Question 1: Logistic regression

```
1. python3 Classification.py --algorithm logistic --data linear --feature linear --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  1.0
        Accuracy: validation set:  1.0
        Accuracy: test set:  1.0
2. python3 Classification.py --algorithm logistic --data quadratic --feature linear --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  0.6416666666666667
        Accuracy: validation set:  0.675
        Accuracy: test set:  0.655
3. python3 Classification.py --algorithm logistic --data quadratic --feature quadratic --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  0.9733333333333334
        Accuracy: validation set:  0.98
        Accuracy: test set:  0.955
4. python3 Classification.py --algorithm logistic --data noisy_linear --feature linear --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  0.6833333333333333
        Accuracy: validation set:  0.9
        Accuracy: test set:  0.85
5. python3 Classification.py --algorithm logistic --data mnist --feature linear --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  0.8713333333333333
        Accuracy: validation set:  0.8485
        Accuracy: test set:  0.847
```

Discussion： For this algorithm it works very well on those four linear separated data (task 1, task task 3 and task 4). But when it comes to quadratic data, this algorithm doesn’t work very well. Then when we change our data feature to quadratic, it can separate data very well.

Question 2: Pocket algorithm

```
1.  python3 Classification.py --algorithm pocket --data linear --feature linear --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  1.0
        Accuracy: validation set:  1.0
        Accuracy: test set:  1.0
2.  python3 Classification.py --algorithm pocket --data quadratic --feature linear --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  0.5283333333333333
        Accuracy: validation set:  0.545
        Accuracy: test set:  0.52
3.  python3 Classification.py --algorithm pocket --data quadratic --feature quadratic --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  0.97
        Accuracy: validation set:  0.965
        Accuracy: test set:  0.97
4.  python3 Classification.py --algorithm pocket --data noisy_linear --feature linear --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  0.6666666666666666
        Accuracy: validation set:  0.7
        Accuracy: test set:  0.75
5.  python3 Classification.py --algorithm pocket --data mnist --feature linear --step_size 0.1 --
max_iterations 500
        Accuracy: training set:  0.8421666666666666
        Accuracy: validation set:  0.812
        Accuracy: test set:  0.7995
```

Discussion：For this algorithm it works very well on those four linear separated data (task 1, task task 3 and task 4). But when it comes to quadratic data, this algorithm doesn’t work very well. Then when we change our data feature to quadratic, it can separate data very well.

Question 3: Soft-margin SVM algorithm

```
1.python3 Classification.py --algorithm SVM --data linear --feature linear --step_size 0.1 --
max_iterations 500 --reg_coeff 0.1
        Accuracy: training set:  1.0
        Accuracy: validation set:  1.0
        Accuracy: test set:  1.0
2. python3 Classification.py --algorithm SVM --data quadratic --feature linear --step_size 0.1 --
max_iterations 500 --reg_coeff 0.1
        Accuracy: training set:  0.5283333333333333
        Accuracy: validation set:  0.545
        Accuracy: test set:  0.52
3. python3 Classification.py --algorithm SVM --data quadratic --feature quadratic --step_size 0.1 --
max_iterations 500 --reg_coeff 0.001
        Accuracy: training set:  0.9733333333333334
        Accuracy: validation set:  0.98
        Accuracy: test set:  0.965
4. python3 Classification.py --algorithm SVM --data noisy_linear --feature linear --step_size 0.1 --
max_iterations 500 --reg_coeff 0.1
        Accuracy: training set:  0.7
        Accuracy: validation set:  0.95
        Accuracy: test set:  1.0
5. python3 Classification.py --algorithm SVM --data mnist --feature linear --step_size 0.1 --
max_iterations 500 --reg_coeff 0.0
        Accuracy: training set:  0.8578333333333333
        Accuracy: validation set:  0.8415
        Accuracy: test set:  0.841
```

Discussion：For this algorithm it works very well on those four linear separated data (task 1, task task 3 and task 4). But when it comes to quadratic data, this algorithm doesn’t work very well. Then when we change our data feature to quadratic, it can separate data very well.

Question 4: Naive Bayes linear classifier

```
1.  python3 Classification.py --algorithm NB_linear --data linear --feature linear
        Accuracy: training set:  0.9966666666666667
        Accuracy: validation set:  1.0
        Accuracy: test set:  1.0
2.  python3 Classification.py --algorithm NB_linear --data quadratic --feature linear
        Accuracy: training set:  0.6416666666666667
        Accuracy: validation set:  0.67
        Accuracy: test set:  0.655
3.  python3 Classification.py --algorithm NB_linear --data quadratic --feature quadratic
        Accuracy: training set:  0.9683333333333334
        Accuracy: validation set:  0.97
        Accuracy: test set:  0.945
4.  python3 Classification.py --algorithm NB_linear --data noisy_linear --feature linear
        Accuracy: training set:  0.6833333333333333
        Accuracy: validation set:  0.9
        Accuracy: test set:  0.95
5.  python3 Classification.py --algorithm NB_linear --data mnist --feature linear
        Accuracy: training set:  0.7785
        Accuracy: validation set:  0.7575
        Accuracy: test set:  0.758
```

Discussion： For this algorithm it works very well on those four linear separated data (task 1, task task 3 and task 4). But when it comes to quadratic data, this algorithm doesn’t work very well. Then when we change our data feature to quadratic, it can separate data very well.

Question 5: GDA linear classifier

```
1.  python3 Classification.py --algorithm GDA_linear --data linear --feature linear
        Accuracy: training set:  1.0
        Accuracy: validation set:  1.0
        Accuracy: test set:  1.0
2.  python3 Classification.py --algorithm GDA_linear --data quadratic --feature linear
        Accuracy: training set:  0.64
        Accuracy: validation set:  0.675
        Accuracy: test set:  0.655
3.  python3 Classification.py --algorithm GDA_linear --data quadratic --feature quadratic
        Accuracy: training set:  0.9683333333333334
        Accuracy: validation set:  0.965
        Accuracy: test set:  0.945
4.  python3 Classification.py --algorithm GDA_linear --data noisy_linear --feature linear
        Accuracy: training set:  0.6833333333333333
        Accuracy: validation set:  0.95
        Accuracy: test set:  0.95
5.  python3 Classification.py --algorithm GDA_linear --data mnist --feature linear
        Accuracy: training set:  0.8885
        Accuracy: validation set:  0.843
        Accuracy: test set:  0.8355
```

Discussion： For this algorithm it works very well on those four linear separated data (task 1, task 3 and task 4). But when it comes to quadratic data, this algorithm doesn’t work very well. Then when we change our data feature to quadratic, it can separate data very well.

Question 6: Naive Bayes nonlinear classifier

```
1.python3 Classification.py --algorithm NB_nonlinear --data linear --feature linear
        Accuracy: training set:  0.9983333333333333
        Accuracy: validation set:  1.0
        Accuracy: test set:  1.0
2.python3 Classification.py --algorithm NB_nonlinear --data quadratic --feature linear
        Accuracy: training set:  0.96
        Accuracy: validation set:  0.96
        Accuracy: test set:  0.96
3.python3 Classification.py --algorithm NB_nonlinear --data quadratic --feature quadratic
        Accuracy: training set:  0.96
        Accuracy: validation set:  0.95
        Accuracy: test set:  0.965
4.python3 Classification.py --algorithm NB_nonlinear --data noisy_linear --feature linear
        Accuracy: training set:  0.6666666666666666
        Accuracy: validation set:  0.9
        Accuracy: test set:  0.85
5.python3 Classification.py --algorithm NB_nonlinear --data mnist --feature linear
        Accuracy: training set:  0.914
        Accuracy: validation set:  0.9125
        Accuracy: test set:  0.9065
```

Discussion：This algorithm can non-linearly separate data, so it separated all kinds of data very well no matter the data is in linear type or quadratic type. For data with some noise, I think it is inevitable to have lower accuracy.

Question 7: GDA nonlinear classifier

```
1.python3 Classification.py --algorithm GDA_nonlinear --data linear --feature linear
        Accuracy: training set:  1.0
        Accuracy: validation set:  1.0
        Accuracy: test set:  1.0
2.python3 Classification.py --algorithm GDA_nonlinear --data quadratic --feature linear
        Accuracy: training set:  0.96
        Accuracy: validation set:  0.955
        Accuracy: test set:  0.96
3.python3 Classification.py --algorithm GDA_nonlinear --data quadratic --feature quadratic
        Accuracy: training set:  0.97
        Accuracy: validation set:  0.965
        Accuracy: test set:  0.965
4.python3 Classification.py --algorithm GDA_nonlinear --data noisy_linear --feature linear
        Accuracy: training set:  0.6833333333333333
        Accuracy: validation set:  0.85
        Accuracy: test set:  0.85
6.  python3 Classification.py --algorithm GDA_nonlinear --data mnist --feature linear
        It takes too long to run mnist. :(
```

Discussion：This algorithm can non-linearly separate data, so it separated all kinds of data very well no matter the data is in linear type or quadratic type. For data with some noise, I think it is inevitable to have lower accuracy.