Cross Validation

ML MODEL TUNNING _ RANDOM SEARCH CV & GRID SEARCH CV

In [234…

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [235…

```
dataset = pd.read_csv("E:\Data Science & AI\Dataset files\Social_Network_Ads.csv
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values
```

In [236…

```
## Feature Scaling

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [237…

```
## 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, rand
```

In [238…

```
## Training the Kernel SVM model on the Training set

from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[238…

```
▾ SVC
SVC(random_state=0)
```

In [239…

```
## Predicting the Test set results

y_pred = classifier.predict(X_test)
```

In [240…

```
## Making the Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

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

In [241…

```
## Applying k-Fold Cross Validation

from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, c
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

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

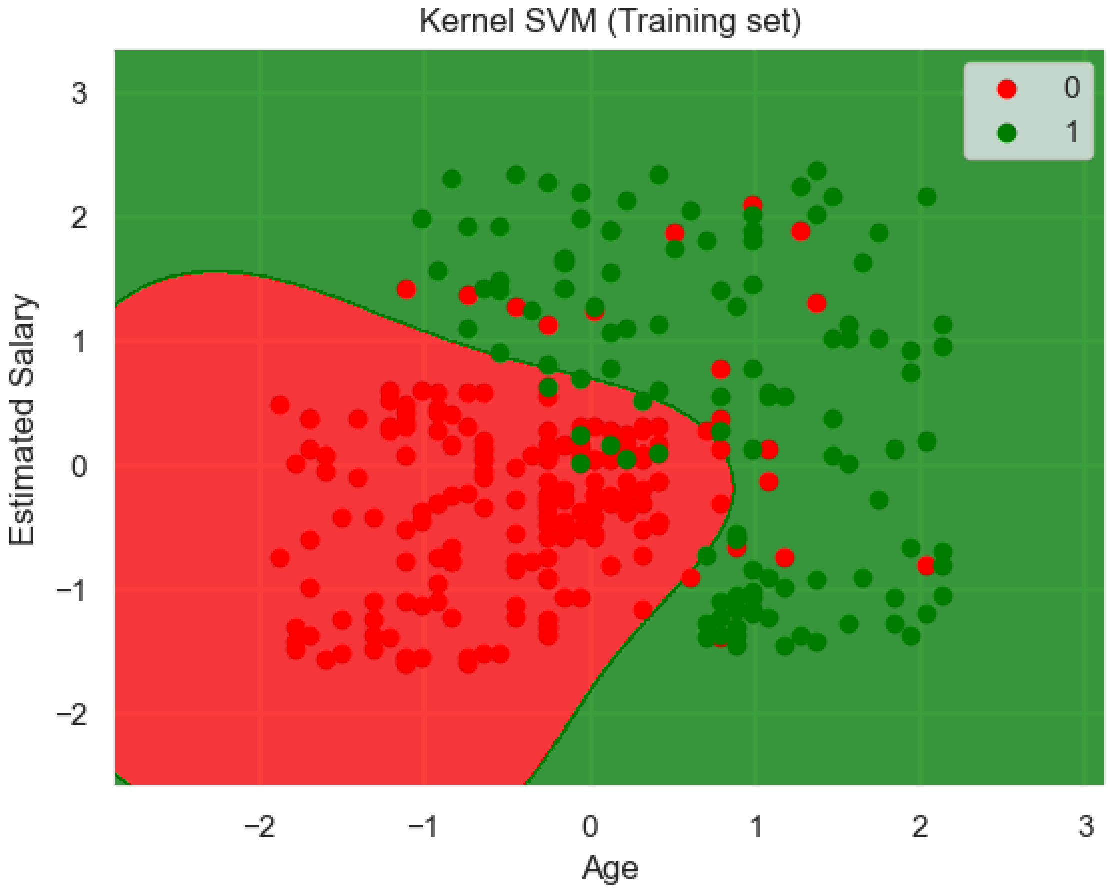
In [242…

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

from sklearn.model_selection import GridSearchCV
parameters = [{'C': [1, 10, 100, 1000], 'kernel': ['linear']},
              {'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': [0.1, 0.2, 0
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'}
```

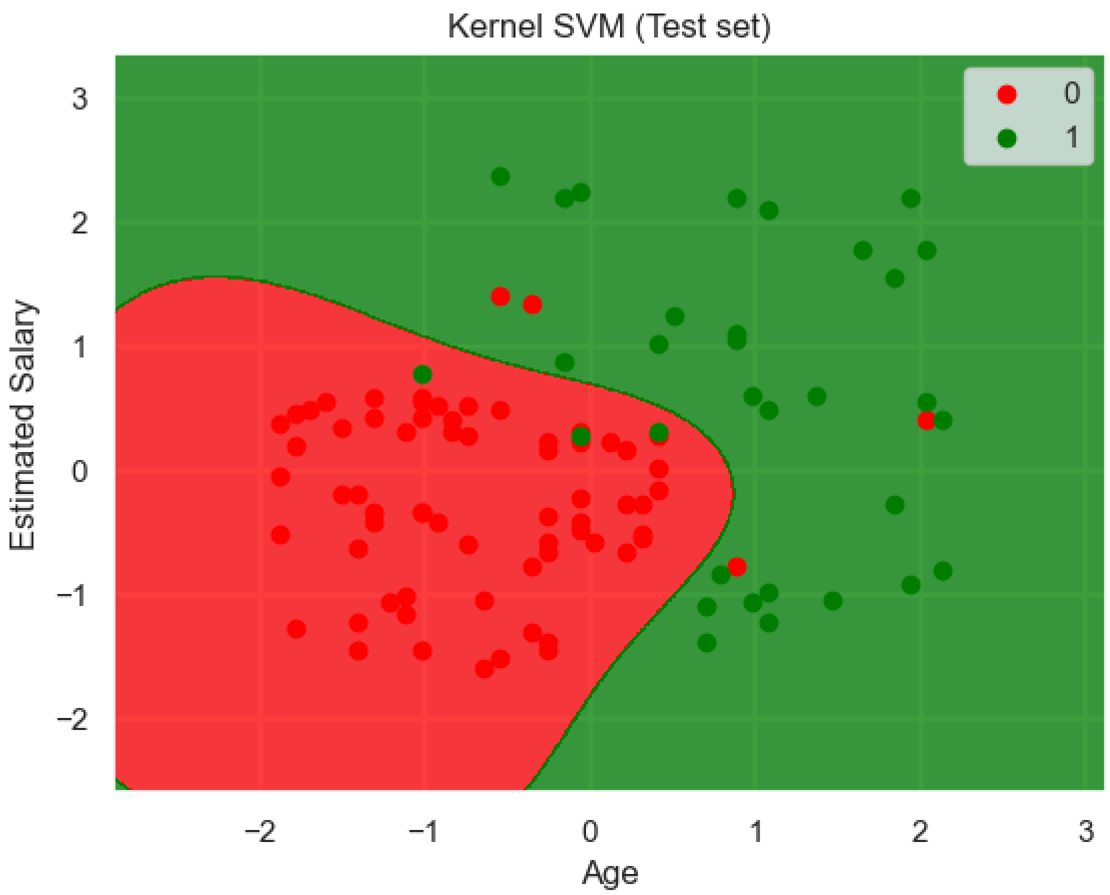
In [243…

```
## 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]
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1]
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).re
             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 (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



In [244…

```
## Visualising 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]
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1]
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).re
             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()
```



In [ ]:

K-FOLD CROSS VALIDATION CODE_ MODEL SELECTION

In [245…

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [246…

```
dataset = pd.read_csv(r"E:\Data Science & AI\Dataset files\Social_Network_Ads.cs
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values
```

In [247…

```
## Feature Scaling

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [248…

```
## 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, rand
```

In [249…

```
## Training the Kernel SVM model on the Training set

from sklearn.svm import SVC
```

```
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[249…

```
▾ SVC
SVC(random_state=0)
```

In [250…

```
## Predicting the Test set results

y_pred = classifier.predict(X_test)
```

In [251…

```
## Making the Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

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

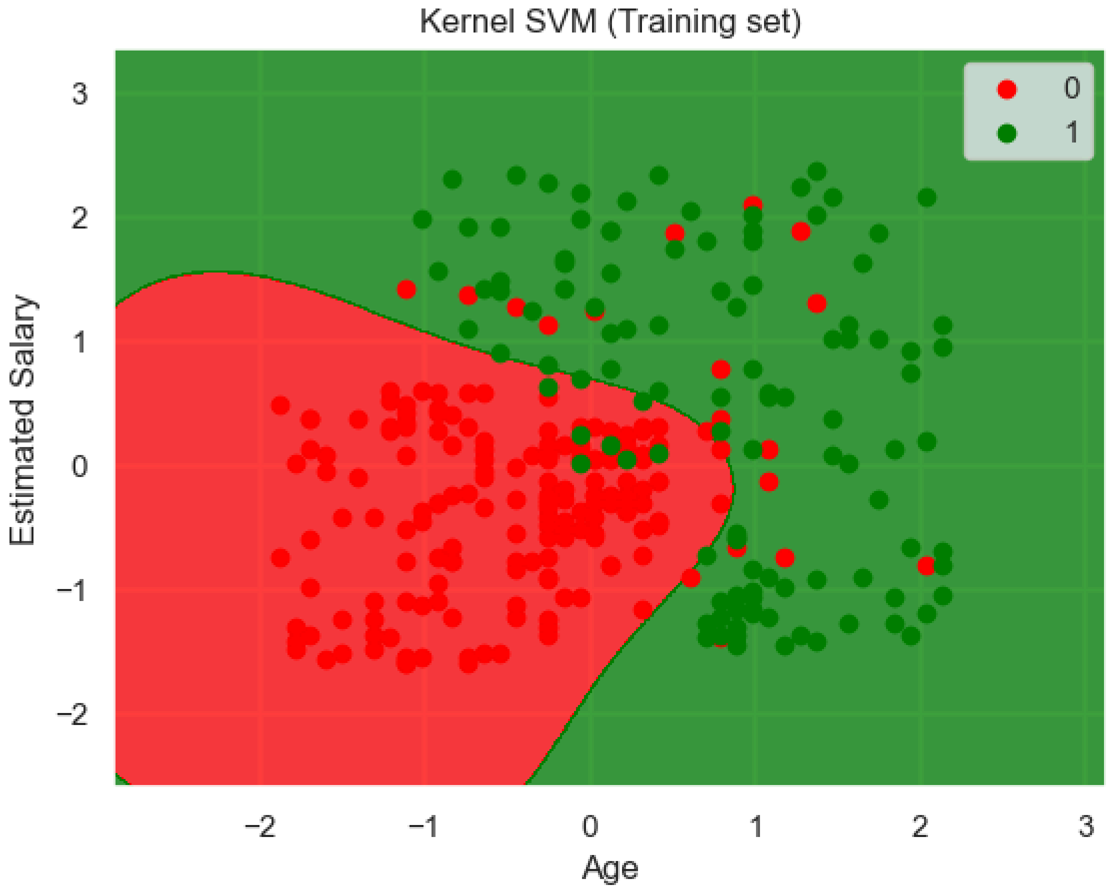
In [252…

```
## Applying k-Fold Cross Validation

from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, c
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

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

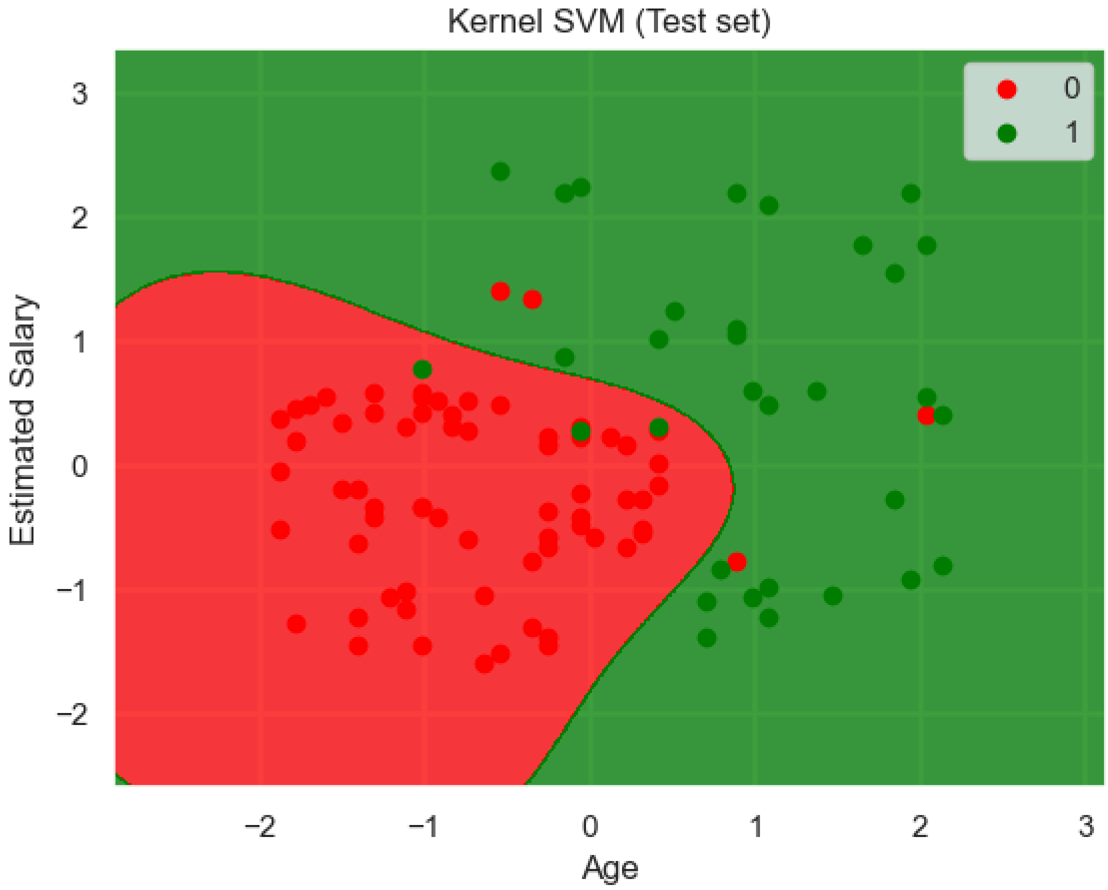
In [253…

```
## 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]
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1]
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).re
             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 (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



In [254…

```
## Visualising 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]
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1]
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).re
             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()
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```