

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
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))
```

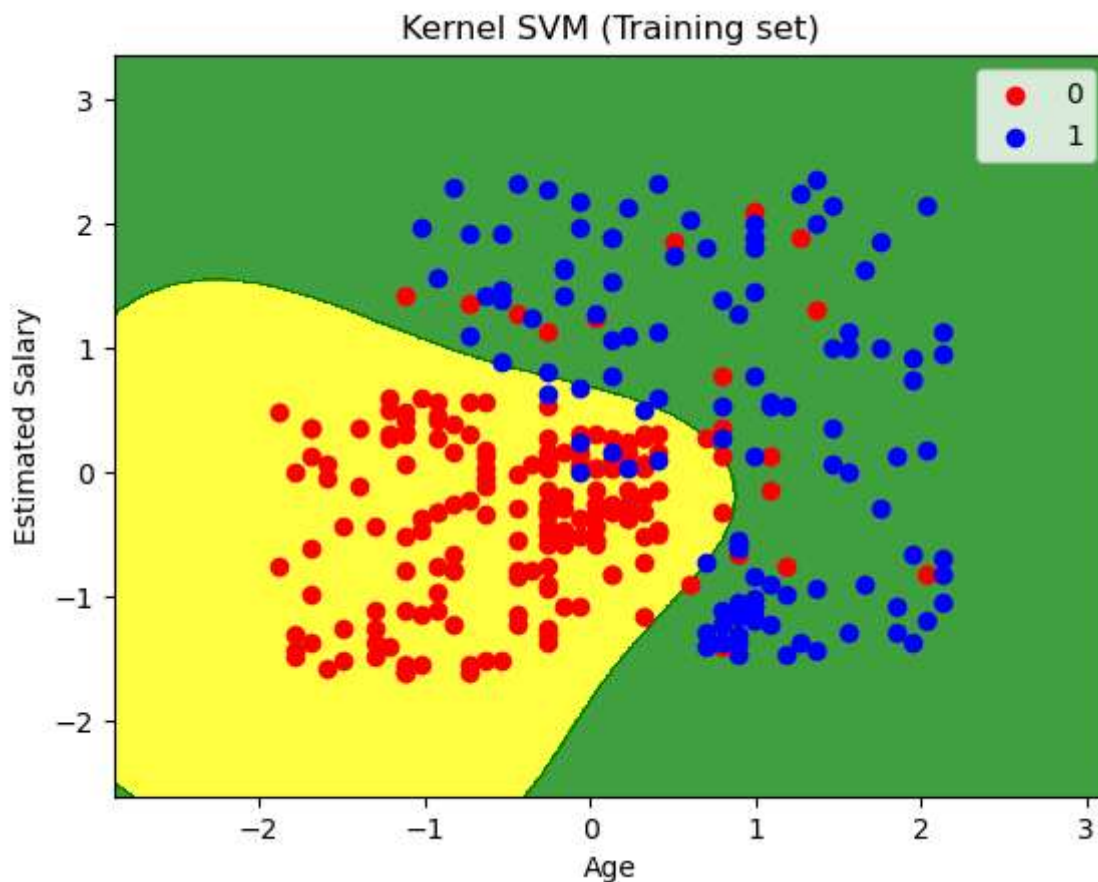
```
Accuracy: 90.00 %  
Standard Deviation: 6.83 %
```

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: UserWarning: *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.

```
plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```

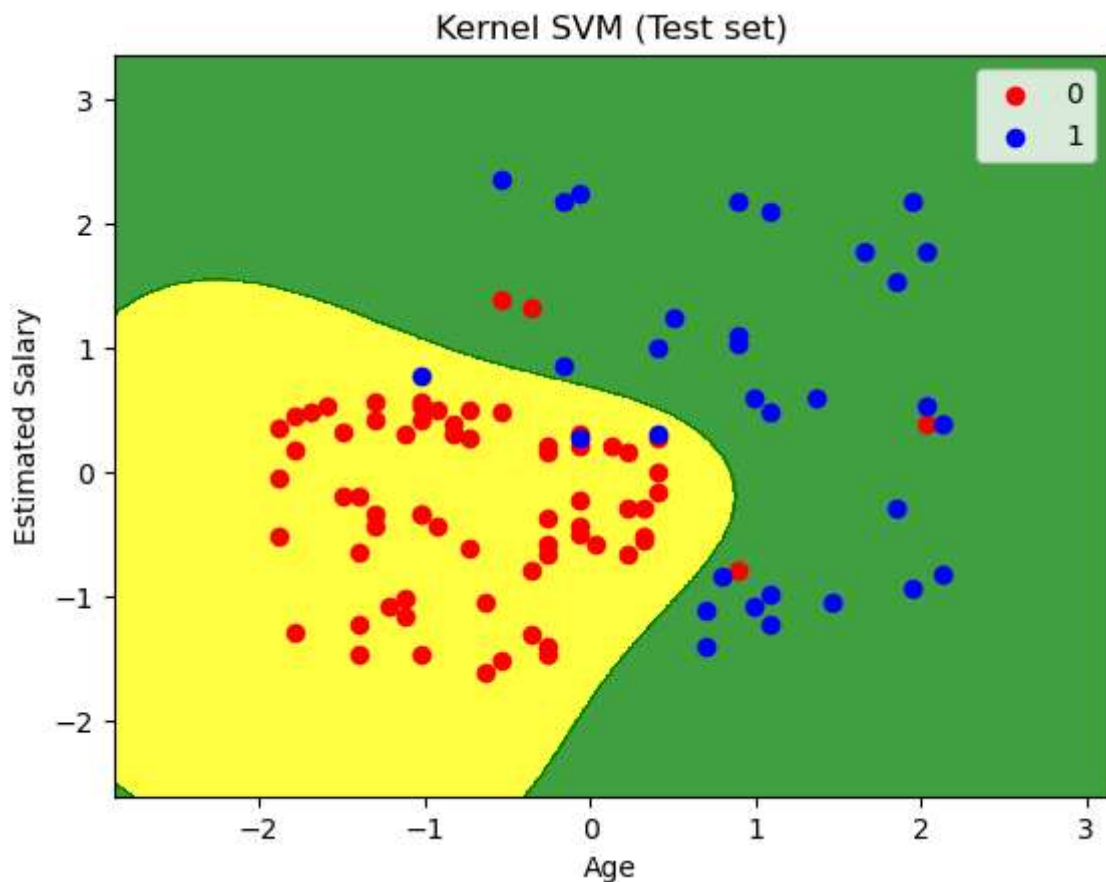


Visualising the Test set results

```
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: UserWarning: *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.

```
plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```



In []: