

## Total Experiments conducted in the Assignment are 32

### Dataset 1

#### Energy consumption data set

In Part-1 I have applied various classification algorithms on the Energy consumption data set and classified the houses as “high energy” consumers or “low energy” consumers.

I have dropped the lights and the date column from the dataset as they are not useful for us when doing the classification. I have split the dataset into training set and testing set with a ratio of 30% testing set and 70% training set. And standard student scaling is done on all the feature variables. Student scaled variable has a mean of 0 and standard distribution of 1.

### Dataset-2

In Part-2, I have tried to predict if a customer is going to quit the services of a Telecom operator based on various features.

I selected this dataset as Churn is a one of the biggest problems in the telecom industry. Research has shown that the average monthly churn rate among the top 4 wireless carriers in the US is 1.9% - 2%. If we are able to predict the churn of a customer this will be worth a lot of money as Telecom company can try to save the customer who may quit their services by offering him better plans and offers.

To better understand this problem we can define Churn as Customer attrition, also known as customer churn, customer turnover, or customer defection, is the loss of clients or customers.

Telephone service companies, Internet service providers, pay TV companies, insurance firms, and alarm monitoring services, often use customer attrition analysis and customer attrition rates as one of their key business metrics because the cost of retaining an existing customer is far less than acquiring a new one. Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients.

Companies usually make a distinction between voluntary churn and involuntary churn. Voluntary churn occurs due to a decision by the customer to switch to another company or service provider, involuntary churn occurs due to circumstances such as a customer's relocation to a long-term care facility, death, or the relocation to a distant location. In most applications, involuntary reasons for churn are excluded from the analytical models. Analysts tend to concentrate on voluntary churn, because it typically occurs due to factors of the company-customer relationship which companies control, such as how billing interactions are handled or how after-sales help is provided.

predictive analytics use churn prediction models that predict customer churn by assessing their propensity of risk to churn. Since these models generate a small prioritized list of potential defectors, they are effective at

focusing customer retention marketing programs on the subset of the customer base who are most vulnerable to churn.

#### Summary statistics of the dataset used:

- Dataset consists of 7043 observations on 21 variables.
- The dependent variable for the Classification model is Churn variable which is Binary column with values “Yes” if the customer quit the services and “No” if the customer is still with the telecom operator.

We then proceed to convert the predictor variable in a binary numeric variable with values 1 for “Yes” and 0 for “No”. And as Machine Algorithms only understand numbers we convert all the categorical variables into dummy variables.

We now have 46 columns.

I have split the dataset into training set and testing set with a ratio of 30% testing set and 70% training set. And standard student scaling is done on all the feature variables. Student scaled variable has a mean of 0 and standard distribution of 1.

### Artificial Neural Network

1. Download and use any neural networks package to classify your classification problems. Experiment with number of layers and number of nodes, activation functions (sigmoid, tanh, etc.), and may be a couple of other parameters.

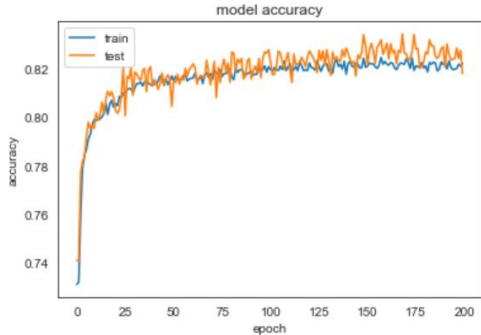
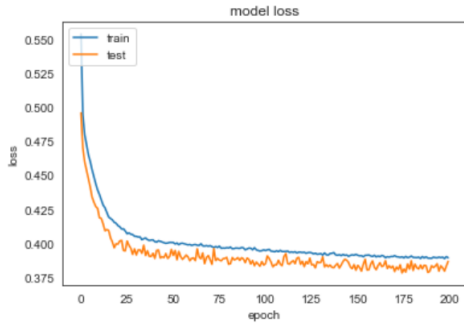
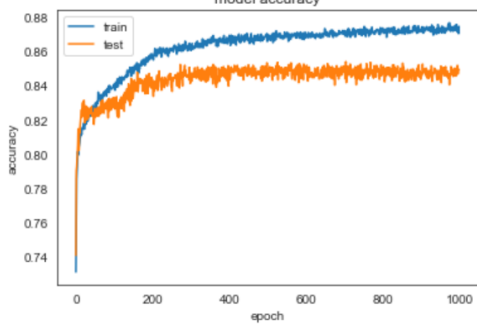
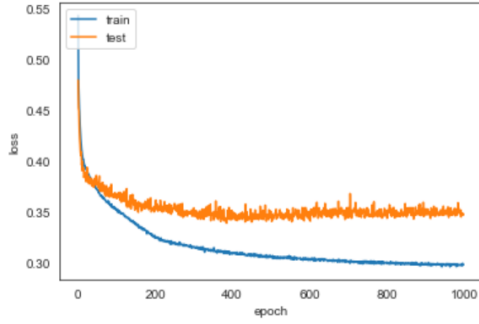
#### Dataset-1

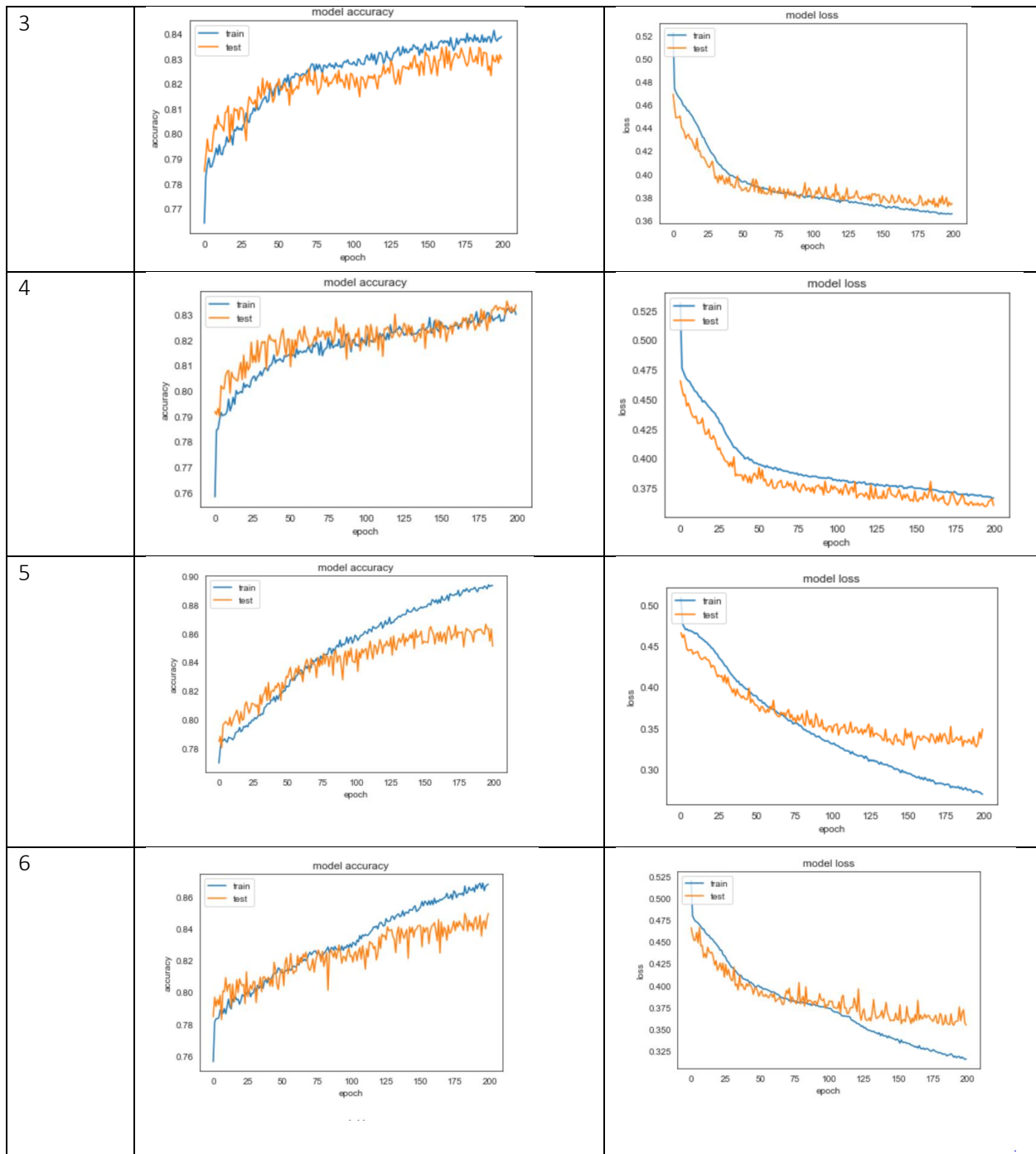
<u>Experment Number</u>	<u>Activation function</u>	<u>No of hidden layers</u>	<u>Neurons in each hidden layer</u>	<u>Confusion matrix</u>	<u>Accuracy achieved</u>	<u>Epochs</u>
1	Relu	2	6	[[3977 399] [ 687 858]]	81.65%	200
2	Relu	3	6	[[3916 460] [ 560 985]]	82.77%	1000
3	tanh	3	6	[[3935 441] [ 556 989]]	83.16%	200
4	tanh	5	6	[[3989 387] [ 600 945]]	83.33%	200
5	tanh	5	14	[[4048 328] [ 499 1046]]	86.03%	200
6	tanh	10	14	[[3959 417] [ 470 1075]]	85.01%	200

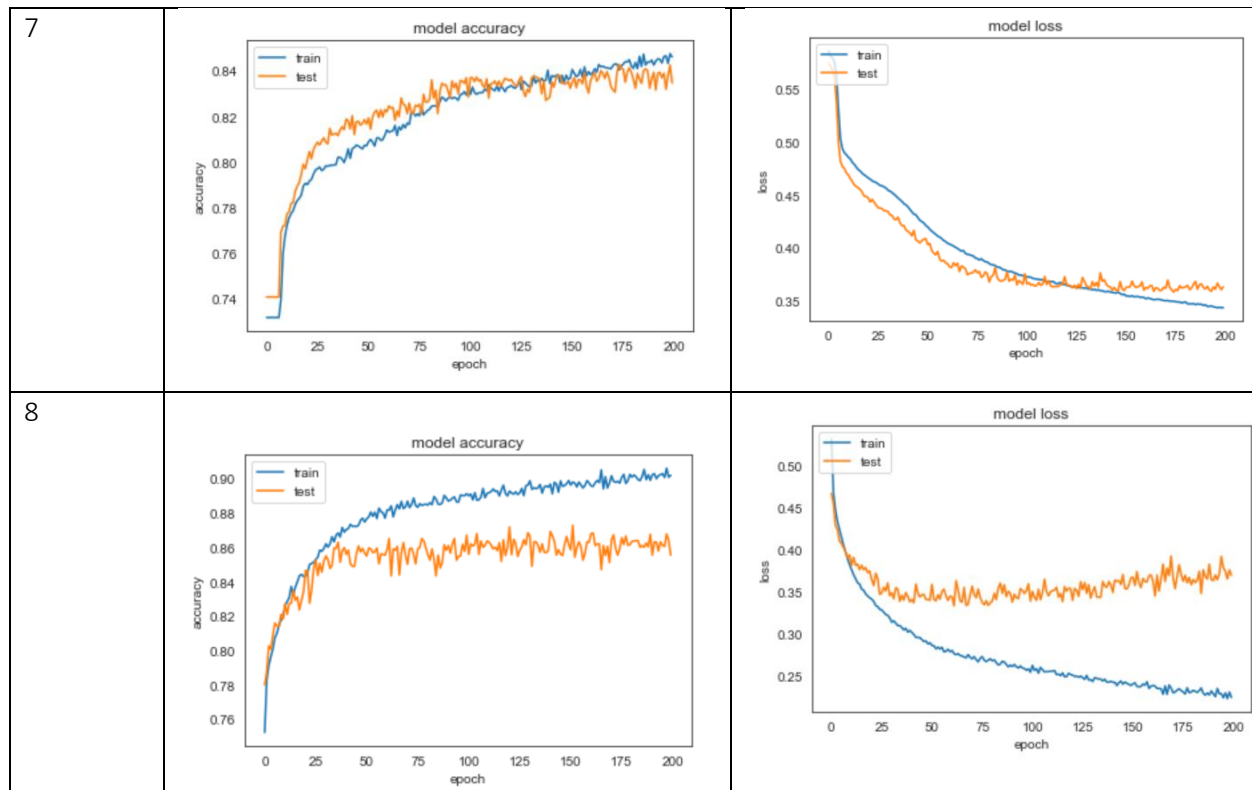
7	sigmoid	5	14	$\begin{bmatrix} 3951 & 425 \\ 525 & 1020 \end{bmatrix}$	83.95%	200
8	Relu	5	14	$\begin{bmatrix} 3917 & 459 \\ 416 & 1129 \end{bmatrix}$	85.22	200

The best result achieved is with tanh activation function and number of neurons equal to 14 in each of the hidden layers. This shows that playing around with the number of neurons more we could have tuned the neural network for more accuracy but there would be a trade off here too, as it would take longer to execute and would be a load on the computer.

### Graphs

Experiment number	Model accuracy vs epoch	Model loss vs epoch
1		
2		





The Orange curve is for the Validation set, I have partitioned the training data into validation set and training set with a split of 30% for validation set and 70% for training set . From Experiment 2 where I have executed 1000 epochs , we see that validation error is not changing much after 200 epochs, that is why I have executed 200 epochs in all the other experiments. This is can be observed in the loss function graph as well.

## Dataset-2

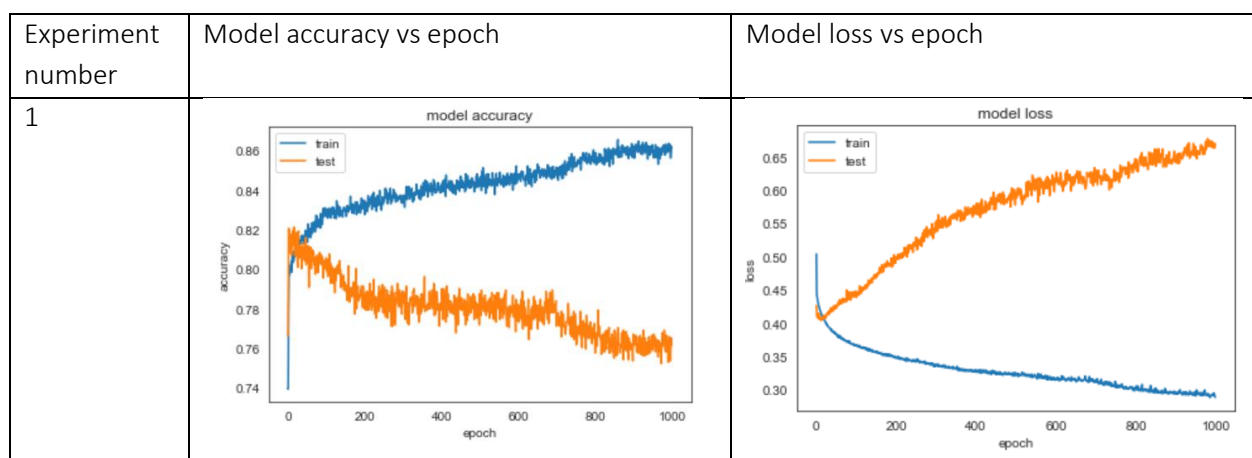
<u>Activation function</u>	<u>No of hidden layers</u>	<u>Neurons in each hidden layer</u>	<u>Confusion matrix</u>	<u>Accuracy achieved</u>	<u>Epochs</u>
Relu	2	14	[[1232 252] [ 304 322]]	73.64	1000
tanh	3	14	[[1303 181] [ 340 286]]	75.3	200
tanh	5	14	[[1218 266] [ 276 350]]	74.31	200
tanh	10	14	[[1182 302] [ 237 389]]	74.45	200
relu	10	14	[[1297 187] [ 309 317]]	76.49	200

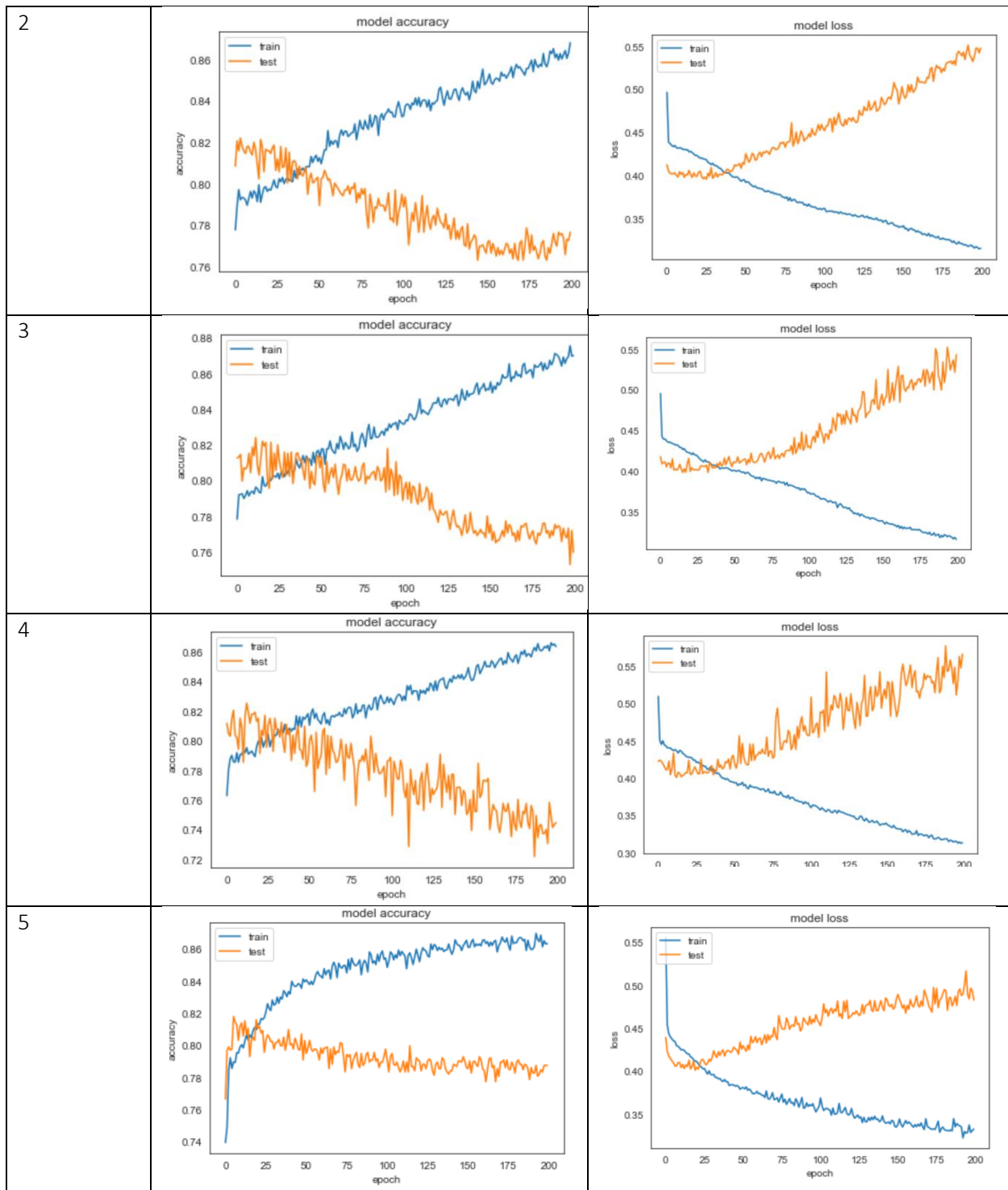
relu	10	50	$\begin{bmatrix} 1260 & 224 \\ 335 & 291 \end{bmatrix}$	73.5	200
tanh	10	50	$\begin{bmatrix} 1193 & 291 \\ 229 & 397 \end{bmatrix}$	75.35	200
tanh	10	100	$\begin{bmatrix} 1484 & 0 \\ 626 & 0 \end{bmatrix}$	70.33	200

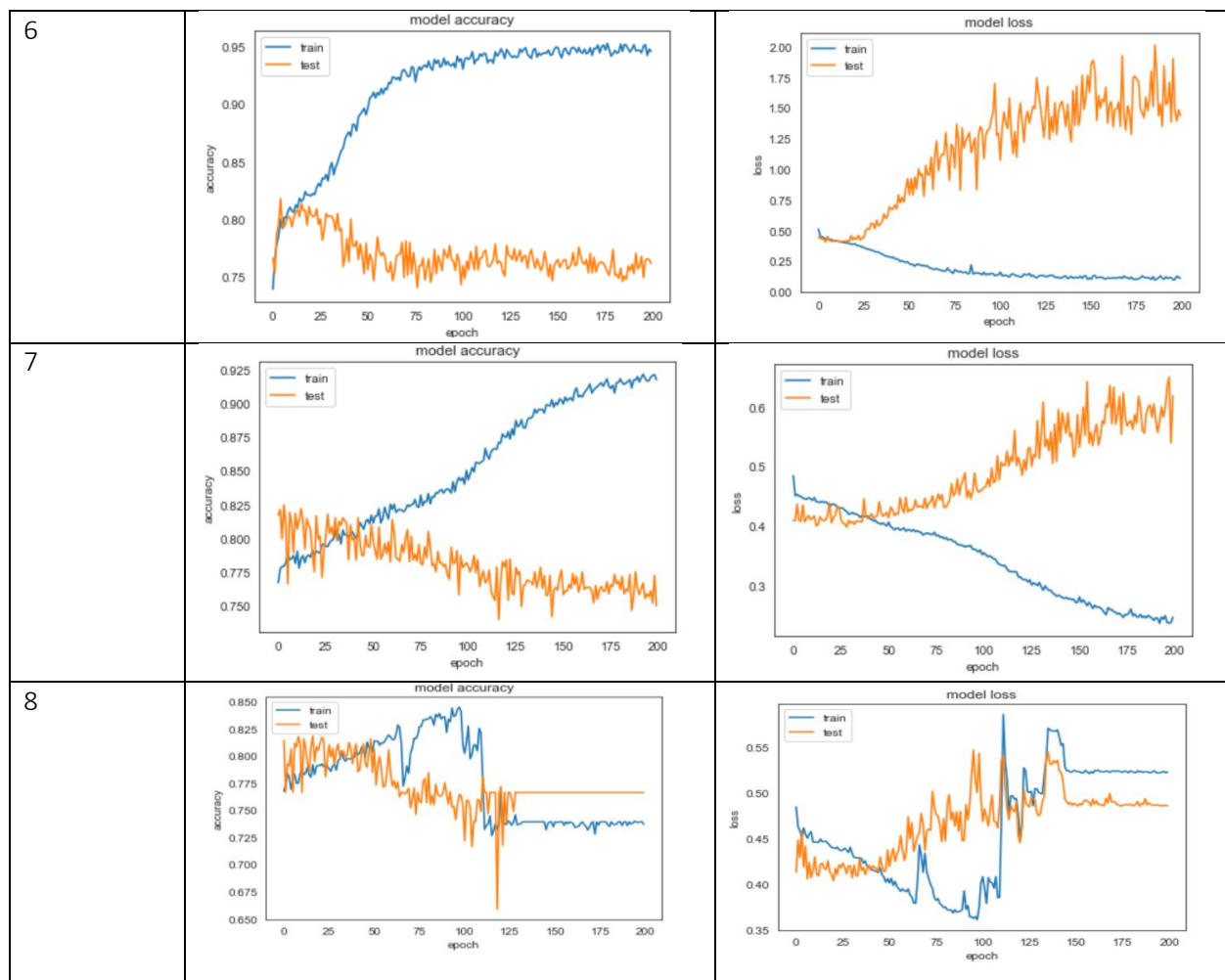
The best result achieved is with relu activation function and number of neurons equal to 14 in each of the hidden layers. This shows that playing around with the number of neurons more we could have tuned the neural network for more accuracy but there would be a trade off here too, as it would take longer to execute and would be a load on the computer. And also we can note that when the number of neurons is made 100(A very large value) The neural net acts funny and there are no **False predictions** and the accuracy falls down. This shows that the number of neurons in each of the hidden layer must be lesser than 100.

From the graphs below we see that for when I have executed for 1000 epochs the models accuracy is going down consistently and training set accuracy is going up, This shows a lot of variance in the model.

### Graphs







All the ANN models have high variance but with accuracy around 75% we cannot say it has high bias, the gap between the validation accuracy rate and training accuracy rate is proof for high variance and each time the model is supplied with a new set of data the accuracy is coming down

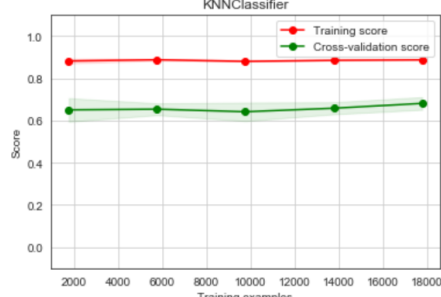
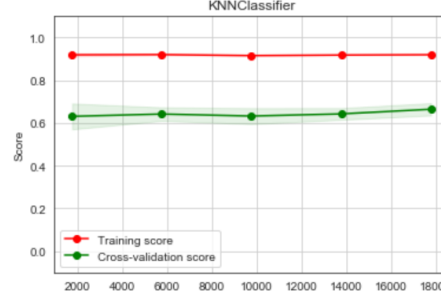
### KNN Classification

2. Download and use any KNN package to classify your classification problems. Experiment with number of neighbors. You can use any distance metric appropriate to your problem. Just be clear to explain why you used the metric that you used.



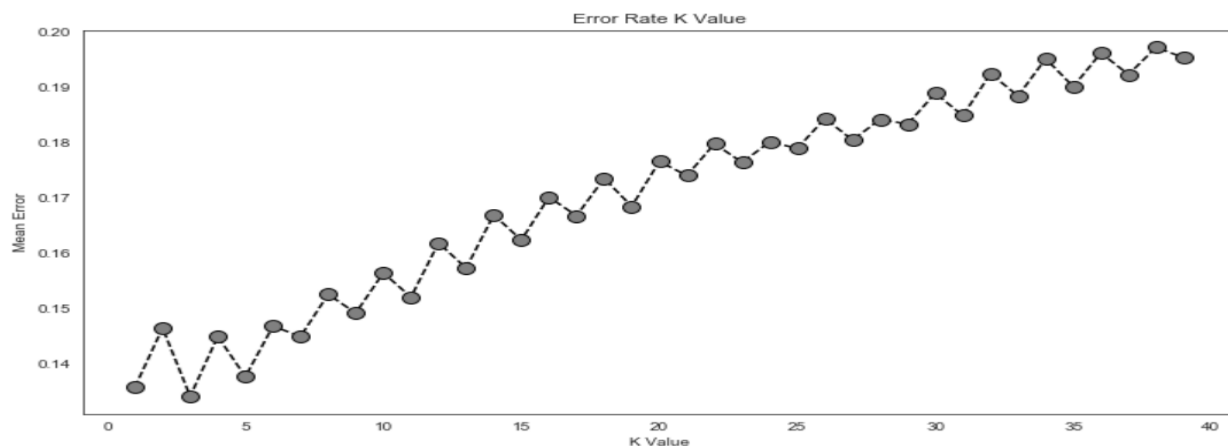
## Dataset:1

## Euclidian Distance

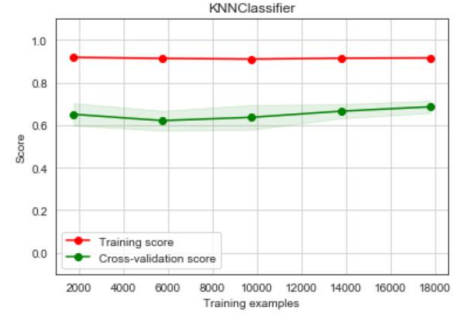
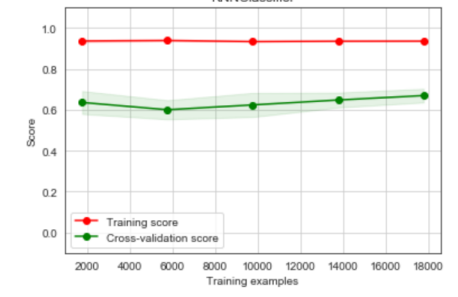
Value of K	Confusion Matrix	Accuracy	Mean Accuracy with CV=10	Standard deviation	Learning curve
5	<code>[[[4100, 276], [ 539, 1006]]]</code>	86.23	85.5	1.16	
3	<code>[[[4058, 318], [ 476, 1069]]]</code>	86.59	85.67	1.24	

To find the best value for K for KNN with Euclidian distance, I have performed grid search and the result obtained for best accuracy and best parameter `0.8567395395975098`  
`{'n_neighbors': 3, 'p': 2}`

This is further proved with the Elbow graph which has K-value on the X-axis and Mean error on the Y-axis



## Manhattan Distance

Value of K	Confusion Matrix	Accuracy	Mean Accuracy with CV=10	Standard deviation	Learning curve
5	[[4119, 257], [ 423, 1122]]	88.51	87.91	1.26	
3	[[4090, 286], [ 388, 1157]]	88.61	88.04	1.02	

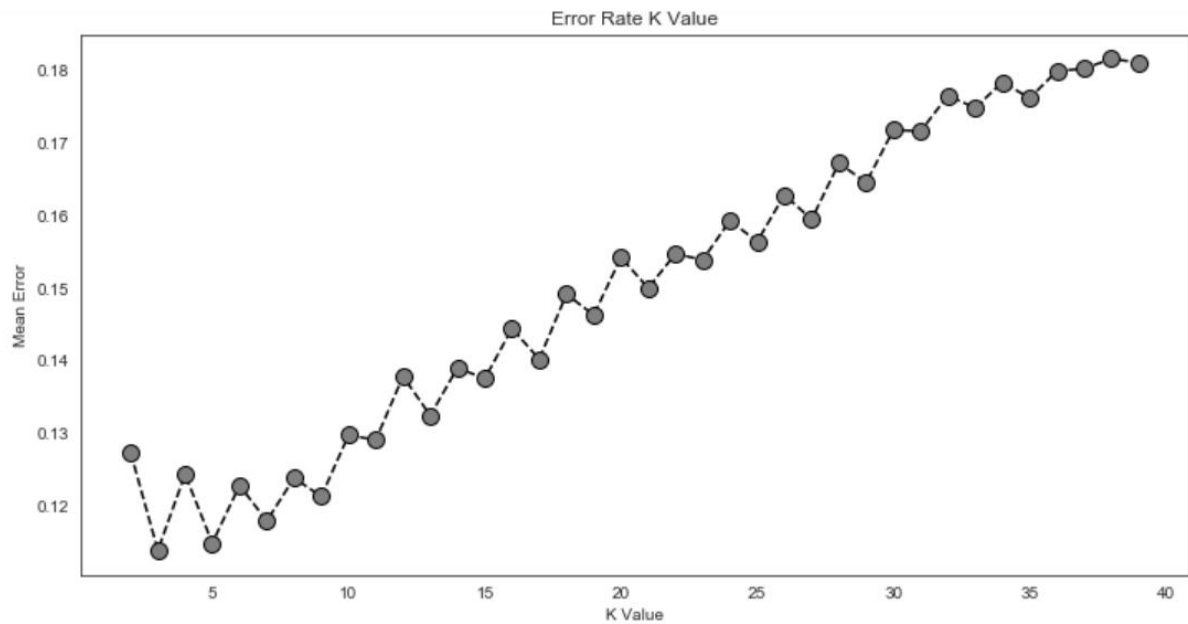
To find the best value for K for KNN with Manhattan distance, I have performed grid search and the result obtained for best accuracy and best parameter

```
0.880483567395396
{'n_neighbors': 3, 'p': 1}
```

The best accuracy achievable is 88% with K=3 and Manhattan distance

The constant gap in the learning curve between the training score and cross-validation score tells us that this model is good and it has neither high variance nor low bias, As the curve is not changing each time the model is supplied with new data this proves the low variance and the accuracy around 85% to 90% shows us high bias. This is for Euclidian distance and Manhattan distance curves

This is further proved with the Elbow graph which has K-value on the X-axis and Mean error on the Y-axis



I have already provided the result for K=3

## Dataset:2

### Euclidian Distance

Value of K	Confusion Matrix	Accuracy	Mean Accuracy with CV=10	Standard deviation	Learning curve
5	[[1271, 213], [ 342, 284]]	73.69	76.77	1.49	
3	[[1242, 242], [ 346, 280]]	72.13	75.03	1.39	

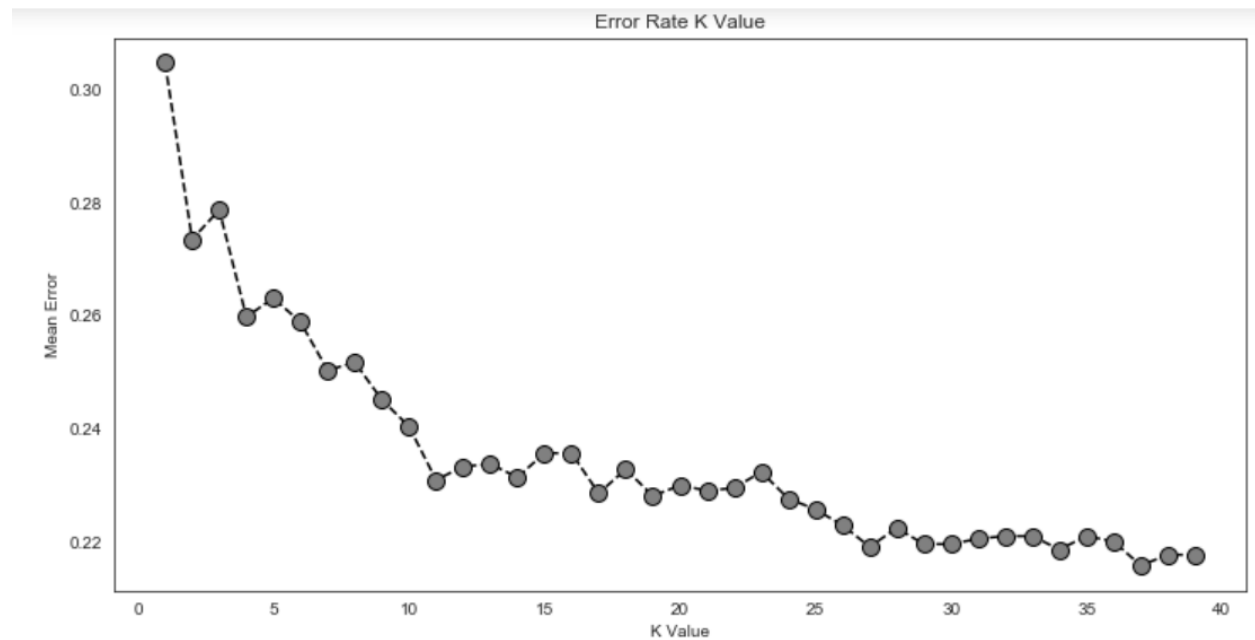
To find the best value for K for KNN with Euclidian distance, I have performed grid search and the result obtained for best accuracy and best parameter

**0.7923608289313288**

**{'n\_neighbors': 38, 'p': 2}**

The best possible accuracy achievable is 79.23% with k=38 and Euclidian Distance

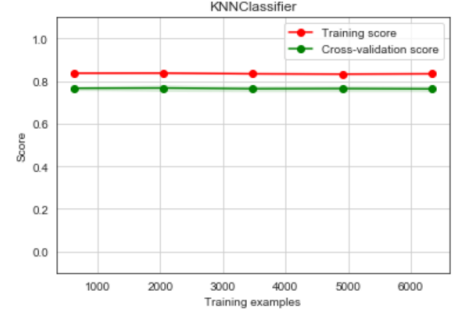
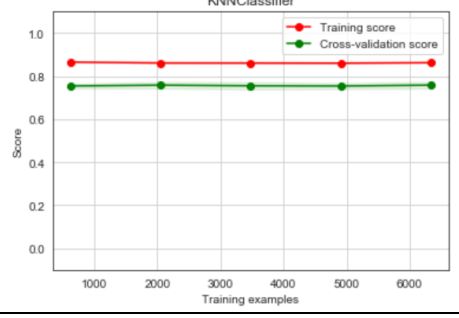
This is further proved with the Elbow graph which has K-value on the X-axis and Mean error on the Y-axis



Result for k=38

38	<pre>[[1326, 158],  [ 301, 325]].</pre>	78.24	79.23	1.48	<p>The graph, titled 'KNNClassifier', plots Score (Y-axis, ranging from 0.0 to 1.0) against Training examples (X-axis, ranging from 0 to 6000). The Training score (red line with dots) and Cross-validation score (green line with dots) are both high and stable, around 0.8, indicating good performance and generalization.</p> <table><tr><th>Training examples</th><th>Training score</th><th>Cross-validation score</th></tr><tr><td>1000</td><td>0.80</td><td>0.78</td></tr><tr><td>2000</td><td>0.80</td><td>0.78</td></tr><tr><td>3000</td><td>0.80</td><td>0.78</td></tr><tr><td>4000</td><td>0.80</td><td>0.78</td></tr><tr><td>5000</td><td>0.80</td><td>0.78</td></tr><tr><td>6000</td><td>0.80</td><td>0.78</td></tr></table>	Training examples	Training score	Cross-validation score	1000	0.80	0.78	2000	0.80	0.78	3000	0.80	0.78	4000	0.80	0.78	5000	0.80	0.78	6000	0.80	0.78
Training examples	Training score	Cross-validation score																								
1000	0.80	0.78																								
2000	0.80	0.78																								
3000	0.80	0.78																								
4000	0.80	0.78																								
5000	0.80	0.78																								
6000	0.80	0.78																								

## Manhattan Distance

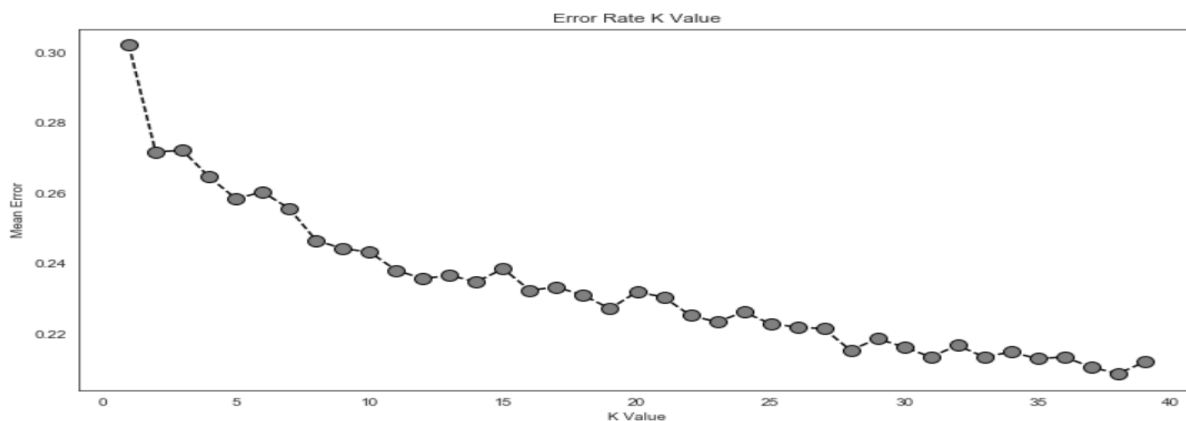
Value of K	Confusion Matrix	Accuracy	Mean Accuracy with CV=10	Standard deviation	Learning curve
5	$\begin{bmatrix} 1269 & 215 \\ 330 & 296 \end{bmatrix}$	74.17	76.77	1.51	
3	$\begin{bmatrix} 1252 & 232 \\ 342 & 284 \end{bmatrix}$	72.79	75.33	0.89	

To find the best value for K for KNN with Manhattan distance, I have performed grid search and the result obtained for best accuracy and best parameter

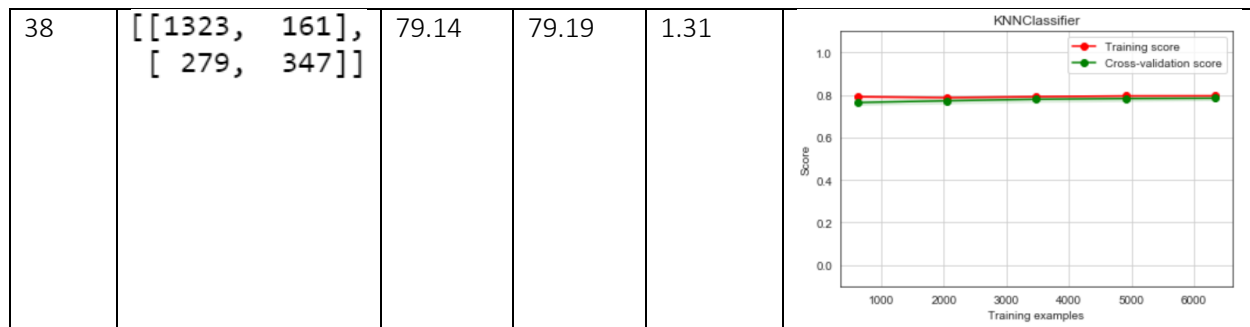
`0.7919544900446973`  
`{'n_neighbors': 38, 'p': 1}`

The best accuracy achievable is 79.19% with K=38 and Manhattan distance

This is further proved with the Elbow graph which has K-value on the X-axis and Mean error on the Y-axis



Result for k=38 with Manhattan Distance



The constant gap in the learning curve between the training score and cross-validation score tells us that this model is good and it has neither high variance nor low bias, As the curve is not changing each time the model is supplied with new data this proves the low variance and the accuracy around 85% to 90% shows us high bias. This is for Euclidian distance and Manhattan distance curves.

- Comparisons of the two learning algorithms using the two data sets.

For the Energy dataset which is the first dataset the one of the ANN classification with tanH activation as performed better then KNN algorithm with a very small margin, but KNN would be preferred as it is easier to explain and faster in execution. But KNN with Manhattan distance with k=3 has accuracy of 88.04 which is the best, better tuning on the ANN would have made it better then the KNN

For the Telecom Churn dataset which is the second dataset, surprisingly the KNN algorithm with K=38 with Euclidian distance has performed best, this is due to the Grid search done, If better hyperparameter tuning was done for the ANN then it would have achieved better accuracy

- Does any of these algorithms perform better than the three algorithms used in assignment 2?  
What is your observation about how these 5 algorithms performed on the two data sets

For the Energy dataset which is the first dataset the one of the ANN classification with tanH activation as performed better then KNN algorithm with a very small margin, but KNN would be preferred as it is easier to explain and faster in execution. But KNN with Manhattan distance with k=3 has accuracy of 88.04 which is the best, better tuning on the ANN would have made it better then the KNN. But Boosting has come close with around 80%. In real time all situations should be considered and like computing power and speed of execution before choosing the algorithm.

For the Telecom Churn dataset which is the second dataset, , Here also Boosting comes close with around 80% accuracy. If better hyperparameter tuning was done for the ANN then it would have achieved better accuracy, Here also Boosting comes close with around 80% accuracy