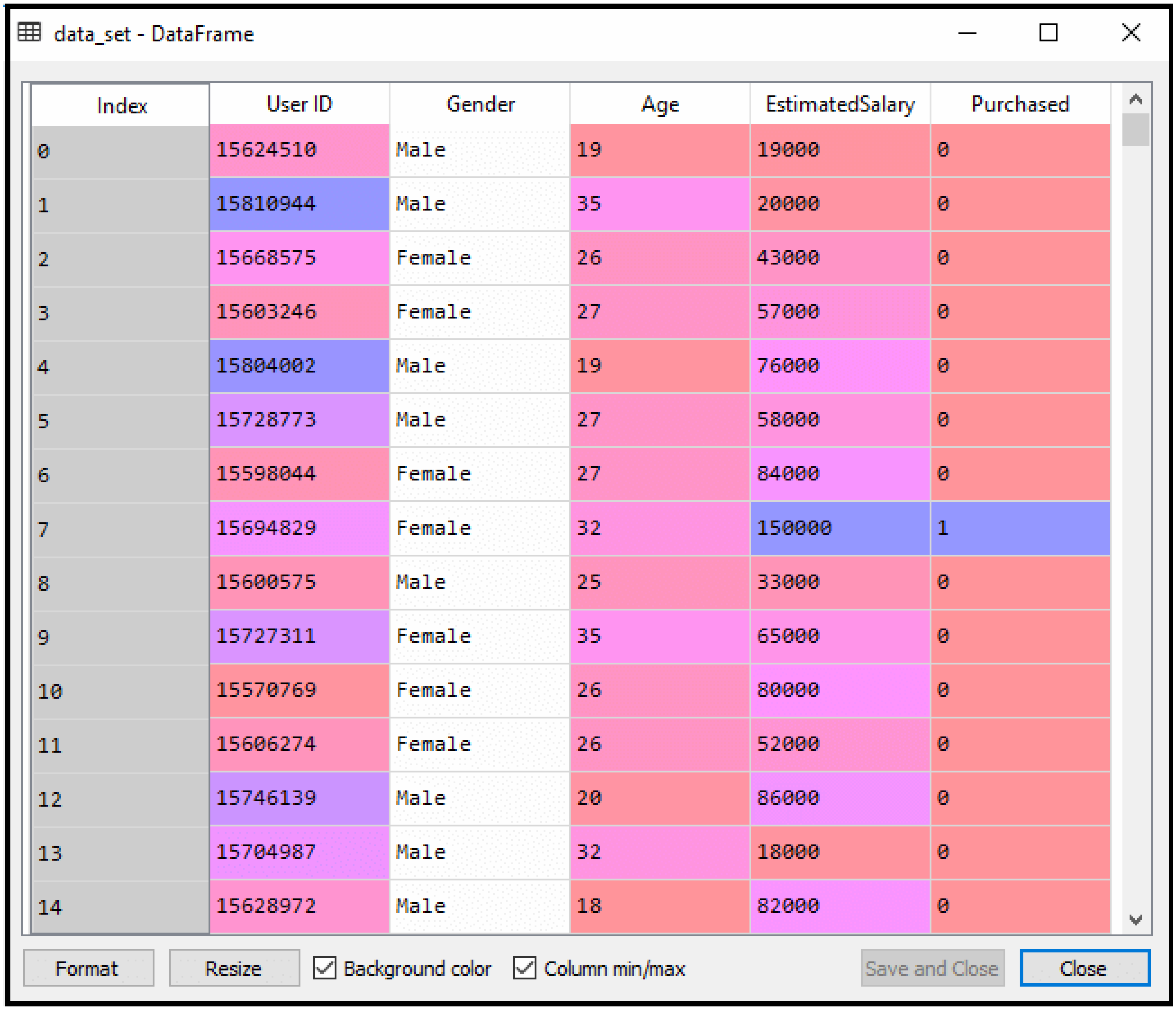
Data Pre-processing step

Below is the code:

#Data Pre-processing Step

```
1. # importing libraries
2. import numpy as nm
3. import matplotlib.pyplot as mtp
4. import pandas as pd


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

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

10.# Splitting the dataset into training and test set.
11. from sklearn.model_selection import train_test_split
12. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)

13.#feature Scaling
14. from sklearn.preprocessing import StandardScaler
15. st_x= StandardScaler()
16. x_train= st_x.fit_transform(x_train)
17. x_test= st_x.transform(x_test)
```

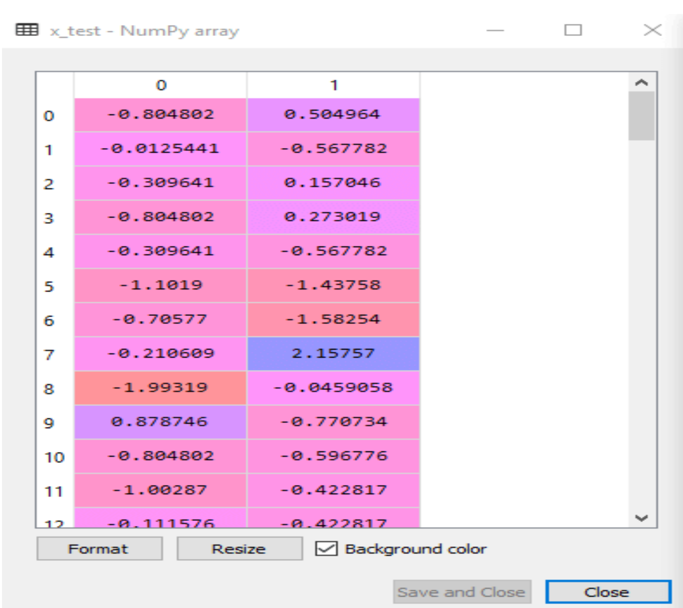
- After executing the above code, we will pre-process the data.
- The code will give the dataset as:



data_set - DataFrame

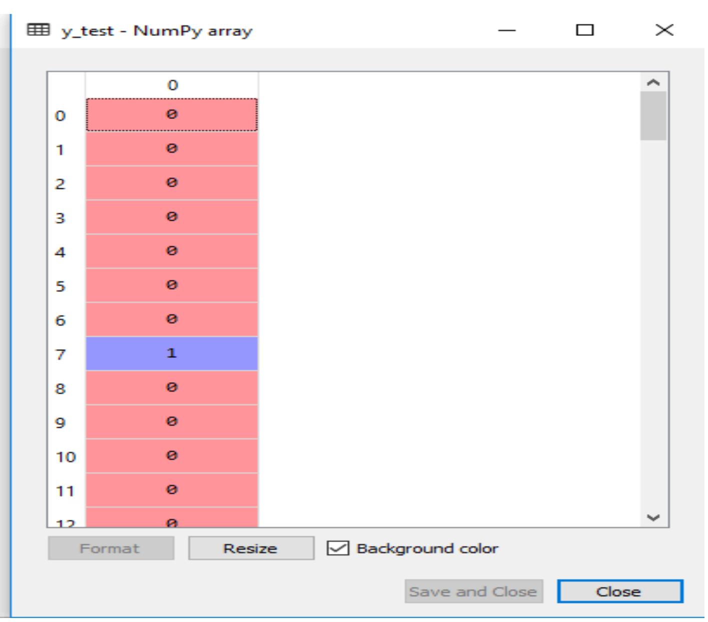
Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0

The scaled output for the test set will be:



x_test - NumPy array

	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817



y_test - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

Fitting the SVM classifier to the training set:

- Now the training set will be fitted to the SVM classifier. To create the SVM classifier, we will import **SVC** class from **Sklearn.svm** library.
- Below is the code for it:

```
1. from sklearn.svm import SVC # "Support vector classifier"
2. classifier = SVC(kernel='linear', random_state=0)
3. classifier.fit(x_train, y_train)
```

- In the above code, we have used **kernel='linear'**, as here we are creating SVM for linearly separable data.
- However, we can change it for non-linear data. And then we fitted the classifier to the training dataset(x_train, y_train)

Output:

```
Out[8]:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=0,
    shrinking=True, tol=0.001, verbose=False)
```

The model performance can be altered by changing the value of **C(Regularization factor), gamma, and kernel**.

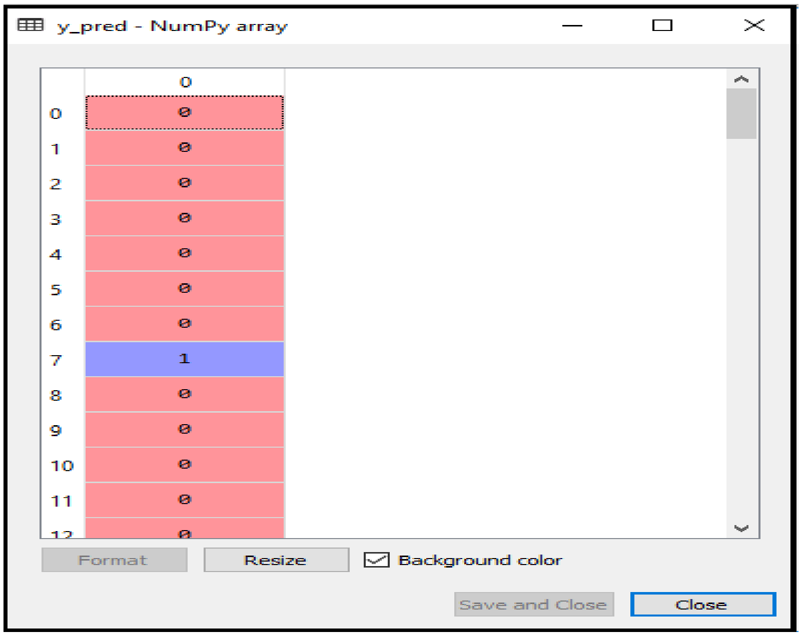
Predicting the test set result:

Now, we will predict the output for test set. For this, we will create a new vector y_pred. Below is the code for it:

```
1. #Predicting the test set result
2. y_pred= classifier.predict(x_test)
```

After getting the y_pred vector, we can compare the result of **y_pred** and **y_test** to check the difference between the actual value and predicted value.

Output: Below is the output for the prediction of the test set:



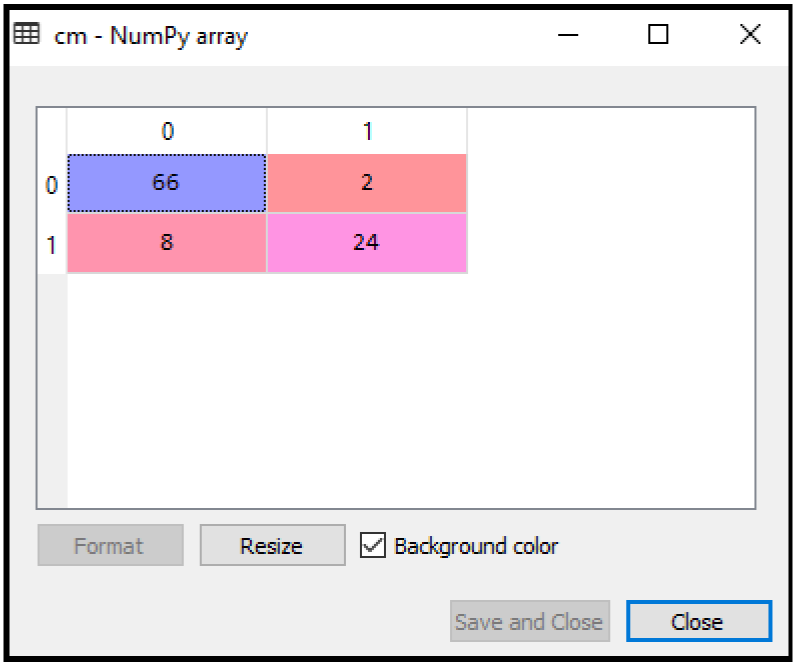
Creating the confusion matrix:

Now we will see the performance of the SVM classifier that how many incorrect predictions are there as compared to the Logistic regression classifier.

- To create the confusion matrix, we need to import the **confusion_matrix** function of the sklearn library. After importing the function, we will call it using a new variable **cm**.
- The function takes two parameters, mainly **y_true**(the actual values) and **y_pred** (the targeted value return by the classifier). Below is the code for it:

```
1. #Creating the Confusion matrix
2. from sklearn.metrics import confusion_matrix
3. cm= confusion_matrix(y_test, y_pred)
```

Output:



- As we can see in the above output image, there are 66+24= 90 correct predictions and 8+2= 10 correct predictions.
- Therefore we can say that our SVM model improved as compared to the Logistic regression model.

Visualizing the training set result:

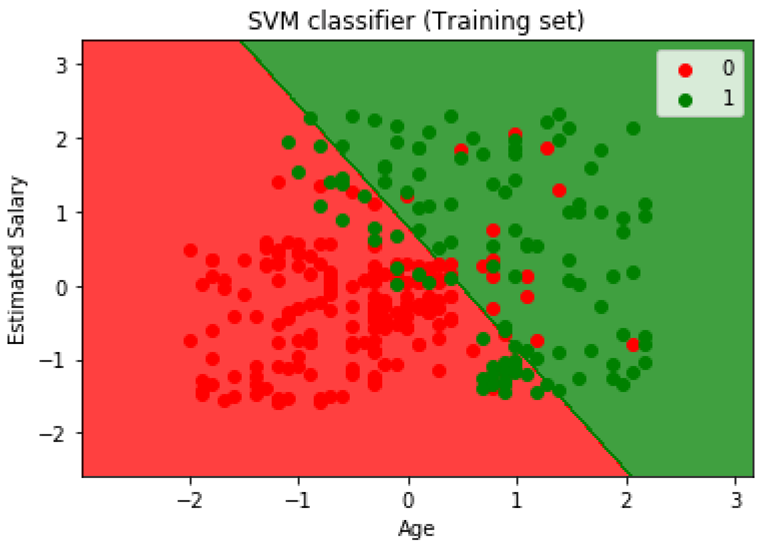
Now we will visualize the training set result, below is the code for it:

```
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, step  =0.01),
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.shape),
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('SVM classifier (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
```

15. mtp.legend()
16. mtp.show()

Output:

By executing the above code, we will get the output as:



- As we can see, the above output is appearing similar to the Logistic regression output.
- In the output, we got the straight line as hyperplane because we have **used a linear kernel in the classifier**.
- And we have also discussed above that for the 2d space, the hyperplane in SVM is a straight line.

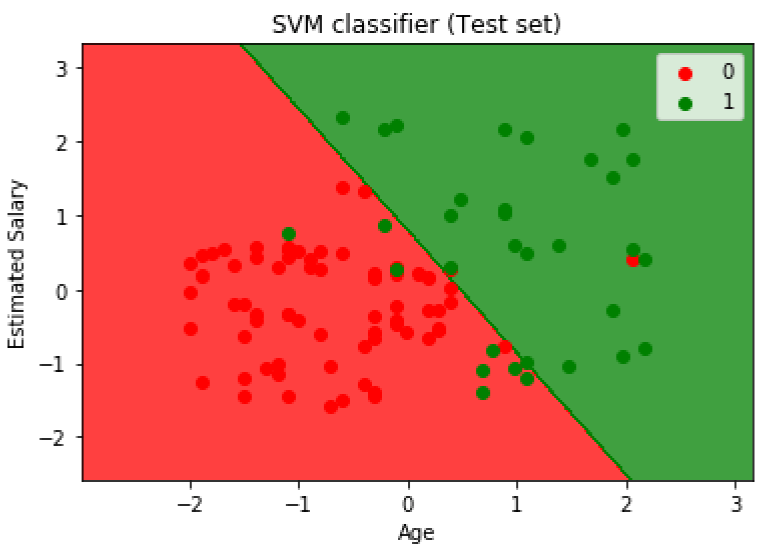
Visualizing the test set result:

1. #Visulaizing the test set result
2. from matplotlib.colors **import** ListedColormap
3. x_set, y_set = x_test, y_test
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, ste p =0.01),
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green')))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. **for** i, j in enumerate(nm.unique(y_set)):

```
11.     mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12.         c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('SVM classifier (Test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()
```

Output:

By executing the above code, we will get the output as:



- As we can see in the above output image, the SVM classifier has divided the users into two regions (Purchased or Not purchased).
- Users who purchased the SUV are in the red region with the red scatter points. And users who did not purchase the SUV are in the green region with green scatter points.
- The hyperplane has divided the two classes into Purchased and not purchased variable.