In [2]:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_repor
t
import matplotlib.pyplot as plt
```

In [3]:

```
perceptron_train = pd.read_csv('../FeaturesCsvFile/featuresfile.csv')
perceptron_test = pd.read_csv('../FeaturesCsvFile/featuresfile_10.csv')
pd.options.display.max_columns = None

print ('(#row,#column) of train dataset' , perceptron_train.shape)
print ('(#row,#column) of test dataset' , perceptron_test.shape)
```

```
('(#row,#column) of train dataset', (417, 46))
('(#row,#column) of test dataset', (518, 46))
```

In [4]:

```
X_train = perceptron_train.values[:, 2:45]
y_train = perceptron_train.values[:, 45]
X_test = perceptron_test.values[:, 2:45]
y_test = perceptron_test.values[:, 45]
```

In [5]:

```
ppn = Perceptron(max_iter=40, eta0=0.1, random_state=1)
ppn.fit(X_train, y_train)
y_pred = ppn.predict(X_test)
```

In [18]:

```
print('Accuracy of Accuracy Score : %.2f' % accuracy_score(y_test,y_pred))
```

```
Accuracy of Accuracy Score : 0.87
```

In [20]:

```
print('Accuracy of Perceptron Score: %.2f' % ppn.score(X_test,y_test))
```

```
Accuracy of Perceptron Score: 0.87
```

In [22]:

```
print ('Confusion_matrix')
print(confusion_matrix(y_test,y_pred))
```

```
Confusion_matrix
[[223  35]
 [ 30 230]]
```

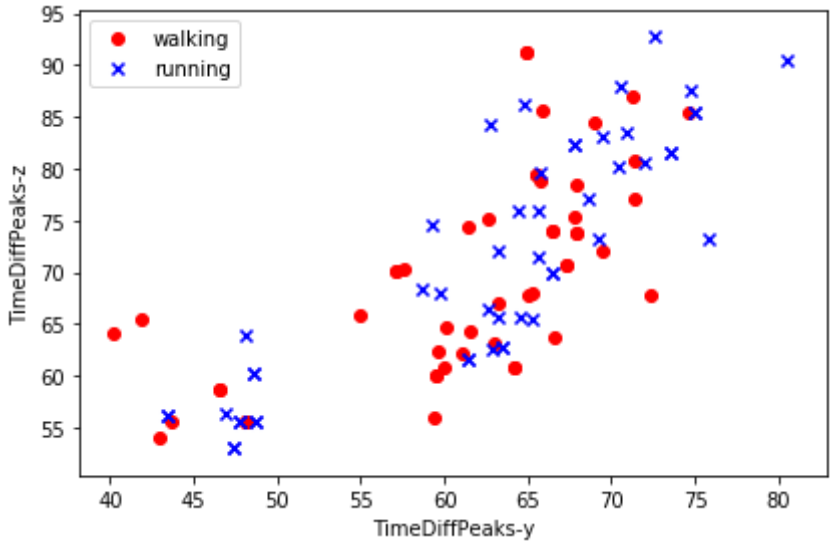
In [40]:

```
print ('Important features')
header = list(perceptron_train.head(1))
for i in range(0,len(ppn.coef_[0])):
    print (i+1,header[i+2],ppn.coef_[0][i])
```

```
Important features
(1, 'Bin1,x', -0.16977406999783445)
(2, 'Bin2,x', -1.0815646018321707)
(3, 'Bin3,x', -1.9577867486305367)
(4, 'Bin4,x', -1.2471331273463679)
(5, 'Bin5,x', 0.70951008977900321)
(6, 'Bin6,x', 1.9878066261171476)
(7, 'Bin7,x', -0.36231032811537289)
(8, 'Bin8,x', -3.3083674892647741)
(9, 'Bin9,x', -3.9264839369849707)
(10, 'Bin10,x', -1.6187632573087811)
(11, 'Bin1,y', -0.65001384163550535)
(12, 'Bin2,y', -2.3856139276609647)
(13, 'Bin3,y', -2.7963128460973725)
(14, 'Bin4,y', -1.9177902271955432)
(15, 'Bin5,y', -1.1891795657443731)
(16, 'Bin6,y', -2.5102637016192655)
(17, 'Bin7,y', -1.4749954714119438)
(18, 'Bin8,y', 0.030242959453366035)
(19, 'Bin9,y', 0.58454906620862035)
(20, 'Bin10,y', 0.65341690434481836)
(21, 'Bin1,z', -0.22895117334842333)
(22, 'Bin2,z', 0.28788337520672441)
(23, 'Bin3,z', -0.068713642445832501)
(24, 'Bin4,z', -2.5128236133892194)
(25, 'Bin5,z', -3.3287458825988301)
(26, 'Bin6,z', -3.3810402789615814)
(27, 'Bin7,z', -1.5604533155591052)
(28, 'Bin8,z', -0.061949601728730153)
(29, 'Bin9,z', 0.043874196703566674)
(30, 'Bin10,z', -0.34931389964077736)
(31, 'TimeDiffPeaks-x', -36.366225442410069)
(32, 'TimeDiffPeaks-y', 31.430106357990134)
(33, 'TimeDiffPeaks-z', 38.452190066290193)
(34, 'AvgAbsDiff-x', -123.62876679629939)
(35, 'AvgAbsDiff-y', -90.099425396863538)
(36, 'AvgAbsDiff-z', -101.82755326677866)
(37, 'AvgAcc-x', 27.749521411401602)
(38, 'AvgAcc-y', 56.016152201096546)
(39, 'AvgAcc-z', -139.29304714169331)
(40, 'StdDev-x', -143.60420869538123)
(41, 'StdDev-y', -108.51305136343348)
(42, 'StdDev-z', -126.95568127565883)
(43, 'AvgResAcc', -8.023092921941064)
```

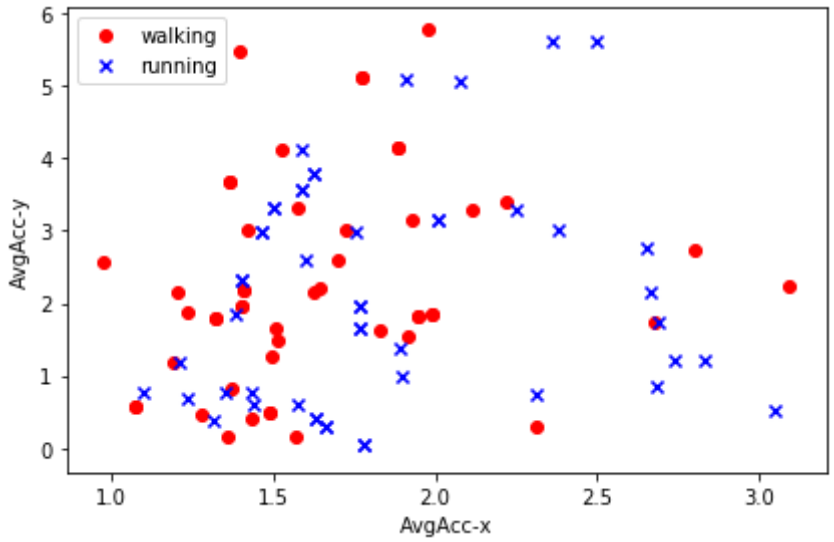
In [57]:

