In [1]:

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
from sklearn import metrics

#importing datasets
data_set= pd.read_csv('user_data.csv')

#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values

# Splitting the dataset into training and test set.
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)

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

In [2]:

```
data_set
```

Out[2]:

	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
...	...	...	...	...	...
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

400 rows × 5 columns

In [3]:

```
x_test
```

Out[3]:

```
array([[-0.80480212,  0.50496393],
       [-0.01254409, -0.5677824 ],
       [-0.30964085,  0.1570462 ],
       [-0.80480212,  0.27301877],
       [-0.30964085, -0.5677824 ],
       [-1.10189888, -1.43757673],
       [-0.70576986, -1.58254245],
       [-0.21060859,  2.15757314],
       [-1.99318916, -0.04590581],
       [ 0.8787462 , -0.77073441],
       [-0.80480212, -0.59677555],
```

```
[-1.00286662, -0.42281668],
[-0.11157634, -0.42281668],
[ 0.08648817,  0.21503249],
[-1.79512465,  0.47597078],
[-0.60673761,  1.37475825],
[-0.11157634,  0.21503249],
[-1.89415691,  0.44697764],
[ 1.67100423,  1.75166912],
[-0.30964085, -1.37959044],
[-0.30964085, -0.65476184],
[ 0.8787462 ,  2.15757314],
[ 0.28455268, -0.53878926],
[ 0.8787462 ,  1.02684052],
[-1.49802789, -1.20563157],
[ 1.07681071,  2.07059371],
[-1.00286662,  0.50496393],
[-0.90383437,  0.30201192],
[-0.11157634, -0.21986468],
[-0.60673761,  0.47597078],
[-1.6960924 ,  0.53395707],
[-0.11157634,  0.27301877],
[ 1.86906873, -0.27785096],
[-0.11157634, -0.48080297],
[-1.39899564, -0.33583725],
[-1.99318916, -0.50979612],
[-1.59706014,  0.33100506],
[-0.4086731 , -0.77073441],
[-0.70576986, -1.03167271],
[ 1.07681071, -0.97368642],
[-1.10189888,  0.53395707],
[ 0.28455268, -0.50979612],
[-1.10189888,  0.41798449],
[-0.30964085, -1.43757673],
[ 0.48261718,  1.22979253],
[-1.10189888, -0.33583725],
[-0.11157634,  0.30201192],
[ 1.37390747,  0.59194336],
[-1.20093113, -1.14764529],
[ 1.07681071,  0.47597078],
[ 1.86906873,  1.51972397],
[-0.4086731 , -1.29261101],
[-0.30964085, -0.3648304 ],
[-0.4086731 ,  1.31677196],
[ 2.06713324,  0.53395707],
[ 0.68068169, -1.089659  ],
[-0.90383437,  0.38899135],
[-1.20093113,  0.30201192],
[ 1.07681071, -1.20563157],
[-1.49802789, -1.43757673],
[-0.60673761, -1.49556302],
[ 2.1661655 , -0.79972756],
[-1.89415691,  0.18603934],
[-0.21060859,  0.85288166],
[-1.89415691, -1.26361786],
[ 2.1661655 ,  0.38899135],
[-1.39899564,  0.56295021],
[-1.10189888, -0.33583725],
[ 0.18552042, -0.65476184],
[ 0.38358493,  0.01208048],
[-0.60673761,  2.331532  ],
[-0.30964085,  0.21503249],
[-1.59706014, -0.19087153],
[ 0.68068169, -1.37959044],
[-1.10189888,  0.56295021],
[-1.99318916,  0.35999821],
[ 0.38358493,  0.27301877],
[ 0.18552042, -0.27785096],
[ 1.47293972, -1.03167271],
[ 0.8787462 ,  1.08482681],
[ 1.10099,  2.15757314],
```

```
       [ 2.06713324,  0.38899135],
       [-1.39899564, -0.42281668],
       [-1.20093113, -1.00267957],
       [ 1.96810099, -0.91570013],
       [ 0.38358493,  0.30201192],
       [ 0.18552042,  0.1570462 ],
       [ 2.06713324,  1.75166912],
       [ 0.77971394, -0.8287207 ],
       [ 0.28455268, -0.27785096],
       [ 0.38358493, -0.16187839],
       [-0.11157634,  2.21555943],
       [-1.49802789, -0.62576869],
       [-1.29996338, -1.06066585],
       [-1.39899564,  0.41798449],
       [-1.10189888,  0.76590222],
       [-1.49802789, -0.19087153],
       [ 0.97777845, -1.06066585],
       [ 0.97777845,  0.59194336],
       [ 0.38358493,  0.99784738]])
```

In [4]:
```
y_test
```

Out[4]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1], dtype=int64)
```

In [5]:
```
#Fitting K-NN classifier 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[5]:
```
KNeighborsClassifier()
```

In [6]:
```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

In [7]:
```
y_pred
```

Out[7]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1], dtype=int64)
```

In [8]:
```
#Now we will create the Confusion Matrix for our K-NN model to see the accuracy of the cla
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

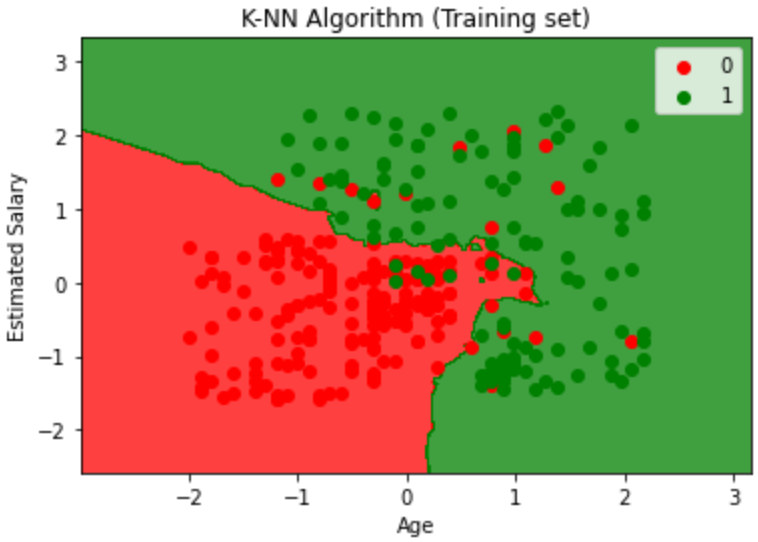
In [9]:
```
cm
```

Out[9]:
```
array([[64,  4],
       [ 3, 29]], dtype=int64)
```

In [10]:
```
#Visulaizing the trianing set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
```

```
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.s
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('K-NN Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.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 2-D array with a single row if you intend to spe
cify 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 2-D array with a single row if you intend to spe
cify the same RGB or RGBA value for all points.
```



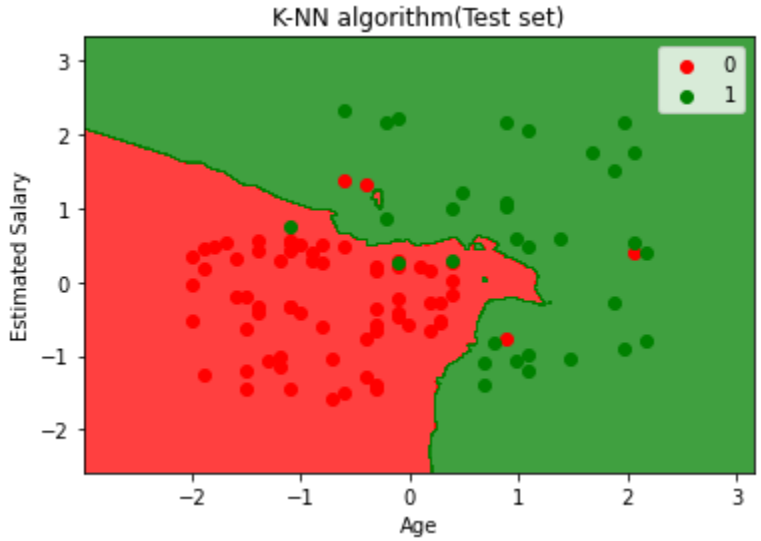
In [11]:
```
"""
As we can see the graph is showing the red point and green points.
The green points are for Purchased(1) and Red Points for not Purchased(0) variable.
The graph is showing an irregular boundary instead of showing any straight line or any cur
"""
```

Out[11]:
```
'\nAs we can see the graph is showing the red point and green points. \nThe green points a
re for Purchased(1) and Red Points for not Purchased(0) variable.\nThe graph is showing an
irregular boundary instead of showing any straight line or any curve because it is a K-NN
algorithm, \n'
```

In [14]:
```
#Visualizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.s
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
```

```
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('K-NN algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.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 2-D array with a single row if you intend to spe
cify 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 2-D array with a single row if you intend to spe
cify the same RGB or RGBA value for all points.
```



In [15]:

```
accuracy = metrics.accuracy_score(y_test,y_pred)
report = metrics.classification_report(y_test,y_pred)
cm = metrics.confusion_matrix(y_test,y_pred)

print("Classification report:")
print("Accuracy: ", accuracy)
print(report)
print("Confusion matrix:")
print(cm)
```

```
Classification report:
Accuracy:  0.93
              precision    recall  f1-score   support

           0       0.96      0.94      0.95        68
           1       0.88      0.91      0.89        32

    accuracy                           0.93       100
   macro avg       0.92      0.92      0.92       100
weighted avg       0.93      0.93      0.93       100

Confusion matrix:
[[64  4]
 [ 3 29]]
```

In [ ]:

```
```