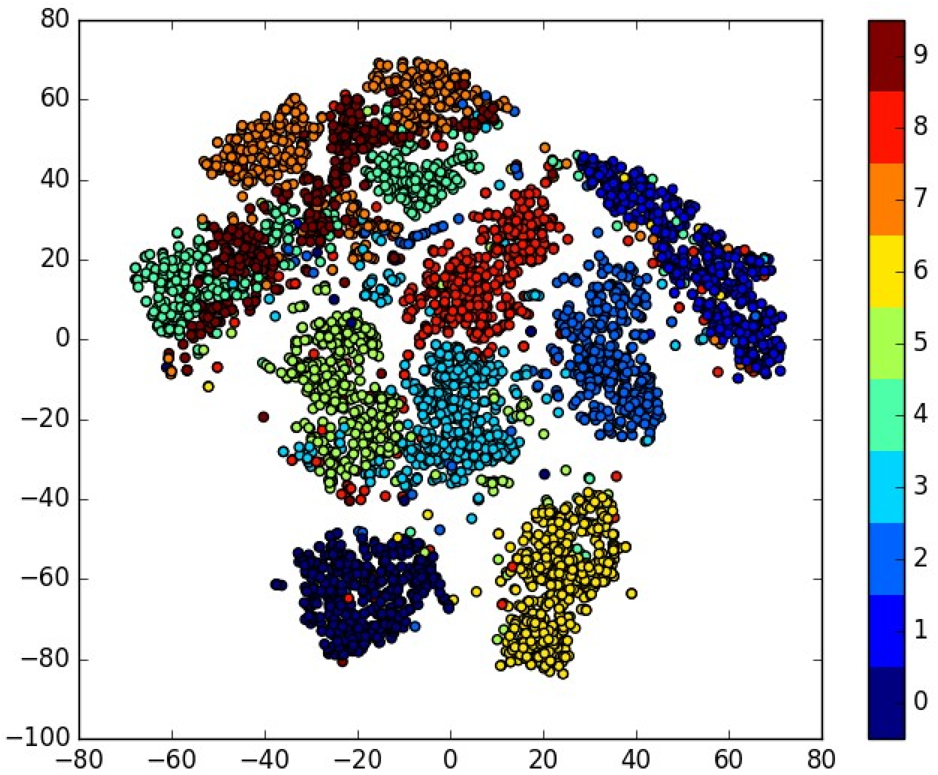
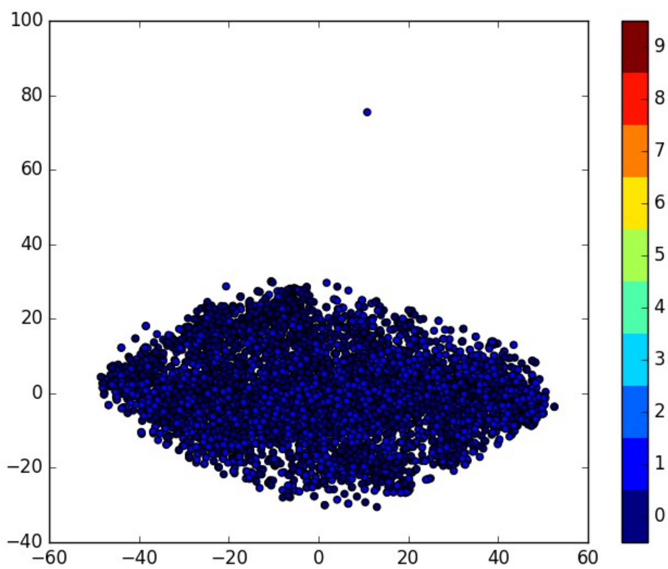
1.

Visualise the three datasets. For this, you'll need load the h5 files and use t-SNE to plot the data. Use sklearn's implementation of t-SNE . You are free to use any parameters so that the plots make sense.

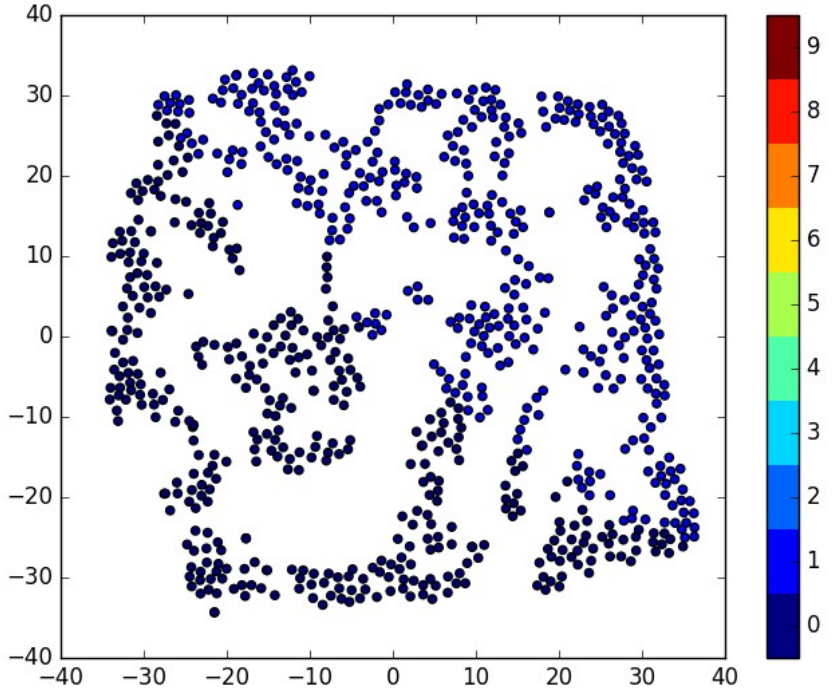
Dataset part_A: Consists of 10 classes



Dataset part_B: Consists of 2 classes



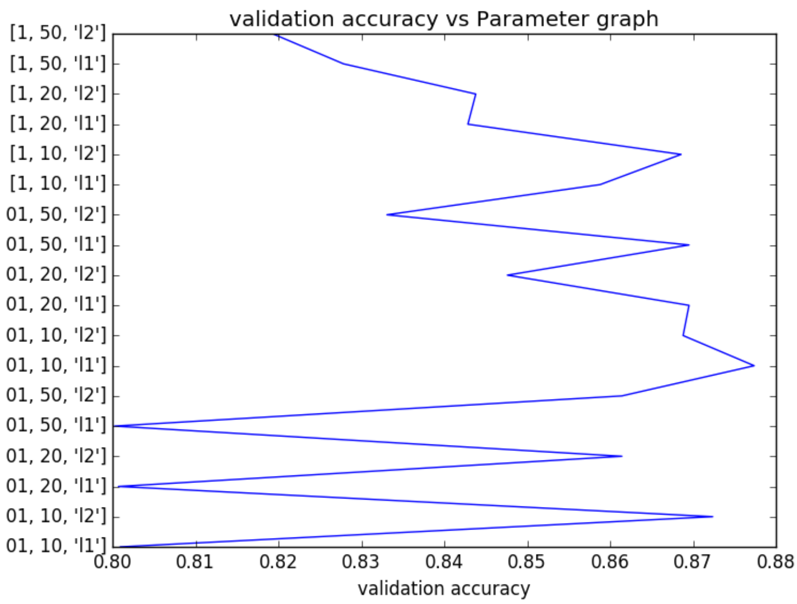
Dataset part_C: Consists of 2 classes



Q2. c. Plot the validation accuracy vs the parameters in the grid.

- **LogisticRegression**

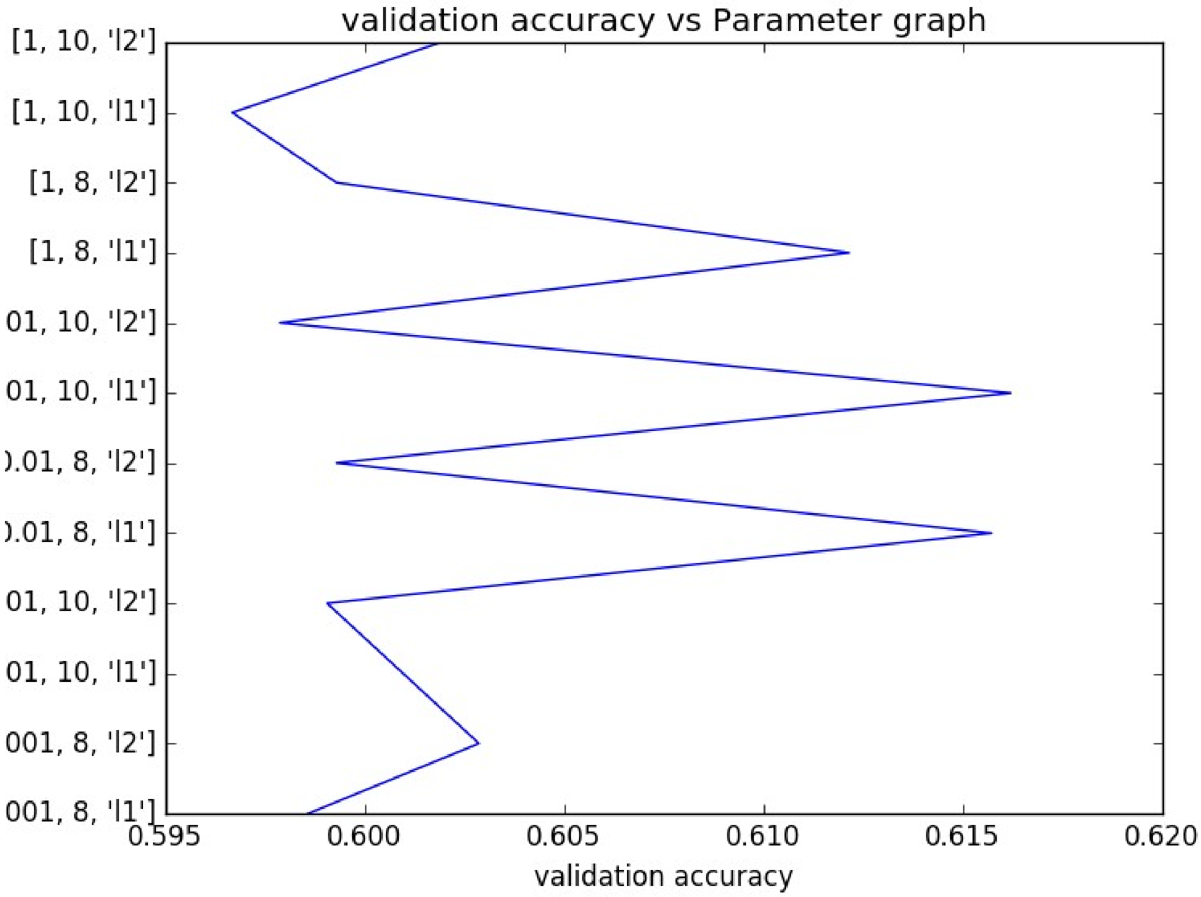
1. Dataset part_A



Best parameters [C,max-iter,penalty] are :

```
garvita@garvita-HP-Pavilion-15-Notebook-PC:~/Desktop/machine learning/homework1_
12909/Template$ python train.py --model_name LogisticRegression --train_data "/h
ome/garvita/Desktop/machine learning/homework1_12909/Template/Data/part_A_train.
h5"
(18, 3)
[0.80095238095238097, 0.87238095238095248, 0.80071428571428582, 0.86142857142857
143, 0.8002380952380953, 0.86142857142857143, 0.87738095238095237, 0.86880952380
952381, 0.86952380952380948, 0.84761904761904761, 0.86952380952380948, 0.8330952
3809523811, 0.85880952380952391, 0.86857142857142855, 0.84285714285714286, 0.843
80952380952379, 0.82785714285714285, 0.81928571428571428]
[0.01, 10, 'l1']
0.877380952381
```

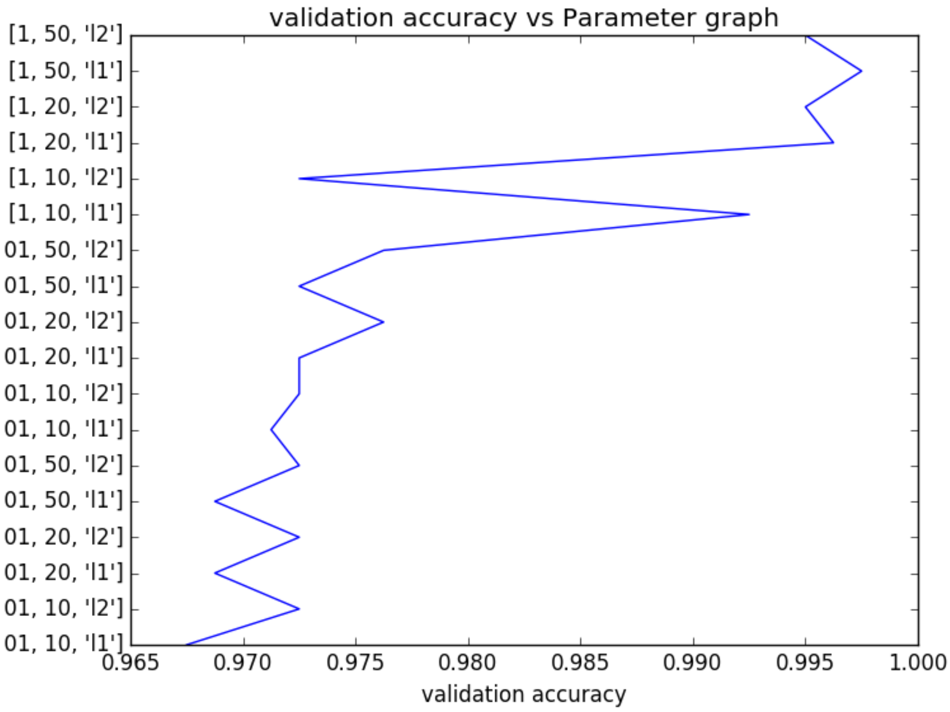
2. Dataset part_B



Best parameters [C,max-iter,penalty] are :

```
12909/Template$ python train.py --model_name LogisticRegression --train_data "/h
ome/garvita/Desktop/machine learning/homework1_12909/Template/Data/part_B_train.
h5"
(12, 3)
[0.01, 10, 'l1']
0.61619047619
```

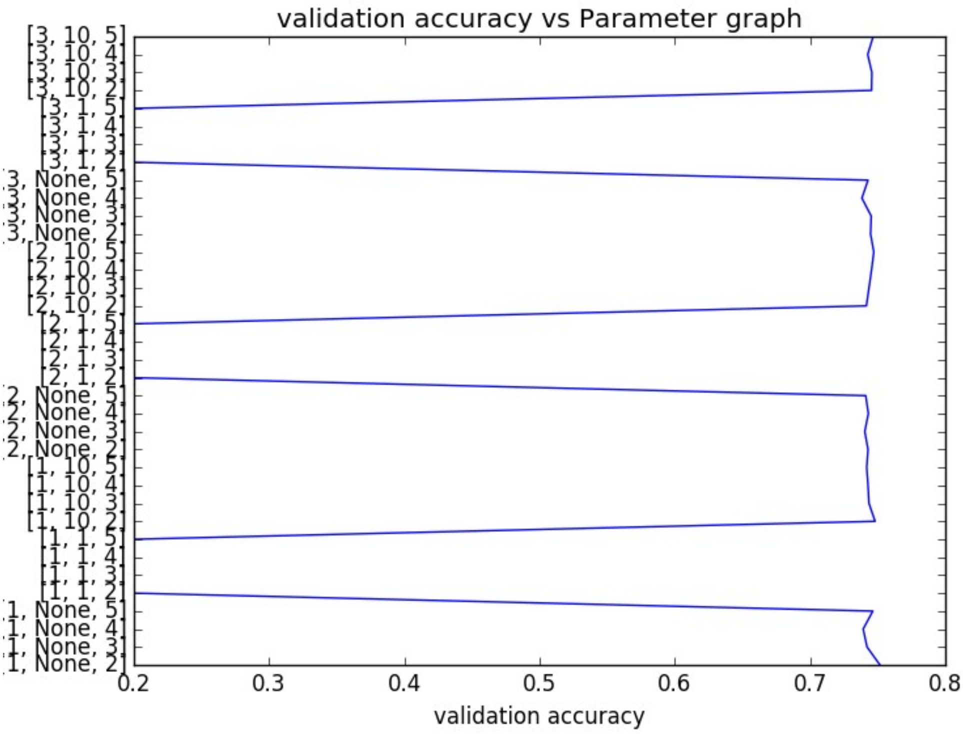
3.DataSet part_C



Best parameters [C,max-iter,penalty] are :

```
garvita@garvita-HP-Pavilion-15-Notebook-PC:~/Desktop/machine learning/homework1_
12909/Template$ python train.py --model_name LogisticRegression --train_data "/h
ome/garvita/Desktop/machine learning/homework1_12909/Template/Data/part_C_train.
h5"
(18, 3)
[0.96750000000000003, 0.97250000000000014, 0.96875000000000022, 0.97250000000000
014, 0.96875000000000022, 0.97250000000000014, 0.97125000000000006, 0.9725000000
0000014, 0.97250000000000014, 0.97625000000000006, 0.97250000000000014, 0.976250
00000000006, 0.99250000000000005, 0.97250000000000014, 0.99625000000000008, 0.99
499999999999988, 0.99749999999999994, 0.99499999999999988]
[1, 50, 'l1']
0.9975
```

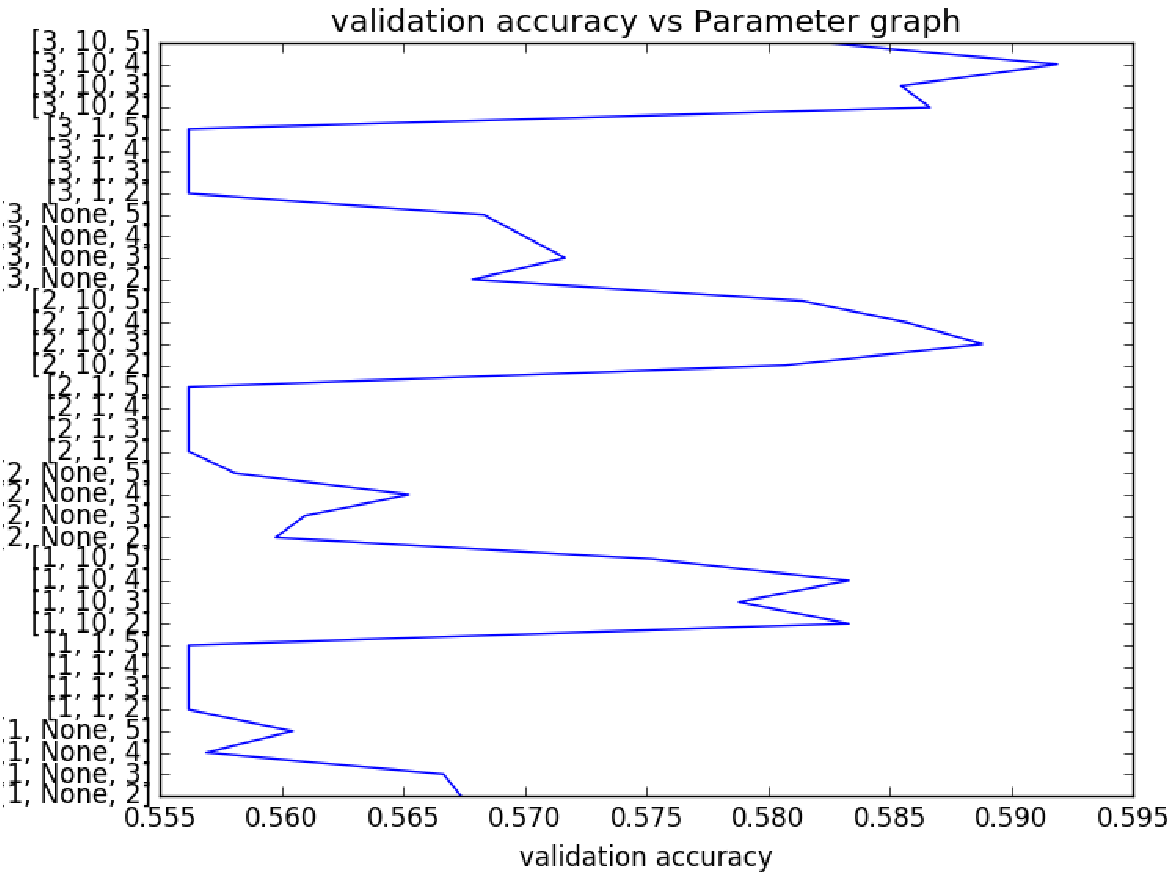
- **DecisionTreeClassifier**
  1. **DataSet part_A**



Best parameters [min_samples_leaf,max_depth,min_samples_split] are :

```
garvita@garvita-HP-Pavilion-15-Notebook-PC:~/Desktop/machine learning/homework1_12909/Template$ python train.py --model_name DecisionTreeClassifier --train_data "/home/garvita/Desktop/machine learning/homework1_12909/Template/Data/part_A_train.h5"
(36, 3)
[1, None, 2]
0.751666666667
```

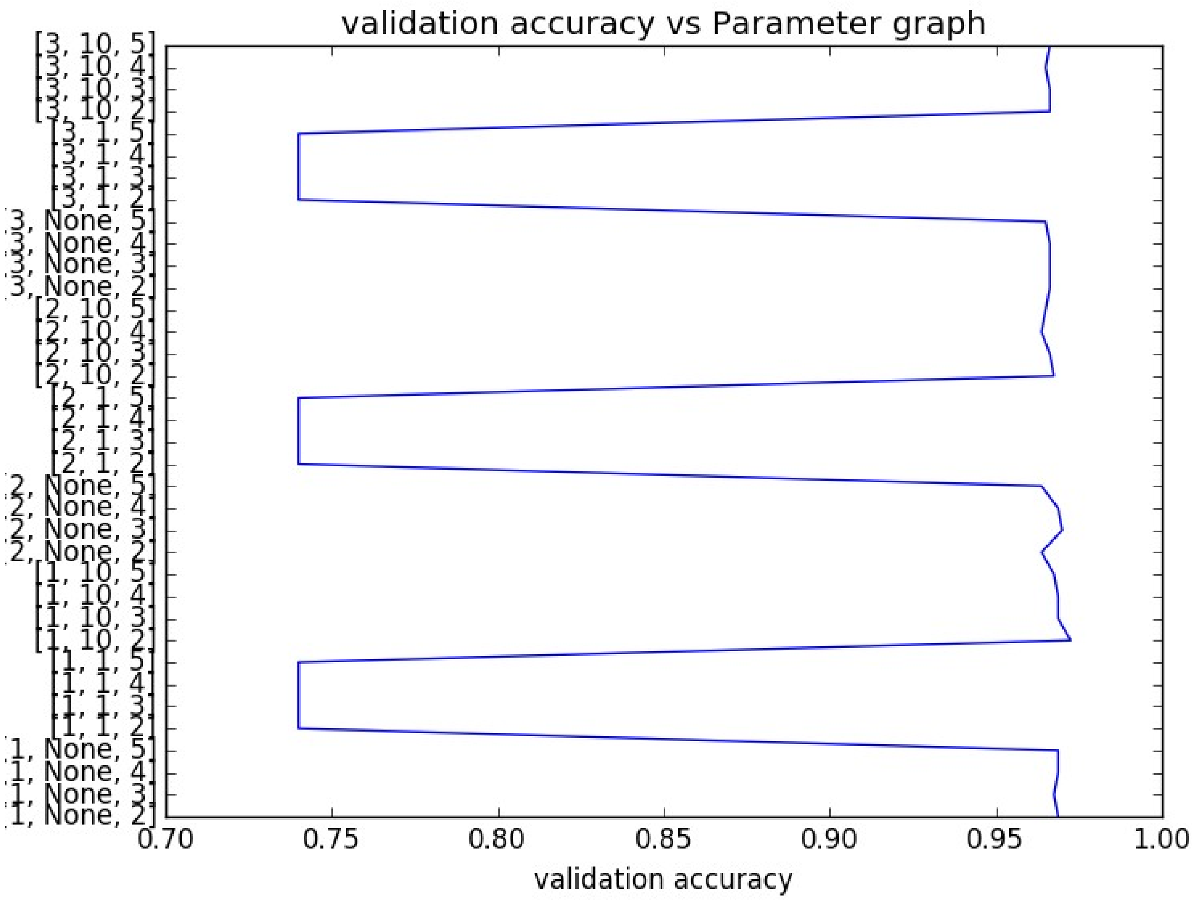
2. Dataset Part_B



Best parameters [min_samples_leaf,max_depth,min_samples_split] are :

```
garvita@garvita-HP-Pavilion-15-Notebook-PC:~/Desktop/machine learning/homework1_
12909/Template$ python train.py --model_name DecisionTreeClassifier --train_data
 "/home/garvita/Desktop/machine learning/homework1_12909/Template/Data/part_B_tr
ain.h5"
(36, 3)
[3, 10, 4]
0.591904761905
```

3. Dataset Part_C

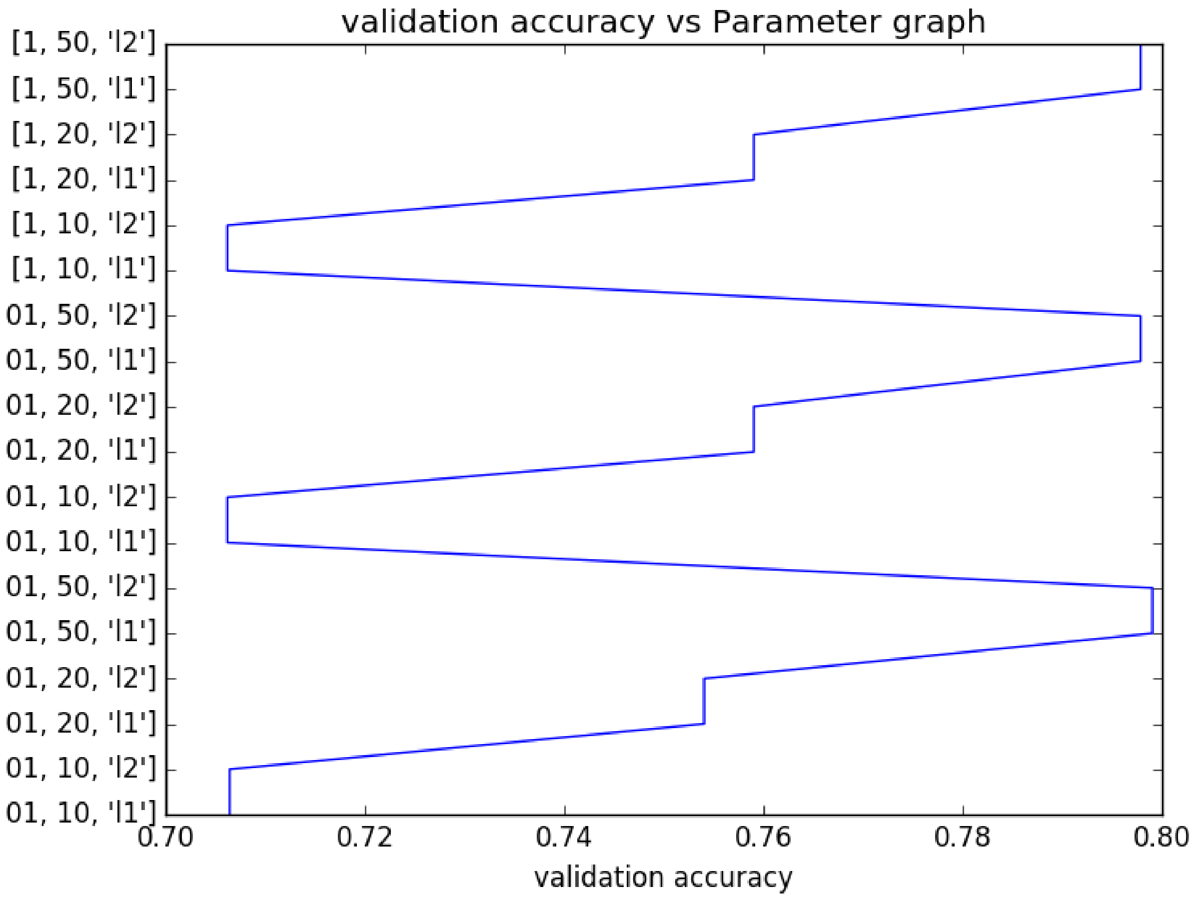


Best parameters [min_samples_leaf,max_depth,min_samples_split] are :

```
garvita@garvita-HP-Pavilion-15-Notebook-PC:~/Desktop/machine learning/homework1_
12909/Template$ python train.py --model_name DecisionTreeClassifier --train_data
 "/home/garvita/Desktop/machine learning/homework1_12909/Template/Data/part_C_tr
ain.h5"
(36, 3)
[1, 10, 2]
0.9725
```

Q3. c. Plot the validation accuracy vs the parameters in the grid.
Logistic Regression

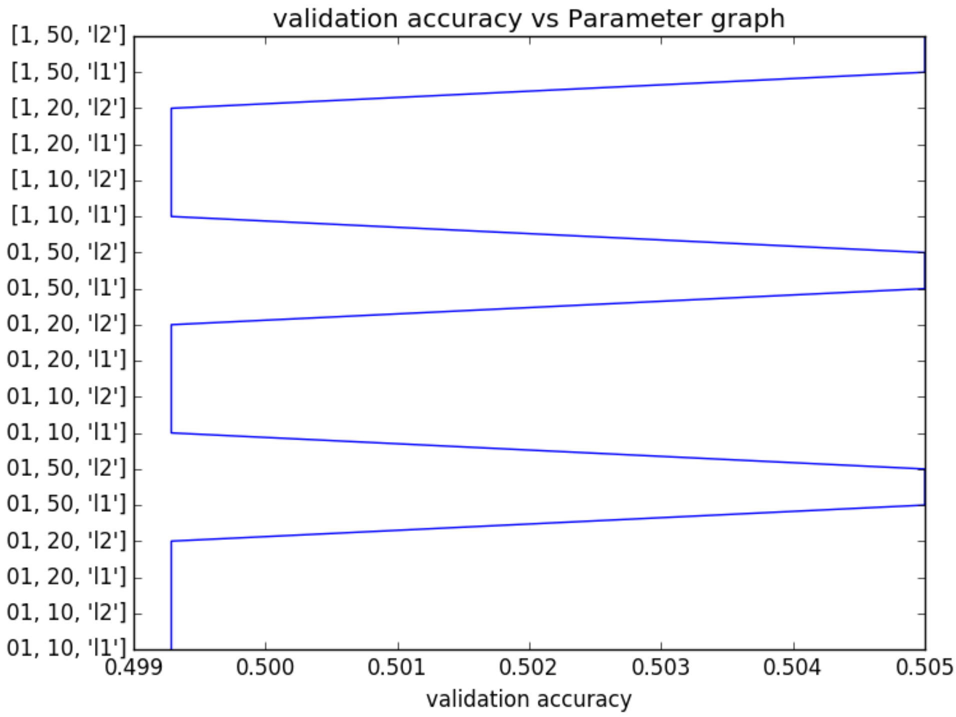
Dataset Part_A



Best parameters [C,max-iter,penalty] are :

```
garvita@garvita-HP-Pavilion-15-Notebook-PC:~/Desktop/machine learning/homework1_
12909/Template$ python train.py --model_name LogisticRegression --train_data "/h
ome/garvita/Desktop/machine learning/homework1_12909/Template/Data/part_A_train.
h5"
/home/garvita/Desktop/machine learning/homework1_12909/Template/Models/LogisticR
egression.py:39: RuntimeWarning: overflow encountered in exp
  predictions=1/(1+np.exp(-scores))
[0.70642857142857141, 0.70642857142857141, 0.75404761904761908, 0.75404761904761
908, 0.79904761904761901, 0.79904761904761901, 0.70619047619047615, 0.7061904761
9047615, 0.75904761904761908, 0.75904761904761908, 0.79785714285714282, 0.797857
14285714282, 0.70619047619047615, 0.70619047619047615, 0.75904761904761908, 0.75
904761904761908, 0.79785714285714282, 0.79785714285714282]
[0.0001, 50, 'l1']
0.799047619048
```

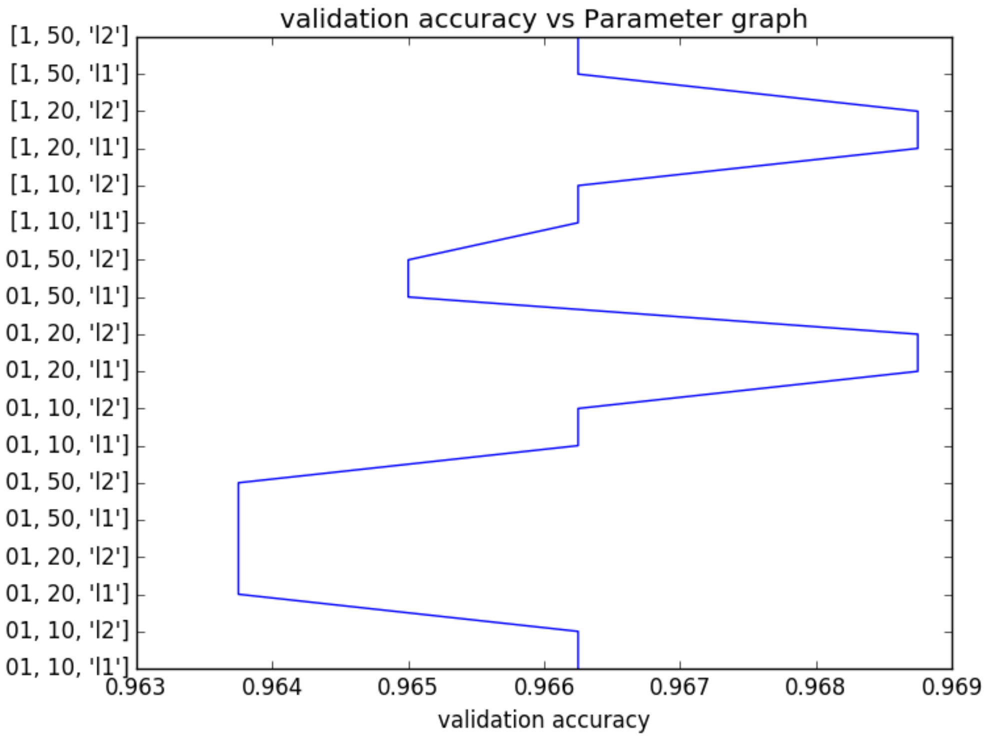
Dataset part_B



Best parameters [C,max-iter,penalty] are :

```
[0.0001, 50, 'l1']
0.505
```

Dataset part_C



Best parameters [C,max-iter,penalty] are :

```
99999994]
[0.01, 20, 'l1']
0.96875
```

Theory Questions

Q1. The minima of a given function may be found using its first order derivative and equating it to zero (and second order derivative > 0, etc). Consider the case of a simple linear regression model. Why don't we then, in all cases, simply find the minima of this function using a similar approach, instead of using gradient descent which is obviously slower.

Ans: The reasons for using gradient descent rather than finding first order derivative and equating it to 0 are as follows:

- All cost functions are **not differentiable every where**. Even if they are, it is **difficult to find analytical solution of it.**
  For example: If a function is of order 5 then it's derivative will be of order 4.In case of finding minima, we equate that to 0. Now, it is very difficult to solve this fourth order equation to find minimum points.
- In gradient descent **we improve our answer at each step** and get an **answer close to "true answer"** whereas using derivative the **best answer** close to true one can be **skipped.**
- Most of the model functions are **convex**. So, it is **assured** that gradient descent will take it to **extrema**.
- **Complex functions** are more easily solved using gradient descent.

Q2. How is machine learning different from function approximation? Would the two be the same if we had all the possible data that the model is expected to ever see?

- Machine learning **learns from data without relying** on **any rule based programming**.It learns from **each experience E** and **use that knowledge** for different data.
  Whereas in function approximation,**relationship between variables is determined**.For that it should have **idea about independent and dependent variables.**It **doesn't learn anything.**
- Function approximation provides a simple formulation of problem whereas machine learning captures all patterns across decision boundary.

If we had all the possible data that the model is expected to ever see, then function approximation and machine learning **may or may not be the same** due to following reasons:

- It the **size of dataset is large**, then function approximation **will give more erroneous output** as errors will not be reduced in due course of time rather it will get added whereas in **machine learning** this **error will decrease with time as the machine learns and with increase in training data.**
- Function approximation **wont be able to handle more parameters** as function will become **more complex** and

4. Let $x_1, x_2, \ldots, x_n$ be i.i.d. data from a uniform distribution over the disc of radius θ in R^2, i.e., $x_i \in R^2$ and

$$p(x;\theta) = \begin{cases} \frac{1}{\pi\theta^2}, & \|x\| \le \theta \\ 0, & otherwise \end{cases}$$

where $\|x\| = \sqrt{x_1^2 + x_2^2}$. What is the maximum likelihood estimate of θ.

Date: / / Page No.:

Q4 Maximum Likelihood ~~has~~ of θ is given as

$$L(\theta) = \prod_{i=1}^{n} f(x_i;\theta)$$

Taking log likelihood, we get

$$\log L(\theta) = \log\left[\prod_{i=1}^{n} f(x_i;\theta)\right]$$

$$\log L(\theta) = \sum_{i=1}^{n} \log f(x_i;\theta) \quad \text{—(i)}$$

now, $f(x_i;\theta) = \frac{1}{\pi\theta^2}$

putting it in equation (i)

$$l(\theta) = \sum_{i=1}^{n} \log\left(\frac{1}{\pi\theta^2}\right)$$

$$= \sum_{i=1}^{n} -\log\pi - 2\log\theta$$

Taking derivative of $l(\theta)$ with respect to θ, we get

$$\frac{\partial l(\theta)}{d\theta} = \cancel{\sum} - \frac{2n}{\theta}$$

Since, it can be differentiated. Don't use derivative

$$\max(l(\theta)) = \max\left(-\sum_{i=1}^{n} \log\pi + 2\log\theta\right)$$

$$= \min\left(\sum_{i=1}^{n} \log\pi + 2\log\theta\right)$$

$$\simeq \min(2n\log\theta)$$

Date: / / Page No.

$\min(2n \log \theta)$ subject to condition $||x|| < \theta$

So, maximum likelihood estimate of

$$\boxed{\theta \text{ is } \max ||x||}$$

Q3 solution :In Template folder with name question 3

Yes gradient descent can be used .it will be more easily solved with that