Omkar Gurav
BE IT
8048

Rajdhani
DATE / /

Assignment 3

- Title :- Implement SVM for performing classification & find its accuracy on the given data. (Using Python)

- Theory:-

1. What is Support Vector Machine?

→ ① Support vector machines are the supervised machine algorithm used in classification or regression problems.
② It is useful as it transforms users data & with the platform of these transformations, it looks for optimal boundary among possible outputs.
③ Suppose, we consider three hyperplanes (A, B & C) & all three are segregating the classes too.
④ Here, with the help of maximizing the distances in between the nearest data point of hyperplane, a better hyperplane can be ~~shown~~ chosen.
⑤ The distance between data point & hyperplane is called as margin. Margin is ratio of distance of closest examples from the decision line to hyperplane.

Margin = m / |W|

Where m is distance in between nearest training instances of decision boundary & W is weight vector

Rajdhani
DATE / /

2. Explain Support Vectors.

→ ① Support vectors are the hyperplane used to maximize the margin in between two classes.
② Support vectors are data points that are closer to the hyperplane & influence the position & orientation of the hyperplane.
③ By using these support vectors, we maximize the margin of the classifiers.
④ If we delete the support vectors, then it will change the position of the hyperplane.
⑤ It is hyperplane & hyperplanes are the decision boundaries that help classify the data points.
⑥ Data points falling on either side of the hyperplane can be attributed to different classes.

3. Differentiate Hard margin & soft margin SVM.

→ ~~① In 1992,~~
① In 1992, Baseir et al in COL gave hard margin. If data can be easily separable in linear manner without any error then hard margin SVM works efficiently.
② If errors found it is due to margin are smaller or hard margin SVM fails.
③ In 1995, vapni et al gave the extended version of har margin i.e. soft margin.
④ If errors found it can be resolved with the help of slack variables.
⑤ One of the best features of soft margin SVM is the

slack variables allow the violation of the margin of these violated samples are treated as loss term.

⑥ Soft margin SVM has slack variables. It can be added which further allow this classification of difficulty.

4. Explain in brief gamma & regularization parameters.

→ ① Gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' & high values meaning 'close'.

② In other words, with low gamma points far away from plausible ~~se~~ separation line.

③ Where as high gamma means the points close to plausible line are considered.

Regularization Parameters:-

① It also terms as C ~~parameters~~ in python's sklearn library.

② It tells us the svm optimization how much we want to avoid misclassifying each training example.

③ For large values of C, the optimization will choose a smaller margin ~~b~~ hyperline. If that hyperline does a better job of getting all training points classified correctly.

④ Conversly, a very small value of C will cause the optimizer to look for a larger margin separating hyperplane, even if that hyperplane classifies more

Rajdhani
DATE / /

points.

- Conclusion:- SVM is implementated using python.

SVM.ipynb

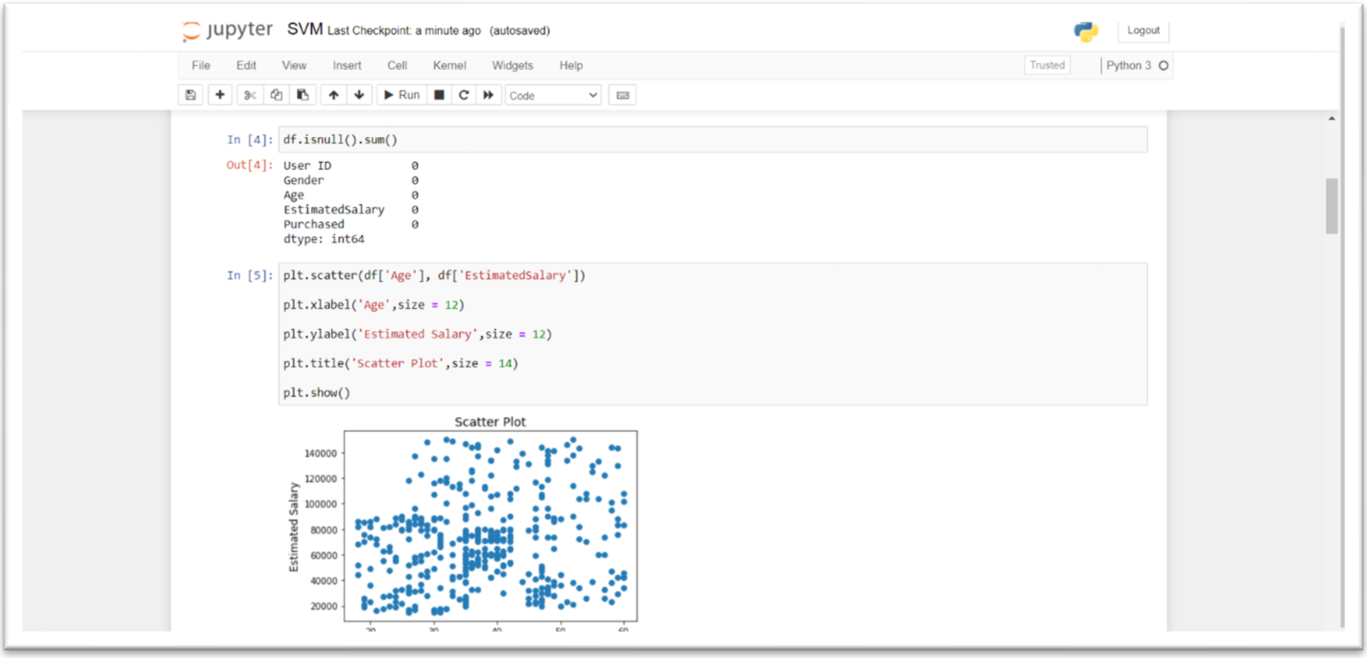
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from mlxtend.plotting import plot_decision_regions

df = pd.read_csv('SVM_DataSet_1.csv')

df

df.isnull().sum()

plt.scatter(df['Age'], df['EstimatedSalary'])

plt.xlabel('Age',size = 12)

plt.ylabel('Estimated Salary',size = 12)

plt.title('Scatter Plot',size = 14)

plt.show()

x = df[['Age','EstimatedSalary']].values

y = df['Purchased'].values
```

```
#splitting  data into training and testing set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.4, random_state = 42)

# Linear kernel
linear = svm.SVC(kernel = 'linear', C = 1)
linear.fit(x_train, y_train)

y_pred = linear.predict(x_test)
y_pred

y_test

confusion_matrix(y_test, y_pred)

accuracy_score(y_test, y_pred)

print(classification_report(y_test, y_pred))

plot_decision_regions(x_train, y_train, clf = linear, legend = 2)

plt.xlabel('Age',size = 12)

plt.ylabel('EstimatedSalary',size = 12)

plt.title('SVM Linear',size = 14)

plt.show()

# rbf kernel
```

```
rbf = svm.SVC(kernel = 'rbf', C = 1e4, gamma = 0.0001)
rbf.fit(x_train, y_train)

y_pred = rbf.predict(x_test)
y_pred

y_test

confusion_matrix(y_test, y_pred)

accuracy_score(y_test, y_pred)

print(classification_report(y_test, y_pred))

plot_decision_regions(x_train, y_train, clf = rbf, legend = 2)

plt.xlabel('Age')

plt.ylabel('EstimatedSalary')

plt.title('SVM Rbf')

plt.show()

# poly kernel
polynomial = svm.SVC(kernel = 'poly', degree = 3, C = 0.01, gamma = 10, max_iter = 1e5)
polynomial.fit(x_train, y_train)

y_pred = polynomial.predict(x_test)
```

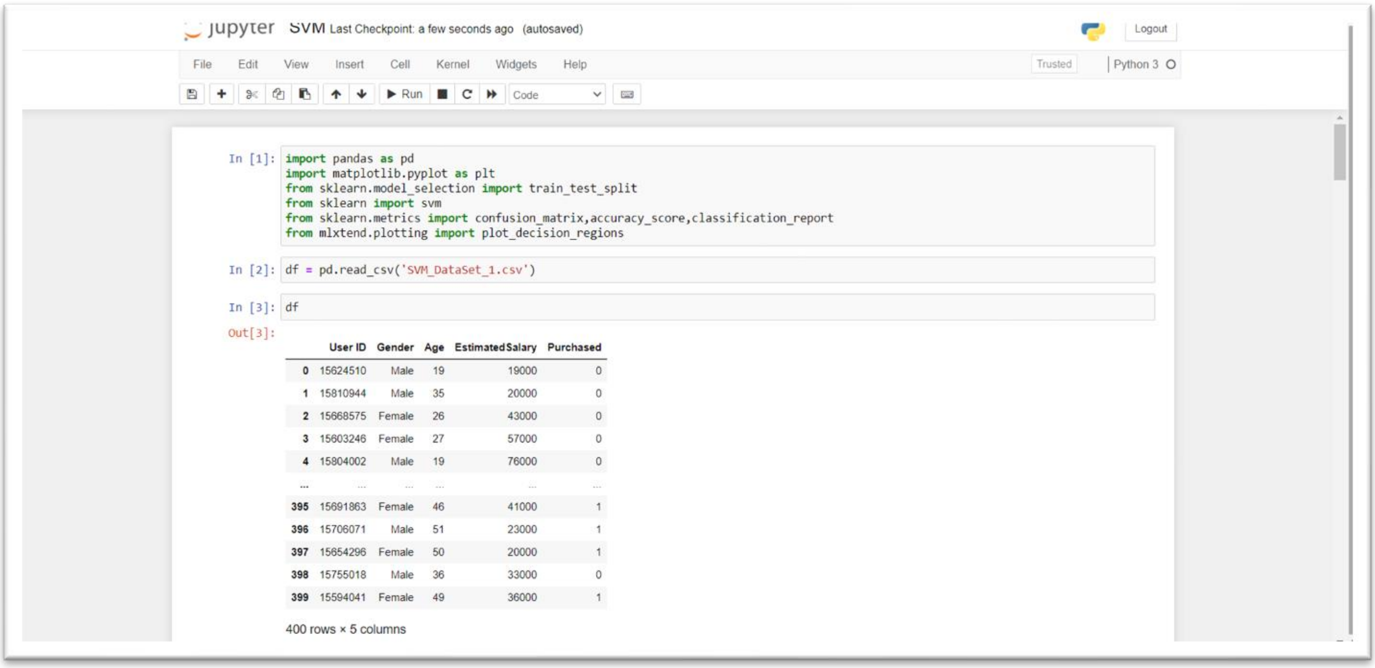
```
y_pred

y_test

confusion_matrix(y_test, y_pred)

accuracy_score(y_test, y_pred)

print(classification_report(y_test, y_pred))

plot_decision_regions(x_train, y_train, clf = polynomial, legend = 2)

plt.xlabel('Age')

plt.ylabel('EstimatedSalary')

plt.title('SVM Poly')

plt.show()
```

Output :



jupyter SVM Last Checkpoint: a minute ago (autosaved)

```
In [4]: df.isnull().sum()

Out[4]: User ID            0
        Gender             0
        Age                0
        EstimatedSalary    0
        Purchased          0
        dtype: int64

In [5]: plt.scatter(df['Age'], df['EstimatedSalary'])
        plt.xlabel('Age',size = 12)
        plt.ylabel('Estimated Salary',size = 12)
        plt.title('Scatter Plot',size = 14)
        plt.show()
```

jupyter SVM Last Checkpoint: a few seconds ago (autosaved)

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
        from sklearn import svm
        from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
        from mlxtend.plotting import plot_decision_regions

In [2]: df = pd.read_csv('SVM_DataSet_1.csv')

In [3]: df

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
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

jupyter SVM Last Checkpoint: 2 minutes ago (autosaved) Logout

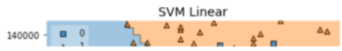
File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3

```
In [6]: x = df[['Age','EstimatedSalary']].values

In [7]: y = df['Purchased'].values

In [8]: #splitting  data into training and testing set
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.4, random_state = 42)

In [9]: # Linear kernel
        linear = svm.SVC(kernel = 'linear', C = 1)
        linear.fit(x_train, y_train)

Out[9]: SVC(C=1, kernel='linear')

In [10]: y_pred = linear.predict(x_test)
         y_pred

Out[10]: array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
                0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
                0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
                0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
                1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
                0, 0, 1, 0, 1, 0], dtype=int64)

In [11]: y_test

Out[11]: array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
                1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
                0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
                1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
                0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0,
                0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
```

jupyter SVM Last Checkpoint: 2 minutes ago (autosaved) Logout

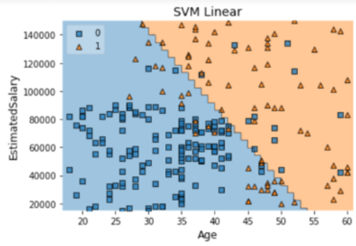
File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3

```
In [12]: confusion_matrix(y_test, y_pred)

Out[12]: array([[95,  5],
                [18, 42]], dtype=int64)

In [13]: accuracy_score(y_test, y_pred)

Out[13]: 0.85625

In [14]: print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.84      0.95      0.89       100
           1       0.89      0.70      0.79        60

    accuracy                           0.86       160
   macro avg       0.87      0.82      0.84       160
weighted avg       0.86      0.86      0.85       160

In [15]: plot_decision_regions(x_train, y_train, clf = linear, legend = 2)

         plt.xlabel('Age',size = 12)

         plt.ylabel('EstimatedSalary',size = 12)

         plt.title('SVM Linear',size = 14)

         plt.show()
```



jupyter SVM Last Checkpoint: 3 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3



In [16]:
```python
# rbf kernel
rbf = svm.SVC(kernel = 'rbf', C = 1e4, gamma = 0.0001)
rbf.fit(x_train, y_train)
```

Out[16]:
```
SVC(C=10000.0, gamma=0.0001)
```

In [17]:
```python
y_pred = rbf.predict(x_test)
y_pred
```

Out[17]:
```
array([1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0], dtype=int64)
```

jupyter SVM Last Checkpoint: 3 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [18]:
```python
y_test
```

Out[18]:
```
array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0], dtype=int64)
```

In [19]:
```python
confusion_matrix(y_test, y_pred)
```

Out[19]:
```
array([[82, 18],
       [23, 37]], dtype=int64)
```

In [20]:
```python
accuracy_score(y_test, y_pred)
```

Out[20]:
```
0.74375
```

In [21]:
```python
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.82      0.80       100
           1       0.67      0.62      0.64        60

    accuracy                           0.74       160
   macro avg       0.73      0.72      0.72       160
weighted avg       0.74      0.74      0.74       160
```

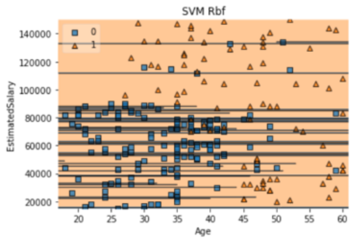
In [22]:
```python
plot_decision_regions(x_train, y_train, clf = rbf, legend = 2)
```

jupyter SVM Last Checkpoint: 4 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

In [22]:
```
plot_decision_regions(x_train, y_train, clf = rbf, legend = 2)
plt.xlabel('Age')
plt.ylabel('EstimatedSalary')
plt.title('SVM Rbf')
plt.show()
```



In [23]:
```
# poly kernel
polynomial = svm.SVC(kernel = 'poly', degree = 3, C = 0.01, gamma = 10, max_iter = 1e5)
polynomial.fit(x_train, y_train)
```

```
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\svm\_base.py:255: ConvergenceWarning: Solver terminated early (max_iter=10000
0).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn('Solver terminated early (max_iter=%i).'
```


jupyter SVM Last Checkpoint: 4 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Out[23]:
```
SVC(C=0.01, gamma=10, kernel='poly', max_iter=100000.0)
```

In [24]:
```
y_pred = polynomial.predict(x_test)
y_pred
```

Out[24]:
```
array([0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0], dtype=int64)
```

In [25]:
```
y_test
```

Out[25]:
```
array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0], dtype=int64)
```

In [26]:
```
confusion_matrix(y_test, y_pred)
```

Out[26]:
```
array([[94,  6],
       [31, 29]], dtype=int64)
```

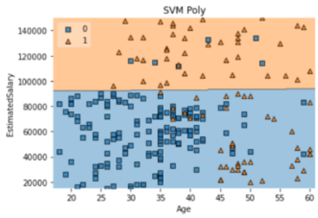
In [27]:
```
accuracy_score(y_test, y_pred)
```

Out[27]:
```
0.76875
```

```
In [28]: print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.94      0.84       100
           1       0.83      0.48      0.61        60

    accuracy                           0.77       160
   macro avg       0.79      0.71      0.72       160
weighted avg       0.78      0.77      0.75       160
```

```
In [29]: plot_decision_regions(x_train, y_train, clf = polynomial, legend = 2)

         plt.xlabel('Age')

         plt.ylabel('EstimatedSalary')

         plt.title('SVM Poly')

         plt.show()
```



S