SVM_Assignment14

Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(color_codes=True)
from sklearn.preprocessing import LabelEncoder
```

Importing the dataset

In [2]:
```
dataset = pd.read_csv("Social_Network_Ads.csv")
```

In [3]:
```
dataset.head()
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [4]:
```
dataset.shape
```

Out[4]:
```
(400, 5)
```

In [5]:
```
dataset.columns
```

Out[5]:
```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

In [6]:
```
dataset.isna().any()
```

Out[6]:
```
User ID            False
Gender             False
Age                False
EstimatedSalary    False
Purchased          False
dtype: bool
```

In [7]:
```
dataset.isna().sum()
```

Out[7]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [8]:
```
dataset["Gender"].unique()
```

Out[8]:
```
array(['Male', 'Female'], dtype=object)
```

In [9]:
```
dataset["Gender"].replace({'Male':'0', 'Female':'1'},inplace=True)
```

In [10]:
```
dataset
```

Out[10]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	0	19	19000	0
1	15810944	0	35	20000	0
2	15668575	1	26	43000	0
3	15603246	1	27	57000	0
4	15804002	0	19	76000	0
...	...	...	...	...	...
395	15691863	1	46	41000	1
396	15706071	0	51	23000	1
397	15654296	1	50	20000	1
398	15755018	0	36	33000	0
399	15594041	1	49	36000	1

400 rows × 5 columns

In [21]:
```
x=dataset.iloc[:,0:4]
```

In [22]:
```
y=dataset.iloc[:,4]
```

Splitting the dataset into the Training set and Test set

In [23]:
```
from sklearn.model_selection import train_test_split
```

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [26]:
```
len(X_train)
```

Out[26]:
```
320
```

In [27]:
```
len(X_test)
```

Out[27]:
```
80
```

Feature Scaling

In [28]:
```
from sklearn.preprocessing import StandardScaler
```

In [29]:
```
sc_X = StandardScaler()
```

In [30]:
```
X_train = sc_X.fit_transform(X_train)
```

In [31]:
```
X_test = sc_X.transform(X_test)
```

Fitting SVM to the Training set

In [32]:
```
from sklearn.svm import SVC
```

In [33]:
```
model=SVC(kernel='linear')
```

In [34]:
```
model.fit(X_train,y_train)
```

Out[34]:
```
SVC(kernel='linear')
```

Predicting the Test set results

In [35]:
```
model.predict(X_test)
```

Out[35]:
```
array([0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [36]:
```
X_test
```

Out[36]:
```
array([[ 0.24944539,  1.01892912, -0.31388835, -1.3079089 ],
       [-1.08709592,  1.01892912, -0.31388835,  0.01947796],
       [-1.3221496 ,  1.01892912, -1.06981891,  0.39460903],
       [ 1.66898307,  1.01892912, -0.21939703,  0.1349029 ],
       [ 1.2342395 ,  1.01892912,  1.10348145,  0.51003397],
       [ 1.2196318 ,  1.01892912,  2.04839465,  1.0871587 ],
       [ 1.32416858,  1.01892912, -0.88083627,  0.36575279],
       [-0.76250532,  1.01892912, -0.31388835, -0.586503  ],
       [-0.16609669, -0.98142253, -0.21939703,  2.1259832 ],
       [-0.53431431,  1.01892912,  1.10348145, -1.45219009],
       [ 0.39910776, -0.98142253, -1.16431023,  0.4523215 ],
       [-1.02743265,  1.01892912, -0.21939703, -0.586503  ],
       [ 0.75595295,  1.01892912, -1.35329288,  0.53889021],
       [ 1.43944754,  1.01892912, -0.21939703,  1.60657095],
       [-1.09279614,  1.01892912, -1.4477842 , -1.2213402 ],
       [-1.34346592, -0.98142253,  1.00899013,  0.53889021],
       [-0.99947775,  1.01892912,  0.91449881, -1.19248396],
       [ 0.6807016 , -0.98142253, -0.12490571,  0.0483342 ],
       [-1.37821347,  1.01892912,  1.00899013, -0.99049031],
       [-0.91352622, -0.98142253,  0.06407693,  0.01947796],
       [ 0.45393915, -0.98142253, -1.06981891,  0.56774645],
       [ 1.61710683, -0.98142253, -1.82574948,  0.42346527],
       [-0.45162606,  1.01892912, -0.12490571,  0.01947796],
       [-0.52273179,  1.01892912, -1.16431023, -1.53875879],
       [-1.38277925, -0.98142253,  2.04839465,  0.91402128],
       [ 0.46704826,  1.01892912, -0.21939703, -0.21137193],
       [ 0.10045528, -0.98142253, -0.21939703,  0.82745257],
       [-1.36811553,  1.01892912,  0.06407693,  0.1349029 ],
       [-1.55808563, -0.98142253, -0.69185363,  0.10604667],
       [-1.15273952,  1.01892912, -0.69185363, -1.59647127],
       [ 1.05369508,  1.01892912, -0.31388835,  0.0483342 ],
       [-0.64922913,  1.01892912,  0.72551617, -1.10591525],
       [ 1.2616622 ,  1.01892912, -0.50287099,  1.23143988],
       [ 0.00521813,  1.01892912,  1.76492069, -1.27905267],
       [-1.24827078, -0.98142253, -0.50287099, -0.78849665],
       [-1.05478532, -0.98142253, -0.31388835, -0.32679687],
       [-0.03775063, -0.98142253, -0.97532759,  0.53889021],
       [ 0.02729074,  1.01892912, -1.35329288, -1.10591525],
       [ 1.4057504 , -0.98142253,  1.00899013, -1.2213402 ],
       [-1.51120934,  1.01892912, -0.59736231,  0.4523215 ],
       [ 1.50719196,  1.01892912, -0.97532759, -0.44222182],
       [ 1.71384257, -0.98142253, -0.69185363, -0.12480322],
       [-1.63517169, -0.98142253, -0.31388835,  0.07719043],
       [-0.78456392,  1.01892912, -0.78634495,  1.31800859],
       [ 0.58423197,  1.01892912, -0.12490571,  0.65431516],
       [-1.15978427,  1.01892912,  0.25305957,  0.0483342 ],
       [ 0.69687791, -0.98142253,  0.63102485, -1.10591525],
       [ 0.44755266,  1.01892912, -1.73125816,  0.33689656],
       [ 0.17740129, -0.98142253,  0.72551617, -1.36562138],
       [-1.36158898,  1.01892912, -1.73125816, -1.36562138],
       [-1.24929318,  1.01892912, -1.06981891, -0.38450935],
       [ 0.64427339,  1.01892912, -0.78634495,  0.25032785],
       [-1.0229509 ,  1.01892912, -0.03041439,  1.20258364],
       [-0.2417542 , -0.98142253, -0.69185363, -0.35565311],
       [ 0.2210283 , -0.98142253, -0.31388835, -1.39447761],
       [ 0.7519894 , -0.98142253, -1.73125816,  0.4523215 ],
       [-0.28963888,  1.01892912, -1.16431023,  0.0483342 ],
       [-0.4137973 , -0.98142253, -1.82574948, -0.00937828],
       [-1.47688195,  1.01892912,  0.25305957,  0.0483342 ],
       [-0.00894139,  1.01892912,  0.15856825,  0.0483342 ],
       [-1.62781882, -0.98142253,  0.25305957, -0.73078418],
       [ 0.41160064, -0.98142253, -0.03041439, -0.2690844 ],
       [ 0.44417734,  1.01892912,  1.10348145, -0.75964042],
       [-0.69413065, -0.98142253, -1.54227552, -1.50990256],
       [-1.55690917, -0.98142253, -0.12490571,  0.10604667],
       [ 0.76036467, -0.98142253, -1.9202408 , -0.52879053],
       [ 0.61318126, -0.98142253, -0.03041439, -0.32679687],
       [ 0.1756366 , -0.98142253, -0.59736231, -1.50990256],
       [ 0.76465034, -0.98142253,  0.72551617,  1.34686482],
       [ 1.00800926,  1.01892912, -1.06981891, -1.53875879],
       [ 0.03500775, -0.98142253, -1.35329288, -0.44222182],
       [-1.64143213,  1.01892912,  0.91449881,  1.75085213],
       [ 1.4875563 ,  1.01892912, -0.78634495, -0.24022817],
       [-0.67436894,  1.01892912, -0.40837967,  0.0483342 ],
       [ 0.51270607,  1.01892912, -1.06981891,  1.92398955],
       [ 1.52955869, -0.98142253, -1.63676684,  0.0483342 ],
       [ 1.42076425,  1.01892912, -0.12490571,  0.27918409],
       [ 0.9158253 , -0.98142253, -0.31388835, -0.93277783],
       [-0.16378579,  1.01892912,  0.44204221,  1.8085646 ],
       [ 0.30063535,  1.01892912, -0.12490571,  0.22147161]])
```

In [37]:
```
model.score(X_test,y_test)
```

Out[37]:
```
0.8
```

In [45]:
```
y_pred = model.predict(X_test)
y_pred
```

Out[45]:
```
array([0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

Making the Confusion Matrix

In [46]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

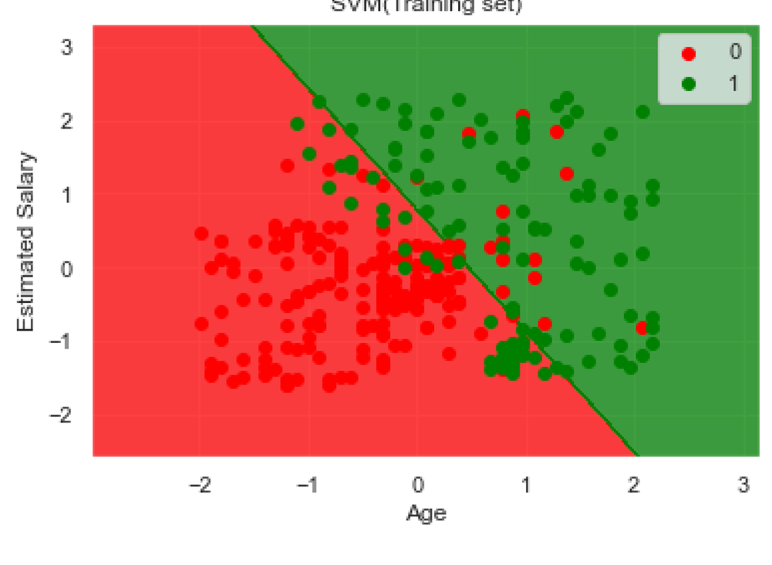
In [47]:
```
print(cm)
```

```
[[54  5]
 [11 10]]
```

Visualising the Training set results

In [56]:
```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:,4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.svm import SVC
classifier = SVC(kernel ='linear', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
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('SVM(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 arra
y 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 arra
y with a single row if you intend to specify the same RGB or RGBA value for all points.
```



Visualising the Test set results

In [57]:
```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:,4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.svm import SVC
classifier = SVC(kernel ='linear', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
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].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('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 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 arra
y 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 arra
y with a single row if you intend to specify the same RGB or RGBA value for all points.
```