In [52]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
```

In [53]:

```
data=pd.read_csv("smartphone_activity_dataset.csv")
data.head(10)
```

Out[53]:

	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	feature_9	feature_10
0	0.289	-0.02030	-0.133	-0.995	-0.983	-0.914	-0.995	-0.983	-0.924	-0.935
1	0.278	-0.01640	-0.124	-0.998	-0.975	-0.960	-0.999	-0.975	-0.958	-0.943
2	0.280	-0.01950	-0.113	-0.995	-0.967	-0.979	-0.997	-0.964	-0.977	-0.939
3	0.279	-0.02620	-0.123	-0.996	-0.983	-0.991	-0.997	-0.983	-0.989	-0.939
4	0.277	-0.01660	-0.115	-0.998	-0.981	-0.990	-0.998	-0.980	-0.990	-0.942
5	0.277	-0.01010	-0.105	-0.997	-0.990	-0.995	-0.998	-0.990	-0.996	-0.942
6	0.279	-0.01960	-0.110	-0.997	-0.967	-0.983	-0.997	-0.966	-0.983	-0.941
7	0.277	-0.03050	-0.125	-0.997	-0.967	-0.982	-0.996	-0.966	-0.983	-0.941
8	0.277	-0.02180	-0.121	-0.997	-0.961	-0.984	-0.998	-0.957	-0.984	-0.941
9	0.281	-0.00996	-0.106	-0.995	-0.973	-0.986	-0.995	-0.974	-0.986	-0.940

10 rows × 562 columns

In [64]:

```
real_x=data.iloc[:,1:3].values
real_y=data.iloc[:,561].values
```

In [65]:

```

training_x,test_x,training_y,test_y=train_test_split(real_x,real_y,test_size=0.25,random_state=0)
```

In [66]:

```
# s_c=StandardScaler()
# training_x=s_c.fit_transform(training_x)
# test_x=s_c.fit_transform(test_x)
```

In [67]:

```
cls_svc=SVC(kernel="linear",random_state=0)
cls_svc.fit(training_x,training_y)
```

Out[67]:

```
SVC(kernel='linear', random_state=0)
```

In [68]:

```
y_pred=cls_svc.predict(test_x)
y_pred
```

Out[68]:

```
array([6, 6, 6, ..., 6, 6, 6], dtype=int64)
```

In [69]:

```
test_y
```

Out[69]:

```
array([4, 3, 6, ..., 4, 4, 5], dtype=int64)
```

In [70]:

```
c_m=confusion_matrix(test_y,y_pred)
c_m
```

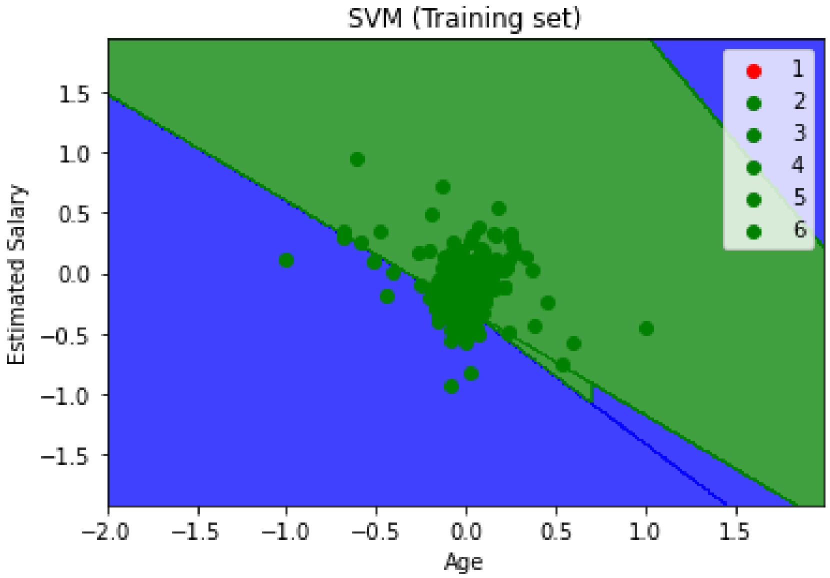
Out[70]:

```
array([[  0,   2,   0,   0,   0, 405],
       [  0,  13,   0,   0,   0, 379],
       [  0,   2,   0,   0,   0, 344],
       [  0,   4,   0,   0,   0, 461],
       [  0,   1,   0,   0,   0, 488],
       [  0,   6,   0,   0,   1, 469]], dtype=int64)
```

In [71]:

```
from matplotlib.colors import ListedColormap
X_set, y_set = training_x,training_y
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, cls_svc.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue', '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()
```

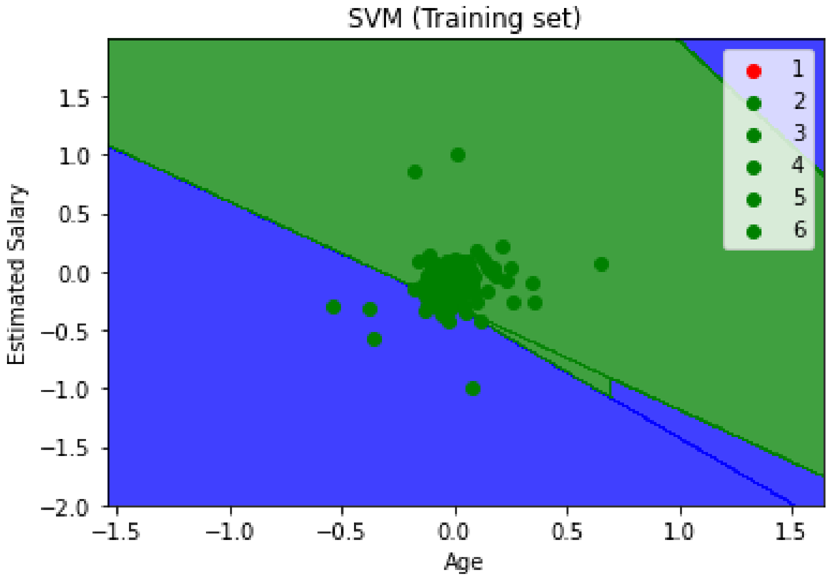
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly want to specify the same RGB or RGBA value for all points.
```


SVM (Training set)
1
2
3
4
5
6
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Estimated Salary
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Age

In [73]:

```
from matplotlib.colors import ListedColormap
X_set, y_set = test_x,test_y
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, cls_svc.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('blue', '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 wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly 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 be avoided as value-mapping wil
ave precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
lly want to specify the same RGB or RGBA value for all points.
```


SVM (Training set)
Estimated Salary
Age
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
● 1
● 2
● 3
● 4
● 5
● 6