K = 3

Predict using K-NN Classifier with K = 3

In [17]:
```
classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski')
classifier.fit(x_train, y_train)
classifier
```

Out[17]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [18]:
```
#Predicting
y_pred = classifier.predict(x_test)
```

Evaluating accuracy using confusion matrix

In [19]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
conf_mtx = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test,y_pred)
```

In [20]:
```
accuracy
```

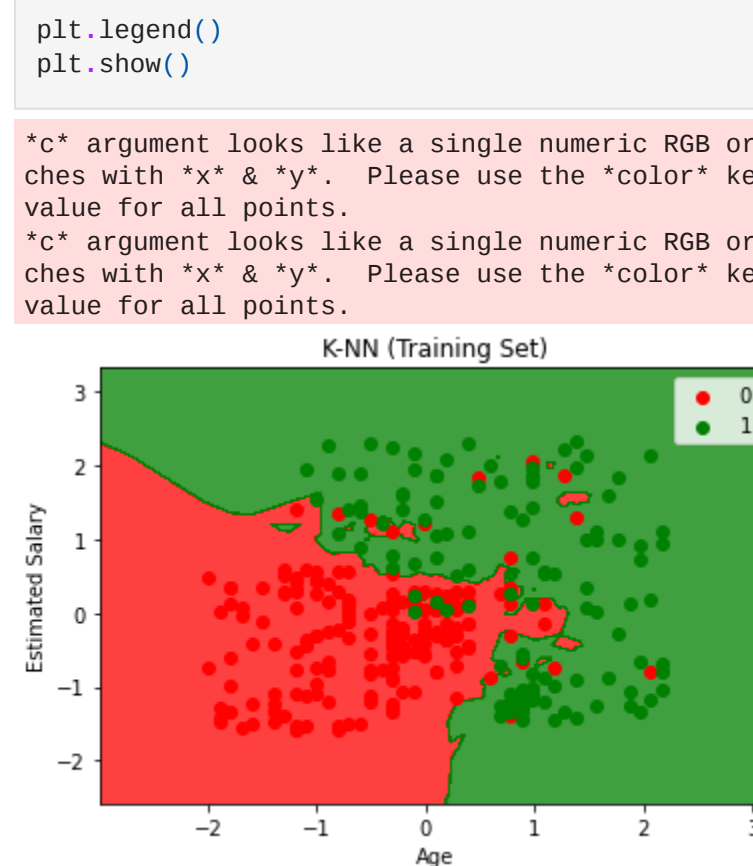
Out[20]:
```
0.93
```

In [21]:
```
conf_mtx
```

Out[21]:
```
array([[64,  4],
       [ 3, 29]], dtype=int64)
```

In [22]:
```
#Visualizing confusion matrix
sns.heatmap(conf_mtx, annot = True)
```

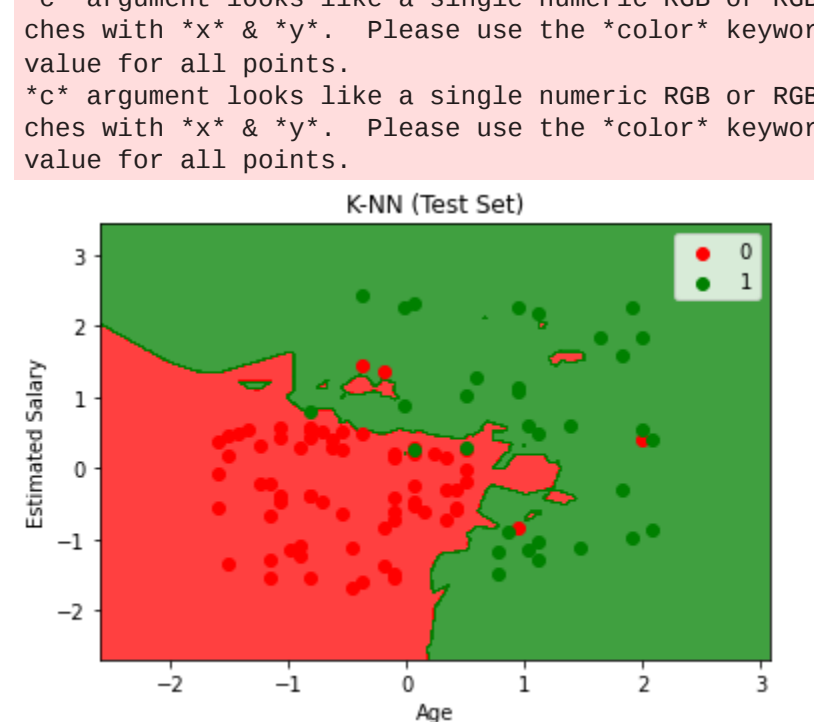
Out[22]:
```
<AxesSubplot:>
```



Visualizing the Training and Test set results

In [23]:
```
from matplotlib.colors import ListedColormap

x1, x2 = np.meshgrid(np.arange(start = x_train[:, 0].min()-1, stop = x_train[:, 0].max()+1, step = 0.01),
                     np.arange(start = x_train[:, 1].min()-1, stop = x_train[:, 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_train)):
    plt.scatter(x_train[y_train == j, 0], x_train[y_train == j, 1],
    c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('K-NN (Training 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 the *color* keyword-argument or provide a 2D array with a single row if you intend 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 the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```



In [24]:
```
x1, x2 = np.meshgrid(np.arange(start = x_test[:, 0].min()-1, stop = x_test[:, 0].max()+1, step = 0.01),
                     np.arange(start = x_test[:, 1].min()-1, stop = x_test[:, 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_test)):
    plt.scatter(x_test[y_test == j, 0], x_test[y_test == 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 should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you intend 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 the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```