

A. Introduction

Assignment Number: 4

Date Of Submission: 19th December 2019.

Student ID Number: 180026646

Parts Implemented:

Basic Agent: **Yes**

Intermediate Agent: **Yes**

Advanced Agent: **Yes**

Compiling and running instructions

1. In the main folder directory "A4-Artificial-Neural-Networks-Ticketing Agent", open the terminal and enter:

"source env/bin/activate "

2. Once the virtual environment is activated, you will see "(env)" at the start of the directory. Next navigate to "A4src/", using:

"cd A4src/"

3. After that, you can run the program using:

"python3 A4main.py <Bas|Int|Adv>"

Example:

source env/bin/activate - In main folder

cd A4src/ - In main folder

python3 A4main.py Bas - in A4src

Libraries used:

All libraries were saved in the "env" folder found in the "A4-Artificial-Neural-Networks-Ticketing Agent" zip. Libraries used include: **sklearn, pandas, numpy, matplotlib, sys**

Saved Neural Network from Basic Agent

The neural network is saved here:

"A4-Artificial-Neural-Networks-Ticketing Agent/hidden_unit_models_final/mynetwork_5.joblib"

This was loaded for the intermediate agent

B. Design, Implementation and Evaluation

B.1 Design and Implementation

B.1.1. PEAS Model

- **Performance measure:** The agent's performance would be measured on how correctly it can assign a ticket request to the appropriate response team. The trained model's classification accuracy will be based on samples from both the training and test data sets i.e training set accuracy and test set accuracy.
- **Environment:** The initial training set will form as the environment as that is where the agent gets its information for learning. For the intermediate section, the responses that the agent receives from its users will also form as part of the environment.
- **Actuators:** The agent can make predictions based on the inputs it receives from the training or test data sets. To make predictions, the agent would have learnt from the training set and from the answers it receives from users. It learns by implementing stochastic gradient descent in the parameter space to minimise the mean squared error.
- **Sensors:** The agent can perceive the means squared error after each training cycle to use as feedback for improving its accuracy. Hence, if it classifies a ticket request wrongly it can learn from that event to improve its algorithm.

B.1.2 Neural Network Design (Basic Agent Implementation)

a) Encoding

The training data is encoded using the one-hot encoding method (**see Figure 1**). The reason why I chose this method is that the outputs are categorical values with a finite set of label values. Since there is no ordinal relationship between the categories, one-hot encoding works well. Moreover, we only had 5 classes, hence the drawback of one-hot encoding having too many classes was not applicable in this instance.

...	IdCards	Staff	Students	Response Team
...	No	No	No	Emergencies
...	Yes	Yes	No	Networking
...	No	Yes	No	Credentials
...	Yes	Yes	No	Datawarehouse
...	Yes	Yes	Yes	Equipment

ENCODED

...	IdCards	Staff	Students	Response Team
...	0	0	0	[0,0,1,0,0]
...	1	1	0	[0,0,0,0,1]
...	0	1	0	[1,0,0,0,0]
...	1	1	0	[0,1,0,0,0]
...	1	1	1	[0,0,0,1,0]

Figure 1: One-hot encoding method applied to the tickets.csv data set

b) Hyperparameter Optimisation

Various tests were implemented to find the best combinations of hyperparameters.

i) Learning rate and momentum

For finding the optimal learning rate and momentum combination, grid search was implemented. Grid search implements the “fit” and “score” methods on the training data to test the momentum and learning rate combinations with the following range: [0.001, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. As a result, 100 model combinations were generated. The combination of having learning rate and momentum set at 0.8 and 0.1 respectively had the highest mean test-score of 98% (**see Figure 9**).

Learning rate = 0.8 and Momentum=0.7, also had a 98% accuracy however it had a higher standard deviation.

ii) Hidden Units

For finding the minimum number of hidden units, I used the network with the best learning rate and momentum combination. After that, models with hidden unit sizes ranging from 1 to 10 were trained and scored (**see Table 2**). Interestingly, 4 hidden units appear to be the least amount hidden units need to get a training set and set score of 100%. Despite this, I decided to select the network with 5 hidden units as it had a much lower training set loss and learnt more efficiently.

c) Early Stopping Criteria

During training, tolerance and no. iterations with no change were used as the stopping criteria. It was initially set at 0.001 tol and 5000 no. iterations with no change by default, however, I noticed that setting was too large as there was a surplus of iterations in some cases, making it hard to distinguish the efficiencies of the models. I then changed it to 0.0001 tol and 10 no. iterations with no change, and that had a positive effect on both the accuracy and the learning rate efficiency (**see Figure 3**).

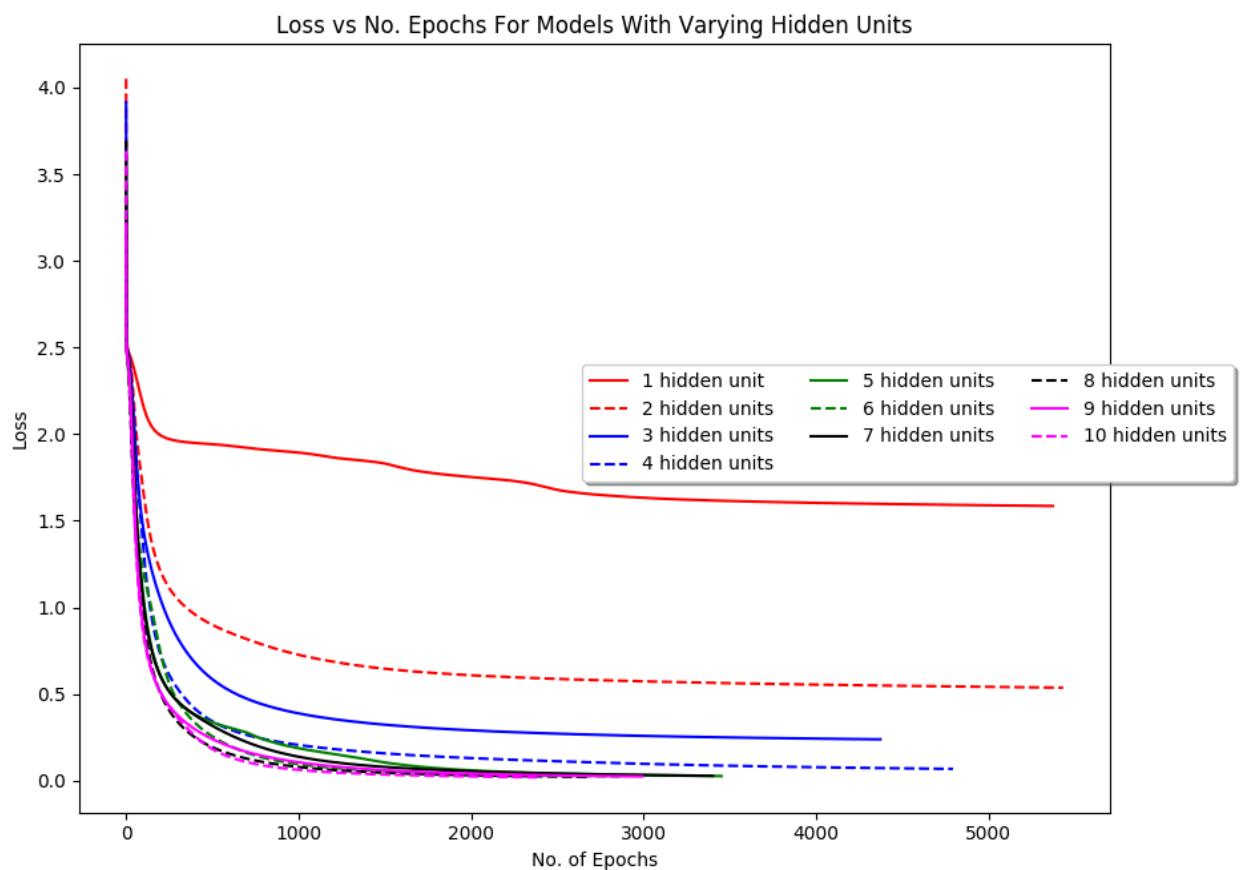


Figure 3: Graph Showing Loss vs No. of Epochs for Models with Varying Hidden units

B.1.3 Ticketing Routing System (Intermediate Agent Implementation)

For the ticketing routing agent, the model with the best combination of hidden units, learning rate & momentum was used. The text-based interface was built using the command line. The user can choose between answering all 9 questions to receive a prediction or answer at **least 3** questions for an early prediction.

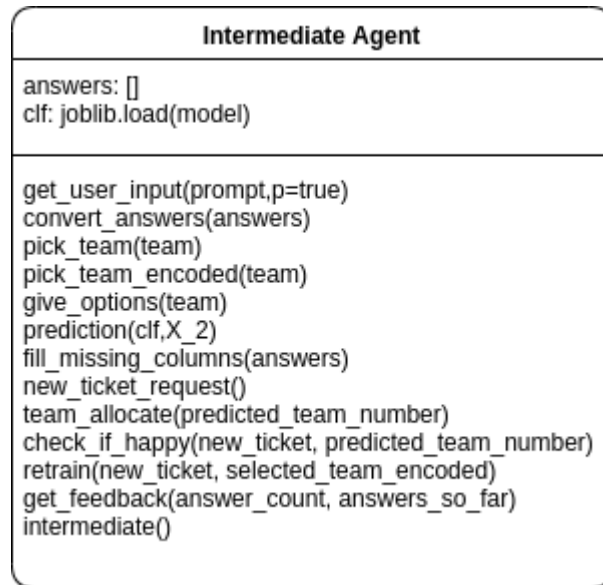


Figure 4: UML diagram showing the main methods and variables used by the intermediate agent

a) Early Prediction

Once a user answers at least 3 questions, a prediction can be made by filling in the remaining values using the mode of the filtered columns (see **Table 2**) of the data set. I chose the filtered mode approach vs the overall mode (unfiltered) because the mode changes based on the answers provided. Moreover, filtering the columns eliminates less likely response teams as seen in **Table 2**, where only Credentials, Datawarehouse and Equipment have that 5-answer input pattern.

					M=Yes	M=No	M=Yes	M=Yes	
Request	Incident	WebServices	Login	Wireless	Printing	IdCards	Staff	Students	Response Team
No	No	No	No	No	Yes	No	No	Yes	Credentials
No	No	No	No	No	No	No	No	Yes	Datawarehouse
No	No	No	No	No	No	No	Yes	Yes	Datawarehouse
No	No	No	No	No	No	No	Yes	Yes	Datawarehouse
No	No	No	No	No	Yes	Yes	Yes	Yes	Equipment
No	No	No	No	No	Yes	No	Yes	No	Equipment
No	No	No	No	No	Yes	Yes	Yes	Yes	Equipment
No	No	No	No	No	Yes	Yes	No	Yes	Equipment
No	No	No	No	No	Yes	No	Yes	No	Equipment
No	No	No	No	No	Yes	No	Yes	No	Equipment

Table 1: A table showing how the data set would be filtered if a user asked for a prediction after answering the first 5 questions. **Green**=filtered columns, **Red**=missing columns, **M**=Mode.

b) Retraining

Retraining occurs when a predicted allocation has been made but the user is not happy with the allocation. Allocations happen after an early prediction or after the user has filled in all 9 answers.

After an early prediction, if the user isn't happy, the system provides a list of available options (omitting the previously predicted team) for the user to choose from. Once the user has chosen their

team of choice, it then asks the user to fill in the rest of the answers. This makes sure that user data is added back to the training table even after an early prediction, which helps improve the system. The full set of answers combined with the user's team of choice is then added to the training table for retraining. The process is the same for when the user has answered 9 questions and isn't happy, except for the additional round of feedback questions.

B.1.4 Different Training Algorithms (Advanced Agent Implementation)

Using scikit-learn I trained five additional classifier algorithms and compared their performance on the test set using the "score" function and by visually analysing them using a confusion matrix. The algorithms were all initialised using the default scikit-learn parameters for consistency.

a) *K*-Nearest Neighbours:

The *K*-nearest neighbours (KNN) classification algorithm is a simple nonparametric procedure for the assignment of a class label to the input pattern based on the class labels represented by the *K*-closest neighbours of the vector (Keller, Gray, & Givens, 1985). The closeness of the neighbours to the vector can be measured using distance functions such as: Manhattan, Euclidean or Minkowski.

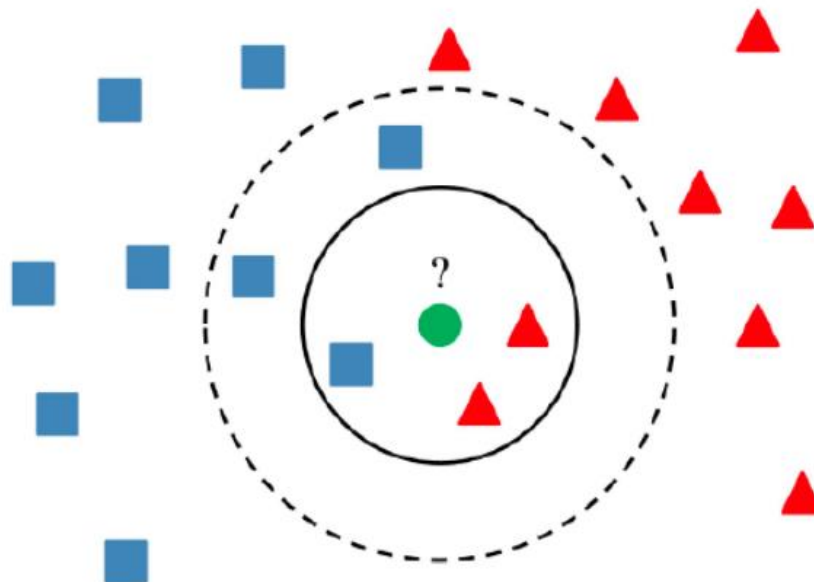


Figure 5: Illustration of the *K*-nearest neighbour algorithm. **Green circle** = sample to be classified. **Blue squares** and **Red triangle** = samples from two different classes in the training set. If *K* is 3, then the 3 nearest neighbours (illustrated by the black circle with the solid line) will decide which class the analysed sample will be assigned. If *K* is 5, the 5 nearest neighbours will be considered (illustrated by the black circle with the dotted line) **Image Source:** (Alaliyat, 2008)

b) Support-Vector Classifier:

Support vector machines (SVM) perform classification in a high-dimensional input space by finding the hyperplane that maximizes the margin between the two classes. The vectors (cases) that define the hyperplane are the support vectors (Suykens & Vandewalle, 1999).

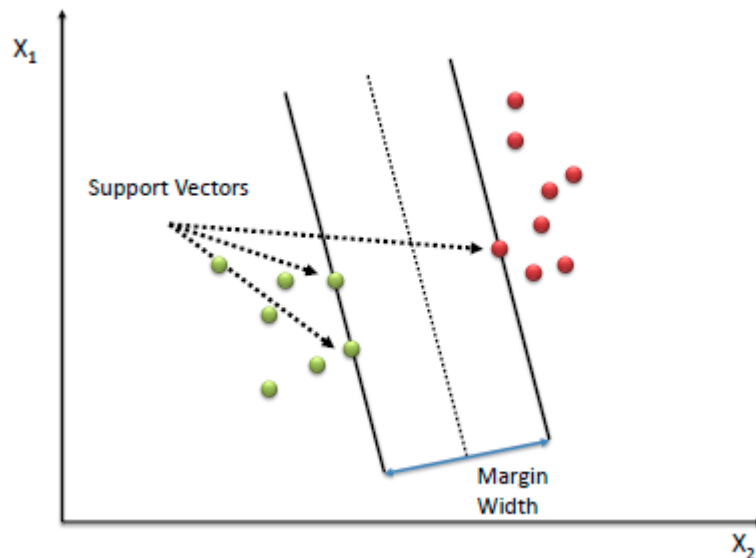


Figure 6: Illustration of classification performed by an SVM. Image Source: (Sayad, 2019b)

c) Naïve Bayesian Classifier:

The Naive Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors. Bayes theorem provides a way of calculating the posterior probability, $P(C|X)$, from $P(C)$, $P(X)$, and $P(X|C)$ (see Figure 7 below). It is called Naïve as it assumes that the effect of the value of a predictor (X) on a given class (C) is independent of the values of other predictors. This assumption is called class conditional independence (Friedman, Geiger, & Goldszmidt, 1997). This can be used for predicting class membership probabilities, such as the probability that a given data item belongs to a class label (Jadhav & Channe, 2013)

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Figure 7: Bayes Theorem. Where:

$P(C|X)$ is the posterior probability of the target class.

$P(C)$ is called the prior probability of class.

$P(X|C)$ is the likelihood which is the probability of predictor of the given class.

$P(X)$ is the prior probability of predictor of class

d) Decision Tree Classifier:

A Decision Tree is a multi-stage classifier with a tree structure that breaks down a dataset into smaller subsets while incrementally developing an associated tree (Swain & Hauska, 1977). The result is a tree with decision nodes and leaf nodes. A decision node (e.g., Weather) has two or more branches (e.g., Sunny, Cloudy and Rainy). The leaf node (e.g., Yes or No) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor is called the root node. This process is recursive and is repeated for every subtree rooted at the new nodes (Sayad, 2019a).

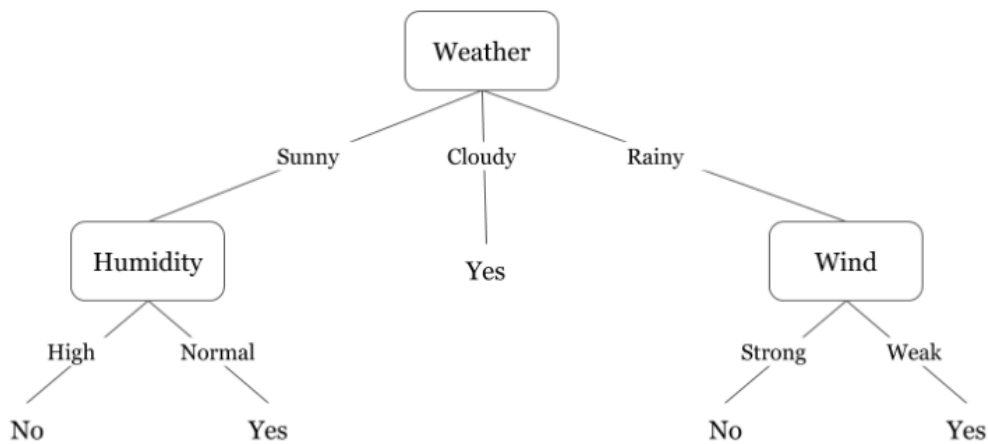


Figure 8: A decision tree for the concept of deciding whether to Play Badminton (Yes/No). Image source: (Gupta, 2019)

B.2 Evaluation

a) Basic Agent

i) Performance by Learning Rate and Momentum

The agent's test score with 100 different combinations of learning rate and momentum was first used to evaluate the performance of the models. Each of the 100 models was trained 10 times separately and the highest mean score achieved was 98%.

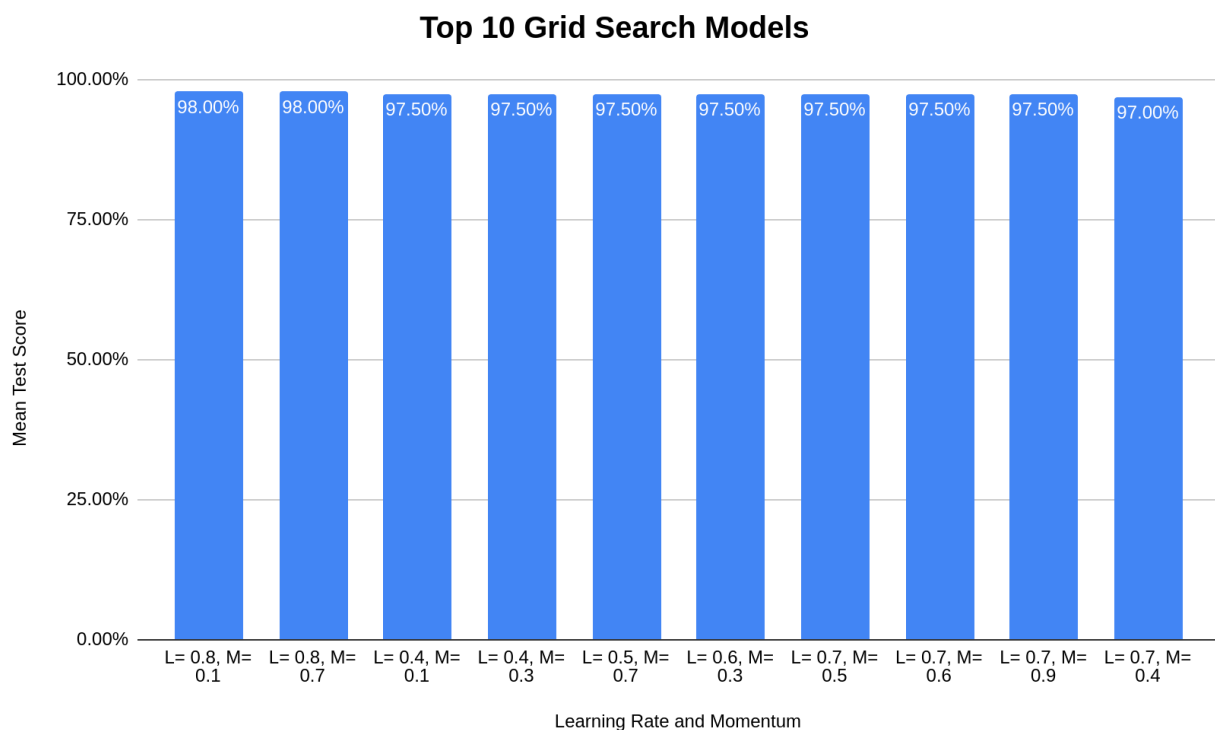


Figure 9: Top 10 models with the best learning rate and momentum combinations. L=Learning rate. M=Momentum.

ii) Performance by Hidden Units

Table 2. shows that there was a strong correlation between the increase in hidden units and test score performance. This may be due to the function converging at higher levels of abstraction. Moreover, the number of iterations appeared to reduce with respect to the hidden unit sizes. Despite the good performance of 10 hidden units, there is still the risk of overfitting the model, hence why I decided to choose the model with 5 hidden units.

Model Name	Training Set Score	Training Set Loss	Test Set Score	Iterations
1 hidden unit	19.50%	1.584927348	22.00%	5374
2 hidden units	88.50%	0.5362651509	84.00%	5434
3 hidden units	97.50%	0.2379743766	94.00%	4374
4 hidden units	98.50%	0.06731943038	96.00%	4795
5 hidden units	100.00%	0.02579760379	100.00%	3453
6 hidden units	100.00%	0.02408899145	100.00%	2861
7 hidden units	100.00%	0.02733212281	96.00%	3403
8 hidden units	100.00%	0.0210883114	98.00%	2672
9 hidden units	100.00%	0.02341073699	100.00%	2992
10 hidden units	100.00%	0.01885864715	100.00%	2382

Table 2: Performance of 10 different models with hidden units sizes ranging from 1-10

iii) Improvements

Despite the high accuracy achieved by the chosen model, the model could still be improved. For instance, the 'sgd' solver was used throughout and was not tested against "adam". Other activation functions and an adaptive learning rate could have been tested. These tests may not have been useful for this dataset due to its small size, however, expanding the scope of grid search to the other hyperparameters may be useful for a much larger data set.

b) Intermediate Agent

For evaluating the ticketing routing agent, the pro's and con's of the system were considered.

Pro's

- Even after an early prediction, the system can still receive answers from the user to improve the model
- Using the mode to fill in the missing values, is readily comprehensible and simple to calculate.

Cons

- There could be instances where there is no mode (haven't found in our data set).
- The user may not want to give answers after it has chosen its response, hence there is a chance of losing out on user data during early predictions.

Improvements:

To improve the system, the system could allow the user to answer multiple questions at the same time, this would speed up the ticket filling time. Moreover, the system could make predictions in batches of 3 instead of after each question. This could help improve the accuracy of each prediction.

C) Advanced Agent

For comparing and evaluating the different algorithms, the mean test score was used. After multiple runs, the results showed that the Decision Tree classifier outperformed the other algorithms with a

mean test score of 100%. Conversely, the Naïve Bayes classifier performed the worst with a mean test score of 80% (**see Figure 10**).

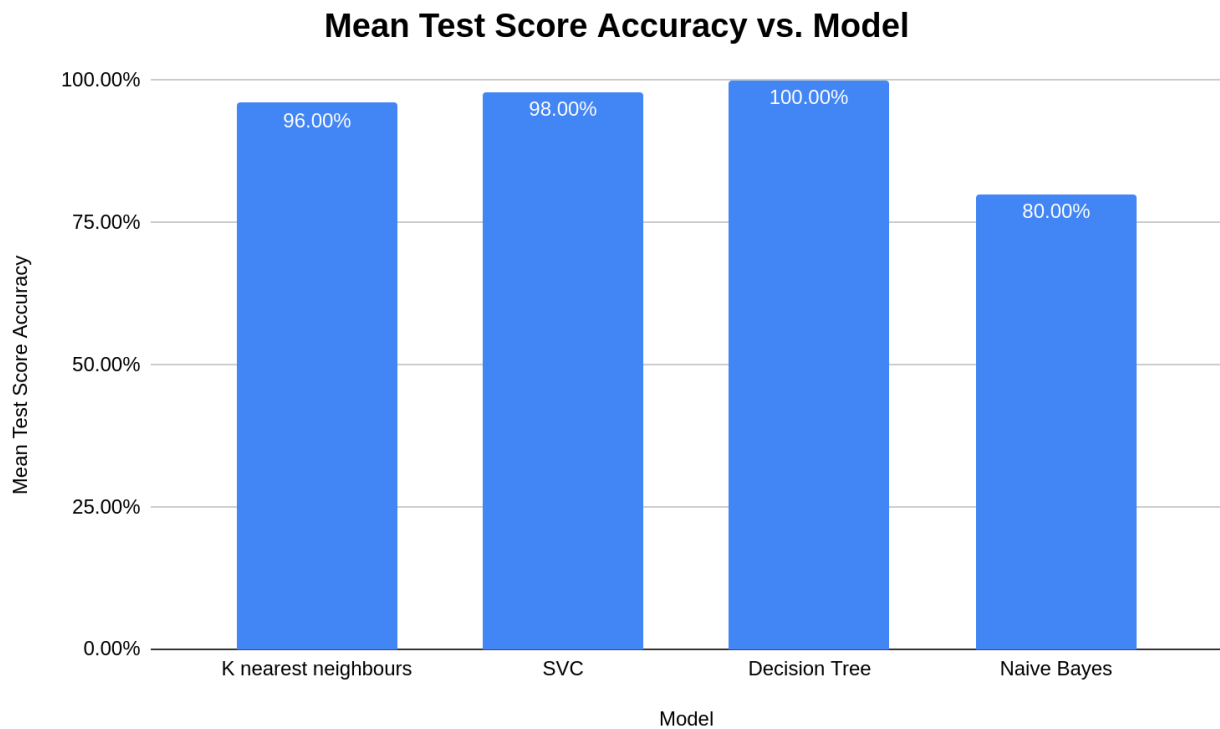


Figure 10: Graph showing the mean test score of the 4 different classifier algorithms.

Looking at the Naïve Bayes' confusion matrix (**see Figure 11**) we can see that it struggled to accurately classify inputs associated with Credentials, Equipment and Networking correctly. This may be because the training set only had 200 samples and Naïve Bayes classifiers tend to require a large data set to obtain good results (Jadhav & Channe, 2013).

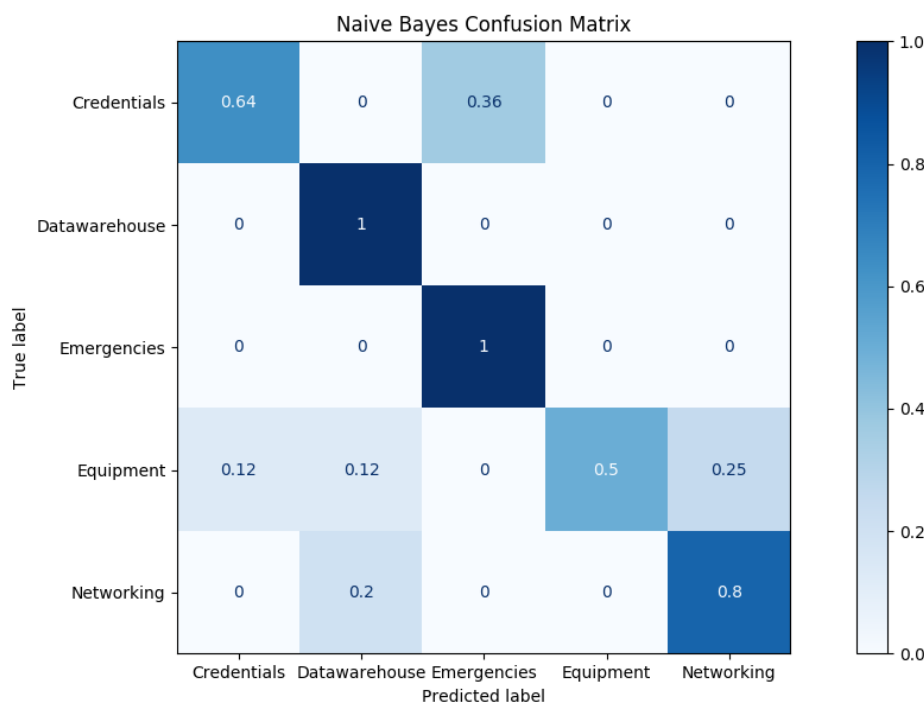


Figure 11: Naïve Bayes Classifier Confusion Matrix

On the other hand, the decision tree's high accuracy could be due to the tree-pruning it implements which is used to improve the prediction and classification accuracy by minimising the over-fitting problem decision tree's have (Jadhav & Channe, 2013)

Moreover, the SVM and KNN classifiers performed well with test score's of 98% and 96% respectively. SVM's strong performance with the small data set may be due to how SVMs only require a dozen of examples for training (Bhavsar & Ganatra, 2012). KNNs tend to perform well on datasets with many class labels (Bhavsar & Ganatra, 2012) and have shown to have good accuracy in general (Bhavsar & Ganatra, 2012; Jadhav & Channe, 2013).

Attributes	KNN	Naïve Bayes	Decision Tree	SVM
Deterministic/Non-deterministic	Non-deterministic	Non-deterministic	Deterministic	Non-deterministic
Effective with	Small Data	Huge Data	Large Data	Small Data
Learning Speed	Fast	Fast	Fast	Average
Tolerance to Overfitting	Very Good	Very Good	Good	Good
Accuracy	Very Good	Average	Excellent	Very Good

Table 3: Attribute comparison between the 4 different classifier algorithms

C. Test Summary

Below is a summary of all the tests that were run on the system:

Test	Agent	Method	Description	Works?
1	Basic	Encode	Encodes the data set using the one-hot encoding method	Yes
2	Basic	Grid Search	Runs grid search and exports data into a CSV	Yes
3	Basic	Basic	Plots the models for basic	Yes
4	Intermediate	Get User Input	Gets the user's input.	Yes
5	Intermediate	Give Options	Gives the user response team options based on what was wrongly predicted before	Yes
6	Intermediate	Prediction	Makes a prediction based on the answers provided	Yes
7	Intermediate	Fill Missing Columns	During an early prediction, fills in the missing columns by using the mode	Yes
8	Intermediate	New Ticket Request	Asks for another ticket request	Yes
9	Intermediate	Team Allocate	Allocates a team to the user based on the option chosen	Yes
10	Intermediate	Check If Happy	Checks if the user is happy with the prediction	Yes
11	Intermediate	Get Feedback	Runs a series of feedback questions to get answers for improving the model after an early prediction	Yes
12	Intermediate	Retrain	Retrains the model based on the new full list of answers and the selected team	Yes
13	Intermediate	Intermediate	Runs the ticketing routing agent	Yes
14	Advanced	Advanced	Runs all 4 algorithms and plots their test scores on separate confusion matrices	Yes

Test Evidence:

Test 1: Successfully encodes the inputs and outputs

```
Inputs
[[0 1 1 ... 0 0 0]
 [1 0 1 ... 1 1 0]
 [0 0 0 ... 0 1 0]
 ...
 [0 1 0 ... 0 0 0]
 [0 0 0 ... 1 0 1]
 [0 1 1 ... 0 0 0]]

Outputs
[[0 0 1 0 0]
 [0 0 0 0 1]
 [1 0 0 0 0]
 ...
 [0 0 1 0 0]
 [0 0 0 1 0]
 [0 0 1 0 0]]
```

Test 2: Grid Search Results can be found in the “Results folder”

Test 3: Successfully trains 10 different models

```

learning on dataset Loss vs No. Epochs For Models With Varying Hidden Units
training: 1 hidden unit
Training set accuracy: 0.275000
Training set loss: 1.562320
Test set accuracy: 0.280000
Iterations: 3634.000000
training: 2 hidden units
Training set accuracy: 0.930000
Training set loss: 0.499447
Test set accuracy: 0.840000
Iterations: 6405.000000
training: 3 hidden units
Training set accuracy: 0.975000
Training set loss: 0.194891
Test set accuracy: 0.940000
Iterations: 4993.000000
training: 4 hidden units
Training set accuracy: 0.995000
Training set loss: 0.048618
Test set accuracy: 0.980000
Iterations: 3754.000000
training: 5 hidden units
Training set accuracy: 1.000000
Training set loss: 0.032209
Test set accuracy: 1.000000
Iterations: 3743.000000
training: 6 hidden units
Training set accuracy: 0.995000
Training set loss: 0.030561
Test set accuracy: 0.980000
Iterations: 3677.000000
training: 7 hidden units
Training set accuracy: 1.000000
Training set loss: 0.024071
Test set accuracy: 1.000000
Iterations: 2978.000000
training: 8 hidden units
Training set accuracy: 1.000000
Training set loss: 0.021146
Test set accuracy: 1.000000
Iterations: 2760.000000
training: 9 hidden units
Training set accuracy: 1.000000
Training set loss: 0.022827
Test set accuracy: 1.000000
Iterations: 2807.000000
training: 10 hidden units
Training set accuracy: 1.000000
Training set loss: 0.021844
Test set accuracy: 1.000000
Iterations: 2793.000000

```

Test 4:

A) Successfully gets user's input where a user answers "Yes, Yes" and stores it in the answers array

```

Question 1: Request? (Yes/No/Q)
yes
Question 2: Incident? (Yes/No/Q)
yes
▶ answers = {list} <class 'list': ['yes', 'yes']

```

B) Tells the user that their answer is invalid when they give a wrong option

```

Question 3: WebServices? (Yes/No/Q)
n
Sorry, that is not a valid response
Question 3: WebServices? (Yes/No/Q)

```

Test 5: Successfully provides options for the user whilst omitting the previously predicted team

```

Question 4: Login? (Yes/No/P/Q)
y
Based on your answers, your request will be sent to the Credentials team.
Are you happy with this allocation?(Yes/No)
no
Apologies for that, which team from below would you like to speak to? Please select a number:
1 - Datawarehouse
2 - Emergencies
3 - Equipment
4 - Networking

```

Test 6: Successfully makes a prediction

```

Question 2: Incident? (Yes/No/Q)
yes
Question 3: WebServices? (Yes/No/Q)
p
Sorry, that is not a valid response
Question 3: WebServices? (Yes/No/Q)
no
Question 4: Login? (Yes/No/P/Q)
p
Based on your answers, your request will be sent to the Credentials team.

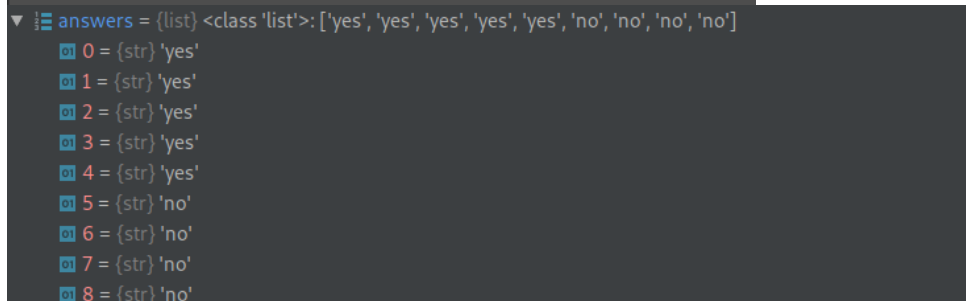
```

Test 7: Successfully fills in missing columns using the mode when the user answered “Yes, Yes, Yes” and asks for a prediction.

```

Question 1: Request? (Yes/No/Q)
yes
Question 2: Incident? (Yes/No/Q)
yes
Question 3: WebServices? (Yes/No/Q)
yes
Question 4: Login? (Yes/No/P/Q)
p

```



```

▼ In [ ]: answers = {list} <class 'list'>: ['yes', 'yes', 'yes', 'yes', 'yes', 'no', 'no', 'no', 'no']
      0 = {str} 'yes'
      1 = {str} 'yes'
      2 = {str} 'yes'
      3 = {str} 'yes'
      4 = {str} 'yes'
      5 = {str} 'no'
      6 = {str} 'no'
      7 = {str} 'no'
      8 = {str} 'no'

```

Test 8: Successfully asks for a new ticket

```

Question 6: Printing? (Yes/No/P/Q)
no
Question 7: ID Cards? (Yes/No/P/Q)
yes
Question 8: Staff? (Yes/No/P/Q)
no
Question 9: Students? (Yes/No/P/Q)
yes
Based on your answers, your request will be sent to the Networking team.
Are you happy with this allocation?(Yes/No)
yes
Have a nice day!
Would you like another ticket request? (Yes/No)

```

Test 9: Successfully allocates the team to the Datawarehouse team based on the user’s choice

```

Apologies for that, which team from below would you like to speak to? Please select a number:
1 - Datawarehouse
2 - Emergencies
3 - Equipment
4 - Networking
]
Thank you for your patience, your request will be sent to the Datawarehouse team.

```

Test 10:

A) Successfully asks if the user is happy. Provides the correct response when the user says no

```

Based on your answers, your request will be sent to the Credentials team.
Are you happy with this allocation?(Yes/No)
no
Apologies for that, which team from below would you like to speak to? Please select a number:
1 - Datawarehouse
2 - Emergencies
3 - Equipment
4 - Networking
]
Thank you for your patience, your request will be sent to the Datawarehouse team.

```

B) Successfully asks if the user is happy. Provides the correct response when the user says yes

```

Based on your answers, your request will be sent to the Networking team.
Are you happy with this allocation?(Yes/No)
yes
Have a nice day!

```

Test 11: Successfully starts the feedback process by continuing from Question 4, where the user made an early prediction

```

Question 1: Request? (Yes/No/Q)
yes
Question 2: Incident? (Yes/No/Q)
yes
Question 3: WebServices? (Yes/No/Q)
]
Sorry, that is not a valid response
Question 3: WebServices? (Yes/No/Q)
no
Question 4: Login? (Yes/No/P/Q)
]
Based on your answers, your request will be sent to the Credentials team.
Are you happy with this allocation?(Yes/No)
no
Apologies for that, which team from below would you like to speak to? Please select a number:
1 - Datawarehouse
2 - Emergencies
3 - Equipment
4 - Networking
]
Thank you for your patience, your request will be sent to the Datawarehouse team.

To help improve our system please answer the remaining question(s)

Question 4 (Feedback): Login? (Yes/No/Q)
]
Sorry, that is not a valid response
Question 4 (Feedback): Login? (Yes/No/Q)
yes

```

Test 12: Successfully retrains the model after the user answers all 9 questions

```

Iteration 4762, loss = 0.06775921
Iteration 4763, loss = 0.06766499
Iteration 4764, loss = 0.06776153
Iteration 4765, loss = 0.06769533
Iteration 4766, loss = 0.06757849
Iteration 4767, loss = 0.06746444
Iteration 4768, loss = 0.06739435
Iteration 4769, loss = 0.06732885
Iteration 4770, loss = 0.06727365
Iteration 4771, loss = 0.06801212
Iteration 4772, loss = 0.06791875
Iteration 4773, loss = 0.06783625
Iteration 4774, loss = 0.06776781
Training loss did not improve more than tol=0.000100 for 1000 consecutive epochs. Stopping.
Our system, has learnt from your input.
(Model has been retrained...)

```

Test 13: Successfully runs intermediate agent with evidence of two tickets

```

Question 1: Request? (Yes/No/Q)
Yes
Question 2: Incident? (Yes/No/Q)
Yes
Question 3: WebServices? (Yes/No/Q)
Yes
Question 4: Login? (Yes/No/P/Q)
Yes
Question 5: Wireless? (Yes/No/P/Q)
Yes
Question 6: Printing? (Yes/No/P/Q)
Yes
Question 7: ID Cards? (Yes/No/P/Q)
Yes
Question 8: Staff? (Yes/No/P/Q)
Yes
Question 9: Students? (Yes/No/P/Q)
Yes
Based on your answers, your request will be sent to the Networking team.
Are you happy with this allocation?(Yes/No)
Yes
Have a nice day!

Would you like another ticket request? (Yes/No)
Yes
Hi there! Please answer the following 9 questions so that your ticket request can be routed to the correct team.
If you'd like us to make a prediction, please answer at least 3 questions
Please answer the questions with either Yes, No, P or Q.
P = early prediction. Please note, that we will need at least 3 answers before this is available
Q = quit

Question 1: Request? (Yes/No/Q)
Yes
Question 2: Incident? (Yes/No/Q)
Yes
Question 3: WebServices? (Yes/No/Q)
Yes
Question 4: Login? (Yes/No/P/Q)
Yes
Question 5: Wireless? (Yes/No/P/Q)
Yes
Based on your answers, your request will be sent to the Datawarehouse team.
Are you happy with this allocation?(Yes/No)
Yes
Have a nice day!

```

Test 14: Successfully runs all 4 algorithms and plots their confusion matrices

```
SVC Test Score: 1.0
SVC Confusion Matrix
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
Naive Bayes Test Score: 0.68
Naive Bayes Confusion Matrix
[[0.22222222 0.         0.77777778 0.         0.         ]
 [0.         0.75      0.         0.         0.25      ]
 [0.         0.         1.         0.         0.         ]
 [0.26666667 0.13333333 0.         0.53333333 0.06666667]
 [0.         0.         0.         0.         1.         ]]
Decision Tree Test Score: 1.0
Decision Tree Confusion Matrix
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
KNN Test Score: 0.98
KNN Confusion Matrix
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0.06666667 0.         0.         0.93333333 0.         ]
 [0.         0.         0.         0.         1.         ]]
```

References:

1. Alaliyat, S. (2008). *Video-based Fall Detection in Elderly's Houses* (Gjøvik University College). Retrieved from <https://www.researchgate.net/publication/267953942>
2. Bhavsar, H., & Ganatra, A. (2012). A Comparative Study of Training Algorithms for Supervised Machine Learning. In *International Journal of Soft Computing and Engineering (IJSCE)*.
3. Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2–3), 131–163.
4. Gupta, S. (2019). ML Decision Tree. *HackerEarth*. Retrieved from <https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/>
5. Jadhav, S. D., & Channe, H. P. (2013). Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques. In *International Journal of Science and Research (IJSR) ISSN* (Vol. 5). Retrieved from www.ijsr.net
6. Keller, J. M., Gray, M. R., & Givens, J. A. (1985). *A Fuzzy K-Nearest Neighbor Algorithm*.
7. Sayad, S. (2019a). Decision Tree - Classification. Retrieved December 18, 2019, from Saed Sayad website: https://www.saedsayad.com/decision_tree.htm
8. Sayad, S. (2019b). Support Vector Machine - Classification (SVM). Retrieved December 18, 2019, from [saedsayad.com](https://www.saedsayad.com/support_vector_machine.htm) website: https://www.saedsayad.com/support_vector_machine.htm
9. Suykens, J. A. K., & Vandewalle, J. (1999). Least Squares Support Vector Machine Classifiers. In *Neural Processing Letters* (Vol. 9).
10. Swain, P. H., & Hauska, H. (1977). The Decision Tree Classifier: Design and Potential. In *IEEE TRANSACTIONS ON GEOSCIENCE ELECTRONICS*.

