svm

April 19, 2023

```
[35]: import numpy as np
      import matplotlib.pyplot as plt
      import pandas as pd
```

```
[36]: dataset=pd.read_csv("C:\\Users\\admin\\Downloads\\Social_Network_Ads.csv")
```

```
[37]: dataset.head()
```

```
[37]:    Age  EstimatedSalary  Purchased
      0   19            19000          0
      1   35            20000          0
      2   26            43000          0
      3   27            57000          0
      4   19            76000          0
```

```
[38]: dataset.shape
```

```
[38]: (400, 3)
```

```
[39]: #split dataset into dependent and independent part
      X=dataset.iloc[:,[0,1]].values
      y=dataset.iloc[:,2].values
```

```
[40]: #split x and y into training and testing 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)
```

```
[41]: #perform feature scaling
      from sklearn.preprocessing import StandardScaler
      sc=StandardScaler()
      X_train=sc.fit_transform(X_train)
      X_test=sc.transform(X_test)
```

```
[42]: #fi svm to the training set
      from sklearn.svm import SVC
      classifier1=SVC(kernel='rbf',random_state=0)
```

```
classifier1.fit(X_train,y_train)
```

```
[42]: SVC(random_state=0)
```

```
[43]: #predict the test set results
      y_pred1=classifier1.predict(X_test)
```

```
[44]: #confusion matrix
      from sklearn.metrics import confusion_matrix,accuracy_score
      cm=confusion_matrix(y_test,y_pred1)
      print(cm)
      accuracy_score(y_test,y_pred1)
```

```
[[64  4]
 [ 3 29]]
```

```
[44]: 0.93
```

```
[45]: #different kernel
      classifier2=SVC(kernel='linear',random_state=0)
      classifier2.fit(X_train,y_train)
```

```
[45]: SVC(kernel='linear', random_state=0)
```

```
[46]: y_pred2=classifier2.predict(X_test)
```

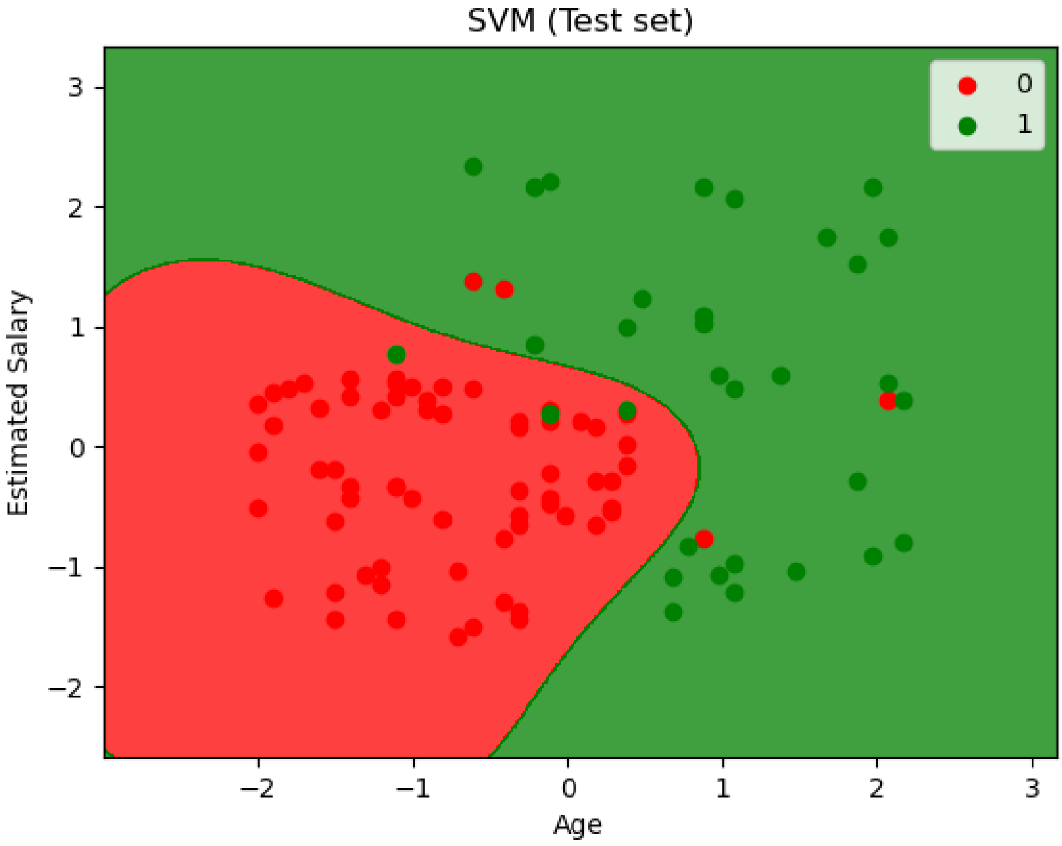
```
[47]: from sklearn.metrics import confusion_matrix,accuracy_score
      cm2=confusion_matrix(y_test,y_pred2)
      print(cm2)
      accuracy_score(y_test,y_pred2)
```

```
[[66  2]
 [ 8 24]]
```

```
[47]: 0.9
```

```
[48]: #visualize the test set results
      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, classifier1.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()
```

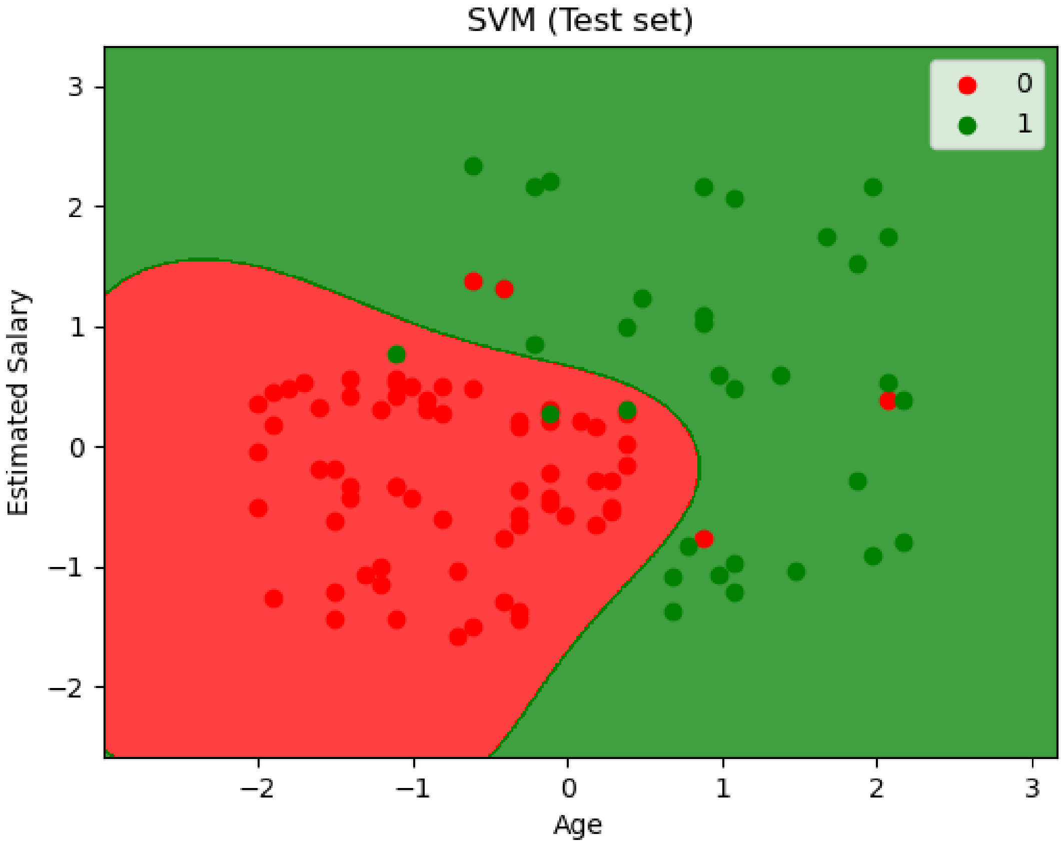
```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value 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.
```



[50]:

```
#visualize the test set results
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, classifier1.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.
```


SVM (Test set)
0
1
3
2
1
0
−1
−2
Estimated Salary
−2
−1
0
1
2
3
Age

```
[ ]:
```