Brian Koehler
UFID 7551-8931

# CAP4770 Assignment 4 Report

## Car Evaluation

1. **How you partitioned the data into training and test datasets**
   All features of the data set were encoded using SciKit-Learn's OneHotEncoder, because many of the features were categorical and would mean nothing to a machine learning model. The data set was then split into 70% for model training, and 30% for model testing using the train_test_split method.

2. **How you tuned the parameters of the training method**
   The neural network's max iterations were set to 450, because this was near the minimum value needed for convergence (leading to near-optimal training time). All were trained using the fit! method, because it would directly mutate the model.

3. **Performance value**
   Each model had a higher accuracy than the previous. Accuracies were determined using the accuracy_score method. Please note them below.

4. **Model size**
   The sizes of the models were very small, as the largest of the four was the Support Vector Machine at only 118 KB. The Naïve Bayes Classifier model was notably small at only 5 KB. Sizes were determined by importing Python's pickle package using PyCall, dumping the object to a file and examining the file on the operating system.

5. **Testing time**
   All were under one second, with the Classification Tree being near the longest. All times were determined using Jupyter's built-in timer. Please note them below.

6. **Training time**
   The neural network was the only one over one second at 1.9 seconds. All times were determined using Jupyter's built-in timer. Please note them below.

| | **Naïve Bayes Classifier** | **Classification Tree** | **Support Vector Machine** | **Neural Network** |
|---|---|---|---|---|
| **Data Preprocessing** | Features encoded using OneHotEncoder<br><br>Split 70-30 using SciKit-Learn's train_test_split function | Features encoded using OneHotEncoder<br><br>Split 70-30 using SciKit-Learn's train_test_split function | Features encoded using OneHotEncoder<br><br>Split 70-30 using SciKit-Learn's train_test_split function | Features encoded using OneHotEncoder<br><br>Split 70-30 using SciKit-Learn's train_test_split function |
| **Training Method Parameters** | N/A | N/A | N/A | max_iter=450 to converge |
| **Accuracy** | 87.48% | 96.53% | 98.07% | 99.81% |
| **Size** | 5 KB | 16 KB | 118 KB | 76 KB |
| **Testing Time** | 0.3s | 0.9s | 0.4s | 0.1s |
| **Training Time** | 0.2s | 0.8s | 0.7s | 1.9s |

## Abalone

1. **How you partitioned the data into training and test datasets**
   Categorical features were encoded using SciKit-Learn's LabelEncoder, because they would otherwise mean nothing to a machine learning model. The number of rings were then placed into classes of 1-7, 8-15, 16-23, and 24-29 in order to improve the accuracy of the model. The data set was then split into 70% for model training, and 30% for model testing using the train_test_split method.

2. **How you tuned the parameters of the training method**
   The neural network's max iterations were set to 430, because this was near the minimum value needed for convergence (leading to near-optimal training time). All were trained using the fit! method, because it would directly mutate the model.

3. **Performance value**
   The accuracies of this data set and models were noticeably lower than the previous data set, with the neural network having the best performance at 85.49%. The Naïve Bayes Classifier was clearly the worst of the four with a 73.76% accuracy. Accuracies were determined using the accuracy_score method.

4. **Model size**
   The sizes of the models were very small, as the largest of the four was the Support Vector Machine at only 107 KB. The Naïve Bayes Classifier model was notably small at only 1 KB. Sizes were determined by importing Python's pickle package using PyCall, dumping the object to a file and examining the file on the operating system.

5. **Testing time**
   All were under half a second and very close in time. All times were determined using Jupyter's built-in timer. Please note them below.

6. **Training time**
   Like the last data set, the neural network was the only one over one second at 3.7 seconds. All times were determined using Jupyter's built-in timer. Please note them below.

|  | **Naïve Bayes Classifier** | **Classification Tree** | **Support Vector Machine** | **Neural Network** |
|---|---|---|---|---|
| **Data Preprocessing** | String features encoded using LabelEncoder<br><br>Rings placed into categories of [1,7], [8, 15], [16, 23], [24, 29]<br><br>Split 70-30 using train_test_split function | String features encoded using LabelEncoder<br><br>Rings placed into categories of [1,7], [8, 15], [16, 23], [24, 29]<br><br>Split 70-30 using train_test_split function | String features encoded using LabelEncoder<br><br>Rings placed into categories of [1,7], [8, 15], [16, 23], [24, 29]<br><br>Split 70-30 using train_test_split function | String features encoded using LabelEncoder<br><br>Rings placed into categories of [1,7], [8, 15], [16, 23], [24, 29]<br><br>Split 70-30 using train_test_split function |
| **Training Method Parameters** | N/A | N/A | N/A | max_iter=430 to converge |
| **Accuracy** | 73.76% | 78.55% | 84.85% | 85.49% |
| **Size** | 1 KB | 80 KB | 107 KB | 41 KB |
| **Testing Time** | 0.3s | 0.3s | 0.2s | 0.1s |
| **Training Time** | 0.4s | 0.4s | 0.2s | 3.7s |

## Madelon

1. **How you partitioned the data into training and test datasets**
   All features of the data set were encoded using SciKit-Learn's OneHotEncoder, because many of the features were categorical and would mean nothing to a machine learning model. The data set was then split into 70% for model training, and 30% for model testing using the train_test_split method.

2. **How you tuned the parameters of the training method**
   The neural network's max iterations was set to 430, because this was near the minimum value needed for convergence (leading to near-optimal training time). All were trained using the fit! method, because it would directly mutate the model.

3. **Performance value**
   The performance of the models on this data set were not good, with the neural network's accuracy being notably below expectations. Accuracies were determined using the accuracy_score method.

4. **Model size**
   Sizes of the models for this data set were greater, as demonstrated by the 7.3 MB and 1.2 MB allocated for the Support Vector Machine and Neural Network, respectively. Sizes were determined by importing Python's pickle package using PyCall, dumping the object to a file and examining the file on the operating system.

5. **Testing time**
   All were under one second, with the Support Vector Machine being near the longest. All times were determined using Jupyter's built-in timer. Please note them below.

6. **Training time**
   The support vector machine was the only one over one second at 1.5 seconds. All times were determined using Jupyter's built-in timer. Please note them below.

| | **Naïve Bayes Classifier** | **Classification Tree** | **Support Vector Machine** | **Neural Network** |
|---|---|---|---|---|
| **Data Preprocessing** | Split using premade partitions | Split using premade partitions | Split using premade partitions | Split using premade partitions |
| **Training Method Parameters** | N/A | N/A | N/A | max_iter=430 to converge |
| **Accuracy** | 59.17% | 75.00% | 68.67% | 49.83% |
| **Size** | 17 KB | 23 KB | 7.3 MB | 1.2 MB |
| **Testing Time** | 0.5s | 0.1s | 0.7s | 0.3s |
| **Training Time** | 0.1s | 0.6s | 1.5s | 0.9s |

## KDD Cup 1999

1. **How you partitioned the data into training and test datasets**
   All features of the data set were encoded using SciKit-Learn's OneHotEncoder, because many of the features were categorical and would mean nothing to a machine learning model. The data set was then split into 70% for model training, and 30% for model testing using the train_test_split method.

2. **How you tuned the parameters of the training method**
   The neural network's max iterations were set to 250 to reach convergence and greatly reduce the training time while retaining good accuracy. Similarly, the max iterations of the support vector machine were set to 250. Although this does not reach convergence, it significantly reduces the training time while retaining good accuracy. All were trained using the fit! method, because it would directly mutate the model.

3. **Performance value**
   The performance values of these models were exceptional, because each was 97% accurate or higher. Perhaps this is the result of the training set being so large. All accuracies were determined using the accuracy_score method.

4. **Model size**
   Sizes for the models of this data set were small once again, with the highest being 109 KB. Sizes were determined by importing Python's pickle package using PyCall, dumping the object to a file and examining the file on the operating system.

5. **Testing time**
   The testing times of the models for this data set were much higher than the previous data sets, as all times were over 20 seconds. This can be attributed to the higher amount of tuples in the data, along with the large number of features. All times were obtained using Jupyter's built-in timer.

6. **Training time**
   The training times of the models for this data set were much higher than the previous data sets, with the neural network time reaching 15.5 seconds. This can be attributed to the higher amount of tuples in the data, along with the large number of features. All times were obtained using Jupyter's built-in timer.

| | **Naïve Bayes Classifier** | **Classification Tree** | **Support Vector Machine** | **Neural Network** |
|---|---|---|---|---|
| **Data Preprocessing** | Used 10% of data file<br><br>All dependent variables classified as normal or attack<br><br>Encoded 2nd, 3rd, and 4th columns using LabelEncoder<br><br>Split into 10-90 using train_test_split | Used 10% of data file<br><br>All dependent variables classified as normal or attack<br><br>Encoded 2nd, 3rd, and 4th columns using LabelEncoder<br><br>Split into 10-90 using train_test_split | Used 10% of data file<br><br>All dependent variables classified as normal or attack<br><br>Encoded 2nd, 3rd, and 4th columns using LabelEncoder<br><br>Split into 10-90 using train_test_split | Used 10% of data file<br><br>All dependent variables classified as normal or attack<br><br>Encoded 2nd, 3rd, and 4th columns using LabelEncoder<br><br>Split into 10-90 using train_test_split |
| **Training Method Parameters** | N/A | N/A | max_iter=250 to reduce time | max_iter=250 to converge and reduce time |
| **Accuracy** | 97.92% | 99.90% | 97.86% | 98.16% |
| **Size** | 2 KB | 10 KB | 1 KB | 109 KB |
| **Testing Time** | 22.3s | 23.5s | 22.3s | 21.5s |
| **Training Time** | 2.9s | 1.9s | 4.5s | 15.5s |