```
#get data first 100 rows
y = perceptron_test.iloc[0:100, 45].values
y = np.where(y == 'walking', -1, 1)
#use TimeDiffPeaks-y and TimeDiffPeaks-z
X = perceptron_test.iloc[0:100, [32, 34]].values
# plot data
plt.scatter(X[:50, 0], X[:50, 1],
            color='red', marker='o', label='walking')
plt.scatter(X[50:100, 0], X[50:100, 1],
            color='blue', marker='x', label='running')

plt.xlabel('TimeDiffPeaks-y')
plt.ylabel('TimeDiffPeaks-z')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



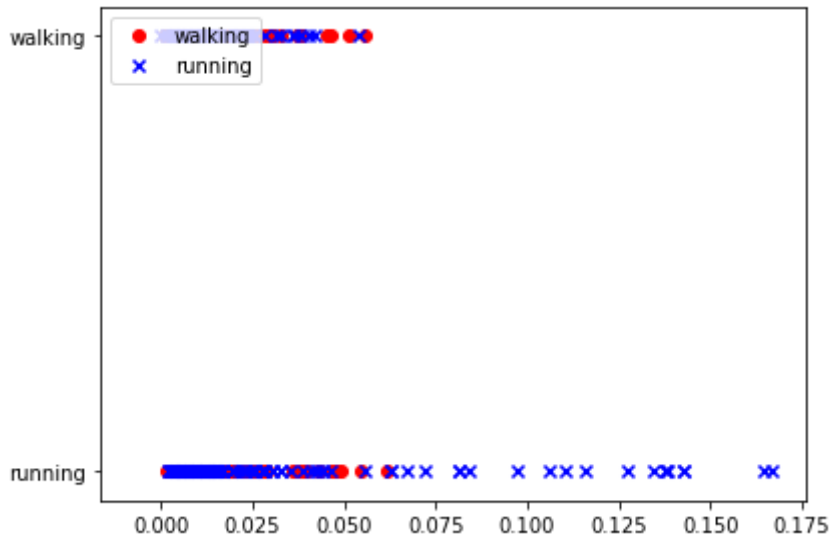
In [58]:

```
y = perceptron_test.iloc[0:100, 45].values
y = np.where(y == 'walking', -1, 1)
#use AvgAcc-x and AvgAcc-y
X = perceptron_test.iloc[0:100, [37, 39]].values
# plot data
plt.scatter(X[:50, 0], X[:50, 1],
            color='red', marker='o', label='walking')
plt.scatter(X[50:100, 0], X[50:100, 1],
            color='blue', marker='x', label='running')

plt.xlabel('AvgAcc-x')
plt.ylabel('AvgAcc-y')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



In [7]:

```
y = perceptron_test.iloc[0:, 45].values
y = np.where(y == 'walking', -1, 1)
#use AvgAcc-x and AvgAcc-y
X = perceptron_test.iloc[0:, [2,45]].values
# plot data
plt.scatter(X[:len(perceptron_test)/2, 0], X[:len(perceptron_test)/2, 1],
            color='red', marker='o', label='walking')
plt.scatter(X[len(perceptron_test)/2:len(perceptron_test), 0], X[len(perceptron_
test)/2:len(perceptron_test), 1],
            color='blue', marker='x', label='running')

plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



In [ ]: