Applied Machine Learning
Assignment 3

Sai Vishnu Kanisetty SXK175300

Objective of this exercise:
Experimentation of classification algorithms (Neural Networks and KNN) for two problems:

- Predict whether the articles published by Mashable will be shared more number of times or less number of times
- Predict the Success of Bank Telemarketing

Index: This exercise has been implemented in Python. Different topics discussed in this report are:

1. Definition for different algorithms used and their hyper parameters
2. Data understanding for both the datasets
3. Results of experimentation for Mashable problem
4. Results of experimentation for Telemarketing
5. Comparison of 5 different algorithms for both the data sets
6. Next Steps

Step 1: Definition for different algorithms used and their hyper parameters

For both the datasets, maximizing the cross validation has been considered as the metric.

Neural Networks:

- All the hyperparameters listed below have been used for experimented in two different architectures
- The first architecture has 2 hidden layers and one output layer, whereas the second architecture has 3 hidden layers and one output layer
- Loss function used in both the architectures is 'binary cross entropy'
- Different hyperparameters experimented:
    - Number of hidden units in two hidden layers architecture:
        - Set of the first and second hidden layer units experimented: [30,15], [20,10]
    - Number of hidden units in three hidden layers architecture:
        - Set of first, second and third hidden layer units experimented: [30,15,8], [20,10,5]
    - Both architectures:
        - Different hidden layer activation functions experimented: ['relu','sigmoid']
        - Output layer activation function used: sigmoid
        - Different epochs experimented: [10,20,30]
        - Different mini-batch sizes experimented: [32,64]
        - Different optimizers experimented: ['adam', 'rmsprop']

K Nearest Neighbors:

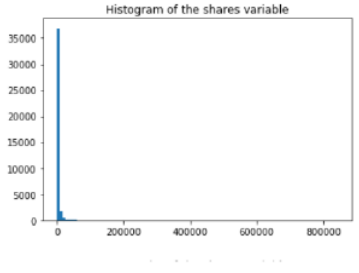
- Hyperparameters [2]:
    - **Neighbors:** Range of number of values experimented [1,5,15,25,49]. Since both the datasets are binary classification problems, only odd values of neighbors are used in experimented, to avoid the ties

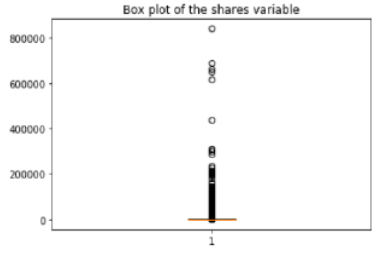
- **Weights:**
  - uniform: uniform weights. All points in each neighborhood are weighted equally
  - distance: weight points by the inverse of their distance.
- **Algorithm:**
  - Ball_Tree: A ball tree recursively divides the data into nodes defined by a centroid C and radius r, such that each point in the node lies within the hyper-sphere defined by r and C. The number of candidate points for a neighbor search is reduced through use of the *triangle inequality*[1]
  - Brute: Brute force search
- **P (distance measure):**
  - 1: Minkowski distance
  - 2: Euclidean distance

Step 2: Data Understanding for both the datasets

Mashable Dataset:

Data has 39,644 records of 61 variables and there are no missing values. Two non-predictive features (URL and Timedelta) have been dropped from further analysis. All the numerical predictor features have been normalized. Some information about the shares variable: Mean being more than the median and having values above the 2 standard deviations (box plot) shows that the data is positively skewed

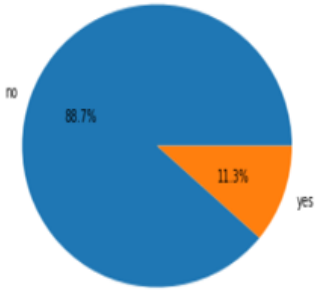




Median of the shares (1400) has been considered cutoff for the classification. The split of the articles to high shares (20,082) and low shares (19,562) will be ~50% because of considering median as the split point.

Telemarketing Data:

Data published in UCI machine learning repository has been used [2] .Data has 20 variables that are likely to describe whether a customer has subscribed to the product in telemarketing or not. All the categorical variables have been converted using one-hot encoding. All the numerical variables have been normalized. Only 11.3% of the customers have prescribed the product.



In both the data sets, data has been randomly divided into 70% train and 30% test.

Step 3: Results of experimentation for Mashable problem

All possible combinations of hyperparameters mentioned in the Step 1 have been used in experimentation. Results mentioned below are the best 5-fold cross-validation accuracy observed in different architectures mentioned in step 1.

Neural Networks:

Experimentation:

- **Two hidden layers:** cv accuracy: 65.93%, Train acc: 67.6%, Test acc: 65.5%, mean time: 25 s

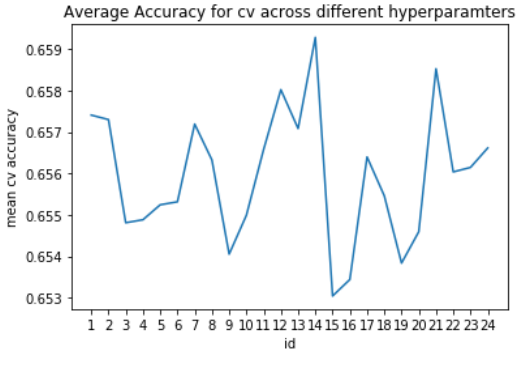
```
Maximum cv accuracy : 0.6593
Estimator with maximum cv accuracy : {'batch_size': 64, 'epochs': 10, 'first
_hidden_units': 20, 'hidden_activation_fun': 'relu', 'optimizer': 'rmsprop',
'output_activation_func': 'sigmoid', 'second_hidden_units': 10}
```

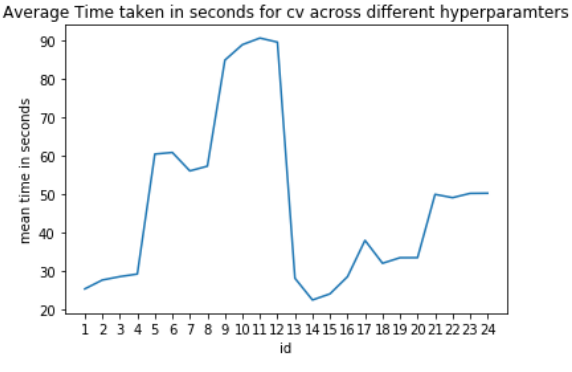
- **Three hidden layers:** cv accuracy: 65.94%, Train acc: 66.8%, Test acc: 65.2%, mean time: 105 s

```
Maximum cv accuracy : 0.6594
Estimator with maximum cv accuracy : {'batch_size': 32, 'epochs': 30, 'first
_hidden_units': 30, 'hidden_activation_fun': 'sigmoid', 'optimizer': 'adam',
'output_activation_func': 'sigmoid', 'second_hidden_units': 15, 'third_hidde
n_units': 8}
```

Summary:

- Between the two architectures, minimal difference has been observed in cv accuracy.
- Considering the execution time, cv accuracy and test accuracy, **Two hidden layers** has been considered as the best methodology. Test accuracy is very near to cv accuracy, which shows that the algorithm is generalizing well with the hyperparameters used
- Findings from experimentation with other hyperparameters using two hidden layers architecture: (Here id represents the combination of hyperparameters used in each experimentation step). (Due to space constraints, id related to relevant findings have been mentioned below)





1. Highest accruay and the least accuracy has been identified for experimentiaon id: 14 and 15, where as only minimal difference has been observed in the execution time. The only difference between the 14 and 15 experimentation id is the hidden layer activation funciton used. For id:14, relu has been usedd and id:15, sigmoind has been used

```
id: 14 | hyperparamter: {'batch_size': 64, 'epochs': 10, 'first_h
idden_units': 20, 'hidden_activation_fun': 'relu', 'optimizer': '
rmsprop', 'output_activation_func': 'sigmoid', 'second_hidden_uni
ts': 10}
```

```
id: 15 | hyperparamter: {'batch_size': 64, 'epochs': 10, 'first_hidden_units': 20, 'hidden_activation_fun': 'sigmoid', 'optimizer': 'adam', 'output_activation_func': 'sigmoid', 'second_hidden_units': 10}
```

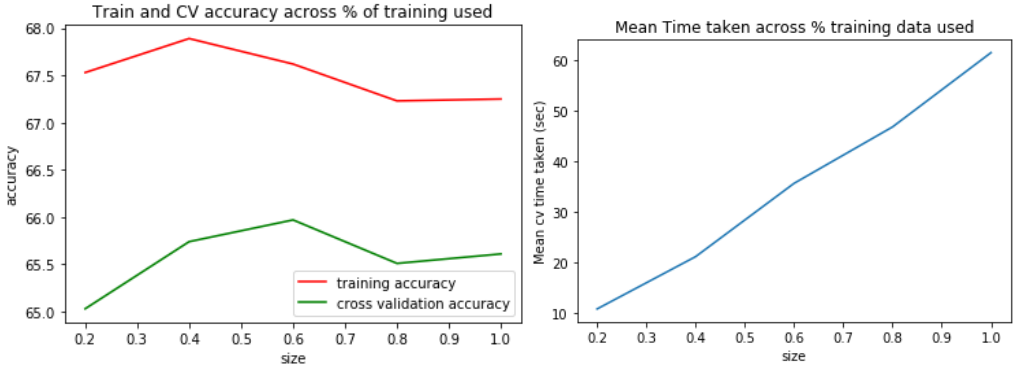
2. Highest time for execution is associaed with id: 12, which has the maximum number of epochs i.e 30.

```
id: 12 | hyperparamter: {'batch_size': 32, 'epochs': 30, 'first_hidden_units': 20, 'hidden_activation_fun': 'sigmoid', 'optimizer': 'rmsprop', 'output_activation_func': 'sigmoid', 'second_hidden_units': 10}
```

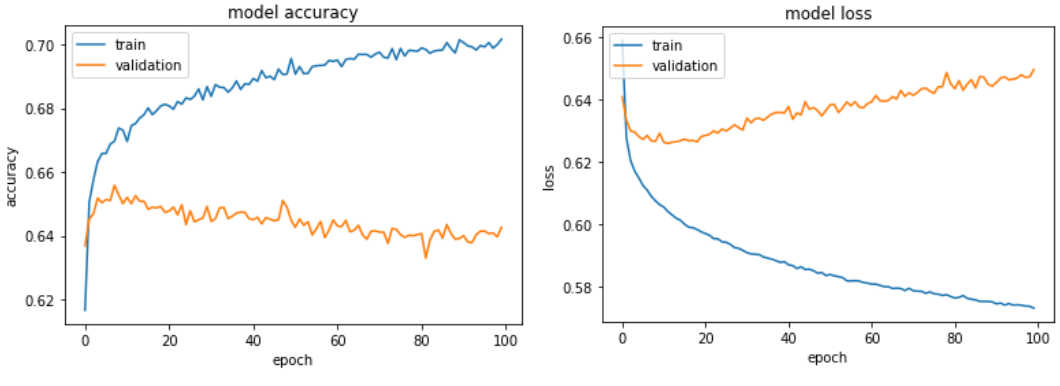
- Confusion matrix for the test data:

```
array([[4115, 2168],
       [1953, 3658]])
```

- Varying the train data size using the best hyperparameters obseverved, shows that the cv accuracy obtained using only 60% of train data is approximatley equal to using 100% of train data and needless to say, execution time using 60% of the data is ~60% of the execution time using whole data



- Experimenting with more number of epochs for the best hyperparameters identified, shows that when epochs=10 has the highest validation accuracy and the validation accuracy decreases post that and train accuracy increases, indicating over fitting.



KNN:

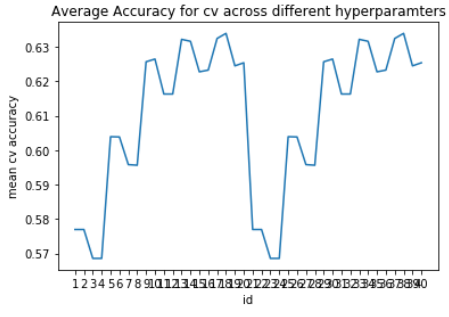
- Maximum cross validation accuracy of 63.39% has been observed using the below mentioned hyperparameters. Though the train accuracy is 100%, test accuracy being close to cv accuracy, shows that the algorithm is generalizing well with the hyperparameters used.

```
Maximum cv accuracy : 0.6339
Estimator with maximum cv accuracy : {'algorithm': 'ball_tree', 'n_neighbor
s': 49, 'p': 1, 'weights': 'distance'}

Train accuracy score of best model: 1.0
Test accuracy score of best model: 0.6333
```

- Findings from the experimentation shows that the least cv accuracy has been observed for id:4 and the best for id 18. The key difference in both the experimentation is the number of neighbors used.

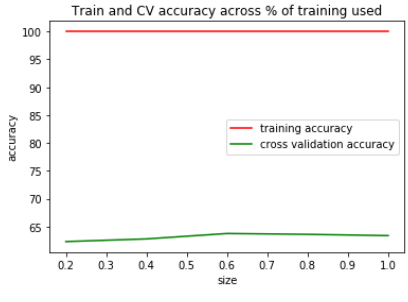
```
id: 4 | hyperparamter: {'algorithm': 'ball_tree', 'n_neighbors': 1,
'p': 2, 'weights': 'distance'}
id: 18 | hyperparamter: {'algorithm': 'ball_tree', 'n_neighbors': 49
, 'p': 1, 'weights': 'distance'}
```



- Confusion matrix for the test data:

```
array([[4389, 2682],
       [1679, 3144]])
```

- Varying the train data size using the best hypermaters identified, shows that the cv accuracy using 20% of the data is approximately same as the cv accuracy with 100% of the training data.



Step 4: Results of experimentation for Telemarketing problem

All possible combinations of hyperparameters mentioned in the Step 1 have been used in experimentation. Results mentioned below are the best 5-fold cross-validation accuracy observed in different architectures mentioned in step 1.

Neural Networks:

Experimentation:

- **Two hidden layers:** cv accuracy: 91.26%, Train acc: 91.8%, Test acc: 91.7%, mean time: 7 s

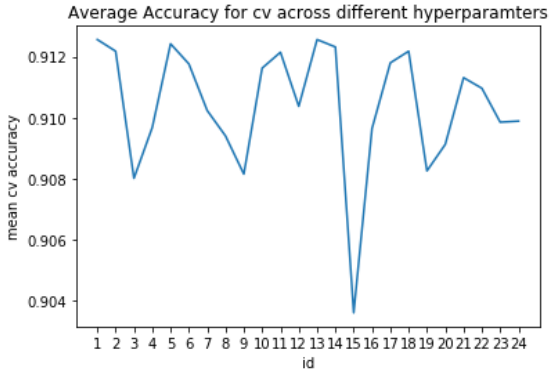
```
Maximum cv accuracy : 0.9126
Estimator with maximum cv accuracy : {'batch_size': 32, 'epochs': 10, 'first
_hidden_units': 30, 'hidden_activation_fun': 'relu', 'optimizer': 'adam', 'o
utput_activation_func': 'sigmoid', 'second_hidden_units': 15}
```

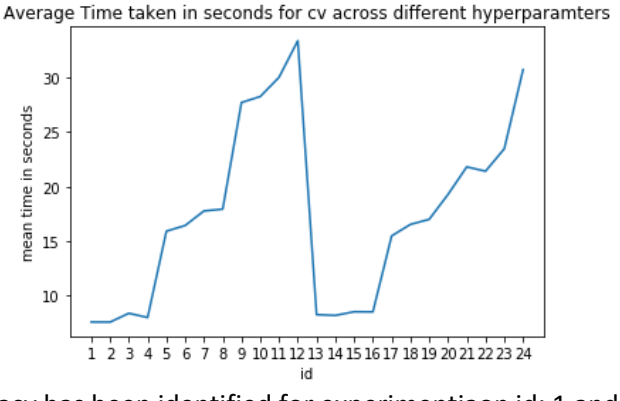
- **Three hidden layers:** cv accuracy: 91.22%, Train acc: 91.66%, Test acc: 91.5%, mean time: 125 s

```
Maximum cv accuracy : 0.9122
Estimator with maximum cv accuracy : {'batch_size': 32, 'epochs': 30, 'first
_hidden_units': 30, 'hidden_activation_fun': 'sigmoid', 'optimizer': 'adam',
'output_activation_func': 'sigmoid', 'second_hidden_units': 15, 'third_hidde
n_units': 8}
```

Summary:

- Between the two architectures, minimal difference has been observed in cv accuracy.
- Considering the execution time and cv accuracy, **Two hidden layers** has been considered as the best methodology. Test accuracy is very near to cv accuracy, which shows that the algorithm is generalizing well with the hyperparameters used.
- Findings from experimentation with other hyperparameters using two hidden layers architecture: (Here id represents the combination of hyperparameters used in each experimentation step). (Due to space constraints, id related to relevant findings have been mentioned below)





1. Highest accruay and the least accuracy has been identified for experimentiaon id: 1 and 15, where as only minimal difference has been observed in the execution time. The only differe nce between the 1 and 15 experimentation id is the hidden layer activation funciton and mi ni batch size used.

```
id: 1 | hyperparamter: {'batch_size': 32, 'epochs': 10, 'first_hid
den_units': 30, 'hidden_activation_fun': 'relu', 'optimizer': 'adam'
, 'output_activation_func': 'sigmoid', 'second_hidden_units': 15}
```

```
id: 15 | hyperparamter: {'batch_size': 64, 'epochs': 10, 'first_hid
den_units': 30, 'hidden_activation_fun': 'sigmoid', 'optimizer': 'a
dam', 'output_activation_func': 'sigmoid', 'second_hidden_units': 1
5}
```

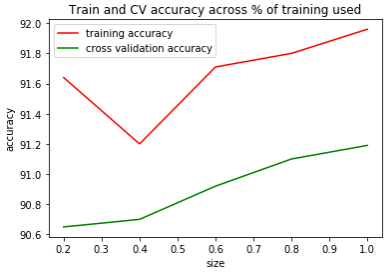
2. Highest time for execution is associaed with id: 12, which has the maximum number of epochs i.e 30.

```
id: 12 | hyperparamter: {'batch_size': 32, 'epochs': 30, 'first_hidd
en_units': 30, 'hidden_activation_fun': 'sigmoid', 'optimizer': 'rms
prop', 'output_activation_func': 'sigmoid', 'second_hidden_units': 1
5}
```

- Confusion matrix for the test data:

```
array([[10571,   617],
       [  397,   772]])
```

- Varying the train data size using the best hypermaters identified, shows that the cv accuracy improves along with the percentage of training data used.



KNN:

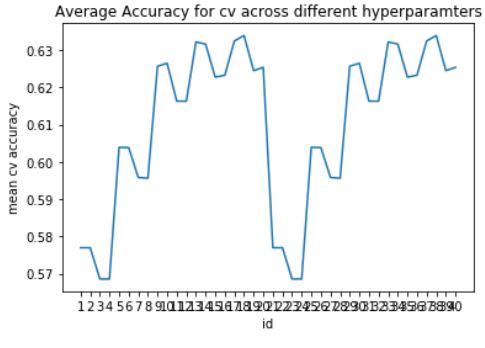
- Maximum cross validation accuracy of 90.65% has been observed using the below mentioned hyperparameters. Though the train accuracy is 100%, test accuracy being close to cv accuracy, shows that the algorithm is generalizing well with the hyperparameters used.

```
Maximum cv accuracy : 0.9065
Estimator with maximum cv accuracy : {'algorithm': 'ball_tree', 'n_neighbor
s': 25, 'p': 2, 'weights': 'distance'}

Train accuracy score of best model: 1.0
Test accuracy score of best model: 0.9087
```

- Findings from the experimentation shows that the least cv accuracy has been observed for id:1 and the best for id 16. The difference in both the experimentation being the number of of neighbors used, weights and the distance measure.

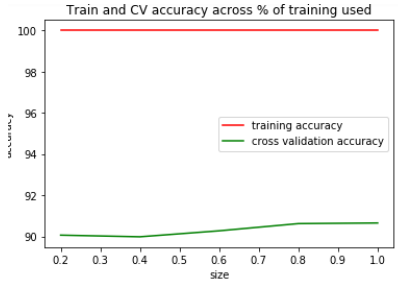
```
id: 1 | hyperparamter: {'algorithm': 'ball_tree', 'n_neighbors': 1,
'p': 1, 'weights': 'uniform'}
 id: 16 | hyperparamter: {'algorithm': 'ball_tree', 'n_neighbors': 2
5, 'p': 2, 'weights': 'distance'}
```



- Confusion matrix for the test data:

```
array([[10697,   857],
       [  271,   532]])
```

- Varying the train data size using the best hypermaters identified, shows that the cv accuracy improves along with the amount of data used.



Summary:

- **NN:** For both datasets, using two hidden layers and 'relu' as an activation for hidden layers resulted in highest cv accuracy
- **KNN**: For both datasets, 'ball tree' algorithm and 'distance' as weights resulted in the highest cv accuracy

Step 5: Comparison of 5 different algorithms for both the data sets

Different algorithms are ranked based on their cross-validation accuracy. All the algorithms used 70% as training data and 30% as test data.

Mashable dataset: Adaboost has the maximum cross validation accuracy

Algorithm	CV accuracy	Rank
SVM	64%	4
Decision Tree	64.3%	3
Adaboost	66.5%	1
ANN	65.9%	2
KNN	63.4%	5

Telemarketing dataset: Decision trees, adaboost, ANN have approximately same cross validation accuracy.

Algorithm	CV accuracy	Rank
SVM	88%	3
Decision Tree	91.3%	1
Adaboost	91.3%	1
ANN	91.3%	1
KNN	90.6%	2

Step 6: Next Step to improve the model accuracy

- Experimenting with dropout and K2 regularization in Neural Networks
- Experimenting with more number of neighbors in KNN

Reference:

1. http://scikit-learn.org/stable/modules/neighbors.html
2. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier\