

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import metrics
```

```
In [2]: dataset = pd.read_csv('Social_Network_Ads.csv')
```

```
In [3]: dataset.head()
```

```
Out[3]:
```

	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

```
In [4]: X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

```
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 = 0.25,
                                                    random_state = 0)
```

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

```
In [7]: from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

```
Out[7]: SVC(random_state=0)
```

```
In [8]: y_pred = classifier.predict(X_test)
```

```
In [9]: from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report
accuracy_score(y_test, y_pred)
print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
print("Accuracy : ", accuracy_score(y_test, y_pred)*100)
print("Report : ", classification_report(y_test, y_pred))
```

Confusion Matrix: $\begin{bmatrix} 64 & 4 \\ 3 & 29 \end{bmatrix}$

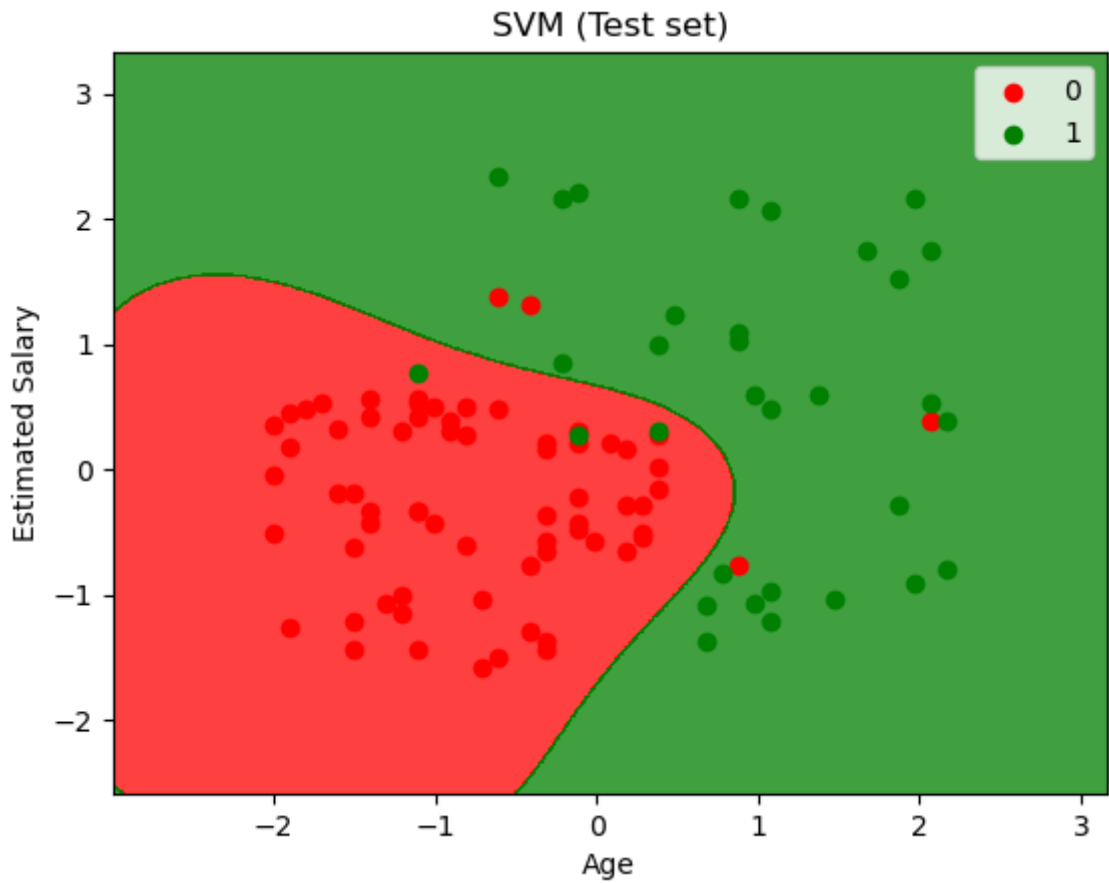
Accuracy : 93.0

Report :	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

```
In [10]: 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('SVM (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.



```
In [ ]:
```