svm

April 28, 2022

[ ]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[ ]:
```
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values   # only age and estimated salary selected
y = dataset.iloc[:, 4].values        # purchases
```

[ ]:
```
# Splitting the dataset into the Training set 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    : scaling values in range of 0 to 1
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

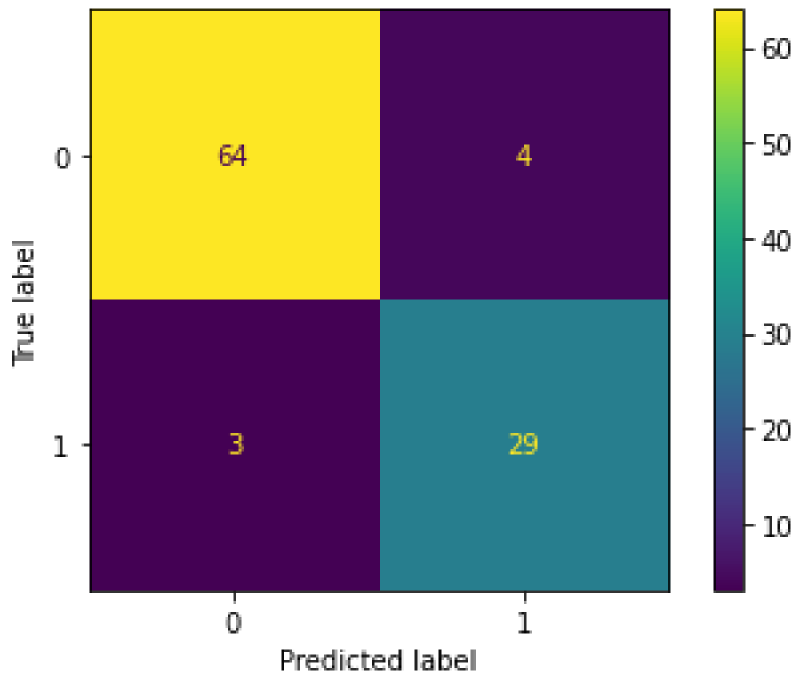
[ ]:
```
# Fitting classifier to the Training set
from sklearn.svm import SVC
classifier = SVC(random_state=0,kernel='rbf') # for non-linear model use this␣
 ↪parameter kernel='rbf'
classifier.fit(X_train, y_train)
```

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

[ ]:
```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

[ ]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=cm).plot()
```

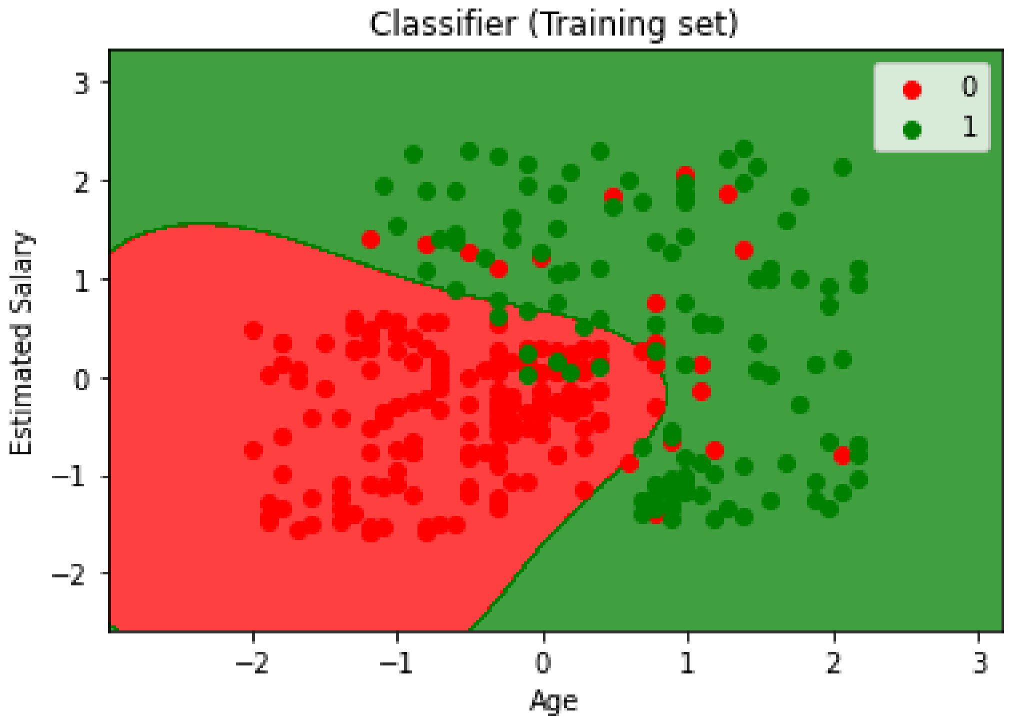
```
[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
     0x7fb4c1aed6f0>
```



```
[ ]: # Visualising the Training set results
     from matplotlib.colors import ListedColormap
     X_set, y_set = X_train, y_train
     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('Classifier (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.
```


Classifier (Training set)
Estimated Salary
Age