Grid Search

## Importing the libraries

In [1]:

import numpy as np

import matplotlib.pyplot as plt import pandas as pd

## Importing the dataset

In [ 2] :

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values y = dataset.iloc[:, -1].values

In [3]:

dataset.head()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Out[3]: |  | | | | |
|  | User ID | Gender | Age | EstimatedSalary | Purchased |
|  | 0 15624510 | Male | 19.0 | 19000.0 | 0 |
|  | 1 15810944 | Male | 35.0 | 20000.0 | 0 |
|  | 2 15668575 | Female | 26.0 | 43000.0 | 0 |
|  | 3 15603246 | Female | 27.0 | 57000.0 | 0 |
|  | 4 15804002 | Male | 19.0 | 76000.0 | 0 |

## Feature Scaling

In [ 5 ] :

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

X = sc.fit\_transform(X)

## Splitting the dataset into the Training set and Test set



In [6]:

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, ra

# Training the Kernel SVM model on the Training set

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)|

# Predicting the Test set results

In [8]:

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

In [9]:

from sklearn.metrlcs import confusion\_matrlX cm = confusion\_matrix(y\_test, y\_pred) print(cm)

[[64 4]

[ 3 29]]

Applying k-Fold Cross Validation

In [10]:

from sklearn.model\_selection import cross\_val\_score

accuracies = cross\_val\_score(estimator = classifier, X = X\_train, y = y\_train, print(”Accuracy: {:.2f} %”.format(accuracies.mean()\*100))

print(”Standard Deviation: {:.2f} %”.format(accuracies.std()\*100))

Accuracy: 90.00 %

Standard Deviation: 6.83 %

## Applying Grid Search to find the best model and the best parameters

In [ 14] :

from sklearn.model\_selection import GridSearchCV

parameters = [{'C': [1,], 'kernel' : ['rbf'], 'gamma' : [ 0.7]}] grid\_search = GridSearchCV(estimator = classifier,

param\_grid = parameters, scoring = 'accuracy',

cv 10,

n\_jobs = -1) grid\_search = grid\_search.fit(X\_train, y\_train) best\_accuracy = grid\_search.best\_score\_ best\_parameters = grid\_search.best\_params\_

print(”Best Accuracy: {:.2f} %".format(best\_accuracy\*100)) print(”Best Parameters: ”, best\_parameters)

Best Accuracy: 91.00 %

Best Parameters: {'C': 1, 'gamma' : 0.7, 'kernel' : 'rbf'}

**Visualising the Training set results**

In [16]:

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[:, np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:,

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T). alpha = 0.75, cmap = ListedColormap(('yellow', '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],

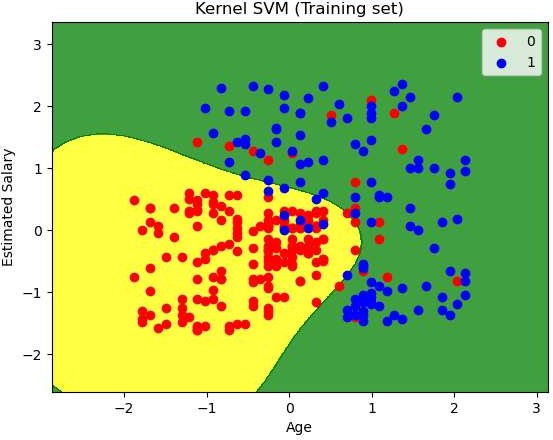
c = ListedColormap(('red', 'blue'))(i), label = j) plt.title('Kernel SVM (Training set)')

plt.xlabel('Age') plt.ylabel('Estimated Salary') plt.legend()

plt.show()

C:\Users\LENOVO\AppData\Local\Temp\ipykernel\_9940\3367766136.py:10: UserWarni ng: \*c\* argument looks like a single numeric RGB or RGBA sequence, which shou ld be avoided as value-mapplng will have precedence in case its length matche s with \*x\* & \*y\*. Please use the \*color\* keyword-argument or provide a 2D ar ray with a single row if you intend to specify the same RGB or RGBA value for all points.

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],



# Visualising the Test set results

In [11]:

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[:, np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:,

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T). 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('Kernel 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 b e avoided as value-mapping will have precedence in case its length matches Wi th 'x' & 'y'. Please use a 2-D array with a single row if you really want to specify the same RGB or RGBA value for all points.

'c' argument looks like a single numeric RGB or RGBA sequence, which should b e avoided as value-mapping will have precedence in case its length matches wi th 'x' & 'y'. Please use a 2-D array with a single row if you really want to specify the same RGB or RGBA value for all points.

