ASSIGNMENT-4

1) This is the code that prints accuracy of the trained model

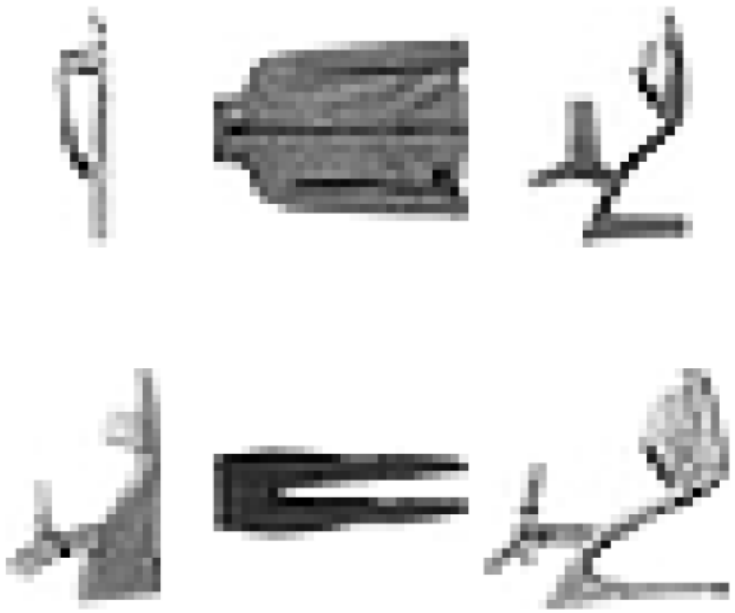
```
model = costing()
all_w,acrcy = model.minCostFun(train)
acrcy_test = model.TestingAccu(test,all_w)
print(acrcy)
print(acrcy_test)
```

```
93.2
86.667
```

2) This is the code for displaying some training data and the output images

```
def displayData(X, example_width=None, figsize=(10, 10)):
    """
    Displays 2D data stored in X in a nice grid.
    """
    # Compute rows, cols
    if X.ndim == 2:
        m, n = X.shape
    elif X.ndim == 1:
        n = X.size
        m = 1
        X = X[None]  # Promote to a 2 dimensional array
    else:
        raise IndexError('Input X should be 1 or 2 dimensional.')

    example_width = example_width or int(np.round(np.sqrt(n)))
    example_height = n / example_width

    # Compute number of items to display
    display_rows = int(np.floor(np.sqrt(m)))
    display_cols = int(np.ceil(m / display_rows))

    fig, ax_array = plt.subplots(display_rows, display_cols, figsize=figsize)
    fig.subplots_adjust(wspace=0.025, hspace=0.025)

    ax_array = [ax_array] if m == 1 else ax_array.ravel()

    for i, ax in enumerate(ax_array):
        ax.imshow(X[i].reshape(example_width, example_width, order='F'),
                  cmap='Greys', extent=[0, 1, 0, 1])
        ax.axis('off')


X, y = model.data_clean(train)
rand_ind = np.random.choice(y.shape[0],6, replace=False) # Randomly select 6 data points to display
a = X[rand_ind, :]
print(y[rand_ind])

displayData(a)
```

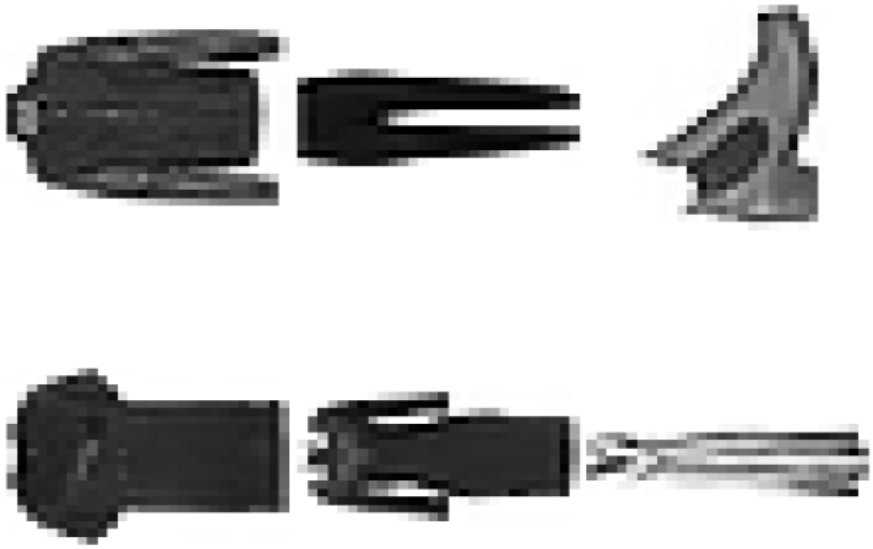
```
[5 4 5 5 1 5]
```



3) This displays some images from the test data with the prediction in an array

```
X_test,y_test=costing().data_clean(test)
rand_ind2 = np.random.choice(y_test.shape[0],6, replace=False)
b=X_test[rand_ind2, :]
p=model.predictOneVsAll(all_w,X,10)
print(y_test[rand_ind2])
displayData(b)
```

```
[4 1 9 0 3 3]
```



4) This plots the cost history vs number of iterations.

```
cost_history = model.J_hist
print(cost_history)
plt.xlabel('Number of iterations')
plt.ylabel('Cost History')
plt.title('Plot of Cost History vs Number of iterations')
a = cost_history
b = np.array([i for i in range(len(cost_history))])
plt.plot(b,a)
```