k-Fold Cross Validation

## 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

## Feature Scaling

In [3]:

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

X = sc.fit\_transform(X)

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



In [4]:

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 [5]:

from sklearn.svm import SVC

classifier = SVC(kernel = 'rbf', random\_state = 0) classifier.fit(X\_train, y\_train)

Out[5]:



.SVC(random\_state=0)|

Predicting the Test set results

In [6]:

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

Making the Confusion Matrix

In [7]:

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

[[64 4]

[ 3 29]]

Applying k-Fold Cross Validation

In [8]:

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 %

# Visualising the Training set results

In [12] :

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],

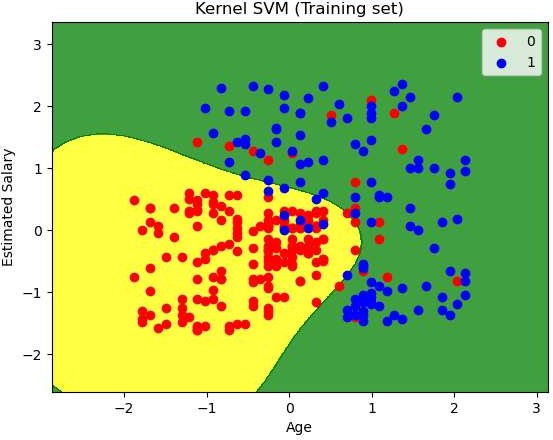
c = ListedColormap(('red', 'blue'))(i), label = j) plt.title('Kernel SVM (Training set)')

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

plt.show()

C:\Users\LENOVO\AppData\Local\Temp\ipykernel\_10276\3367766136.py:10: UserWarn ing: \*c\* argument looks like a single numeric RGB or RGBA sequence, which sho uld be avoided as value-mapping will have precedence in case its length match es with \*x\* & \*y\*. Please use the \*color\* keyword-argument or provide a 2D a rray with a single row if you intend to specify the same RGB or RGBA value fo r 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(('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],

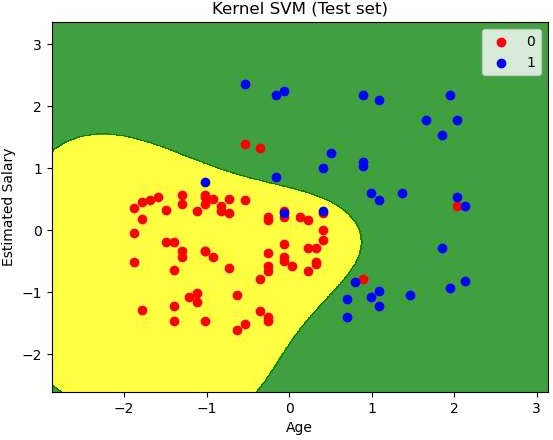
c = ListedColormap(('red', 'blue'))(i), label = j) plt.title('Kernel SVM (Test set)')

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

plt.show()

C:\Users\LENOVO\AppData\Local\Temp\ipykernel\_10276\2136200641.py:10: UserWarn ing: \*c\* argument looks like a single numeric RGB or RGBA sequence, which sho uld be avoided as value-mapping will have precedence in case its length match es with \*x\* & \*y\*. Please use the \*color\* keyword-argument or provide a 2D a rray with a single row if you intend to specify the same RGB or RGBA value fo r all points.

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



In [ ]: