Assignment 4 Report

CAP4770 Spring 2021

Professor Dobra

Miguel Caputo

Car Evaluation Data Set

I found that the best model for this data set is Neural Networks, and the worse one is Naive Bayes Classifier

How you partitioned the data into training and test datasets

- I divided the data using the ScikitLearn function *train_test_split()*
- 70% of the data went into training and 30% went into testing
- I am using the seed: 100 for this test.
- *car_x_train, car_x_test, car_y_train, car_y_test = train_test_split(car_x_encoded,car_y,test_size=0.7, random_state=100)*

How you tuned the parameters of the training method (if any)

I didn't tune the parameters for the Car evaluation Data Set, I did encode them use the ScikitLearn function *OneHotEncoding()*

Naive Bayes Classifier Model

- **The performance/error on the training dataset:** 85.74181117533719 %
- **The size of the model**: 5.341 kB
- **The testing time:** 0.132982 seconds
- **The training time:** 0.028264 seconds

Notes:

- For this model, I found that using the ScikitLearn *CategoricalNB() function*, the model returns a better accuracy than the other type of Naive Bayes functions, the worst one being *GaussianNB()* (overall having 5-6% less precision.)

Decision Trees Model

- **The performance/error on the training dataset:** 97.30250481695568 %
- **The size of the model:** 17.536 kB
- **The testing time:** 0.001553 seconds
- **The training time:** 0.004335 seconds

Notes:

- For this model, I am using the ScikitLearn function *DecisionTreeClassifier()*

Support Vector Machine Model

- **The performance/error on the training dataset:** 96.33911368015414%
- **The size of the model:** 130.856 kB
- **The testing time:** 0.039099 seconds
- **The training time:** 0.065432 seconds

Notes:

- For this model, I am using the ScikitLearn function *SVC()* I found that it has better accuracy than other functions like *LinearSVC()*.

Neural Networks Model

- **The performance/error on the training dataset:** 99.42196531791907 %
- **The size of the model:** 97.856 kB
- **The testing time:** 0.002951 seconds
- **The training time:** 2.744010 seconds

Notes:

- For this model, I am using the ScikitLearn function *MLPClassifier()* and I am changing the max_iter value to 1000
- MLPClassifier(max_iter = 1000)

Car Evaluation Data Set					
Model	**Function Used**	**Performance**	**Size**	**Testing Time**	**Training Time**
Naive Bayes	CategoricalNB()	85.74181117533719 %	5.341 kB	0.132982 seconds	0.028264 seconds
Decision Trees	DecisionTreeClassifier()	97.30250481695568 %	17.536 kB	0.001553 seconds	0.004335 seconds
Support Vector Machine	SVC()	96.33911368015414%	130.856 kB	0.039099 seconds	0.065432 seconds
Neural Networks	MLPClassifier()	99.42196531791907 %	97.856 kB	0.002951 seconds	2.744010 seconds

Abalone Data Set

I found that the best model for this data set is Neural Networks, and the worse one is Naive Bayes Classifier.

How you partitioned the data into training and test datasets

- I divided the data using the ScikitLearn function *train_test_split()*
- 70% of the data went into training and 30% went into testing
- I am using the seed: 100 for this test.
- *abalone_x_train, abalone_x_test, abalone_y_train, abalone_y_test = train_test_split(abalone_x,abalone_y,test_size=0.7, random_state=100)*

How you tuned the parameters of the training method (if any)

I didn't tune the parameters for the Abalone Data Set, I did encode the first column using the SkitLearn function *LabelEncoder()*

For the labels, I added 1.5 to all the rings the I modified the values like:

- If rings between 1 and 9, value = 1
- If rings between 9 and 10, value = 2
- If rings greater than 10, value = 3

Naive Bayes Classifier Model

- **The performance/error on the training dataset:** 70.81339712918661 %
- **The size of the model:** 1.067 kB
- **The testing time:** 0.025575 seconds
- **The training time:** 0.087234 seconds

Notes:

- For this model, I found that using the ScikitLearn *GaussianNB() function*, the model returns a better accuracy than the other type of Naive Bayes functions, the worst one being *MultinomialNB()* (overall having 5% less precision)

Decision Trees Model

- **The performance/error on the training dataset:** 72.08931419457735 %
- **The size of the model:** 87.175 kB
- **The testing time:** 0.006975 seconds
- **The training time:** 0.033380 seconds

Notes:

- For this model, I am using the ScikitLearn function *DecisionTreeClassifier()*

Support Vector Machine Model

- **The performance/error on the training dataset:** 79.10685805422647 %
- **The size of the model:** 0.927 kB
- **The testing time:** 0.006591 seconds
- **The training time:** 0.076971 seconds

Notes:

- For this model, I am using the ScikitLearn function *LinearSVC()* I found that it has better accuracy than other functions like *SVC()*

Neural Networks Model

- **The performance/error on the training dataset:** 80.38277511961722 %
- **The size of the model:** 47.545 kB
- **The testing time:** 0.013309 second
- **The training time:** 3.314147 seconds

Notes:

- For this model, I am using the ScikitLearn function *MLPClassifier()* and I am changing the max_iter value to 1000
- MLPClassifier(max_iter = 1000)

Abalone Data Set					
Model	**Function Used**	**Performance**	**Size**	**Testing Time**	**Training Time**
Naive Bayes	GaussianNB()	70.81339712918661 %	1.067 kB	0.025575 seconds	0.087234 seconds
Decision Trees	DecisionTreeClassifier()	72.08931419457735 %	87.175 kB	0.006975 seconds	0.033380 seconds
Support Vector Machine	LinearSVC()	79.10685805422647 %	0.927 kB	0.006591 seconds	0.076971 seconds
Neural Networks	MLPClassifier()	80.38277511961722 %	47.545 kB	0.013309 second	3.314147 seconds

Madelon Data Set

I found that the best model for this data set is Decision Trees, and the worse one is Neural Networks

How you partitioned the data into training and test datasets

The data was already divided between training and valid, the last one being used as testing.

How you tuned the parameters of the training method (if any)

I didn't tune nor encoded the parameters for the Madelon Data Set.

Naive Bayes Classifier Model

- **The performance/error on the training dataset:** 59.26544240400667 %
- **The size of the model:** 16.696 kB
- **The testing time:** 0.011103 seconds
- **The training time:** 0.026171 seconds

Notes:

- For this model, I found that there is not much difference between using the functions *GaussianNB()* and *MultinomialNB()* so I decided to use *GaussianNB()* which is slightly better.

Decision Trees Model

- **The performance/error on the training dataset:** 73.95659432387313 %
- **The size of the model:** 22.987 kB
- **The testing time:** 0.002058 seconds
- **The training time:** 0.590340 seconds

Notes:

- For this model, I am using the ScikitLearn function *DecisionTreeClassifier()*

Support Vector Machine Model

- **The performance/error on the training dataset:** 68.61435726210351 %
- **The size of the model:** 7301.657 kB
- **The testing time:** 0.494051 seconds
- **The training time:** 1.285359 seconds

Notes:

- For this model, I am using the ScikitLearn function *SVC()* I found that it has better accuracy than other functions like Linear*SVC()*

Neural Networks Model

- **The performance/error on the training dataset:** 49.74958263772955 %
- **The size of the model:** 1611.634 kB
- **The testing time:** 0.003883 seconds
- **The training time:** 0.844653 seconds

Notes:

- For this model, I am using the ScikitLearn function *MLPClassifier()* and I am changing the max_iter value to 1000
- MLPClassifier(max_iter = 1000)

Madelon Data Set					
Model	**Function Used**	**Performance**	**Size**	**Testing Time**	**Training Time**
Naive Bayes	GaussianNB()	59.26544240400667 %	16.696 kB	0.011103 seconds	0.026171 seconds
Decision Trees	DecisionTreeClassifier()	73.95659432387313 %	22.987 kB	0.002058 seconds	0.590340 seconds
Support Vector Machine	SVC()	68.61435726210351 %	7301.657 kB	0.494051 seconds	1.285359 seconds
Neural Networks	MLPClassifier()	49.74958263772955 %	1611.634 kB	0.003883 seconds	0.844653 seconds

Kddcup Data Set (10%)

I found that the best model for this data set is Decision Trees, and the worse one is Naive Bayes Classifier.

How you partitioned the data into training and test datasets

- I divided the data using the ScikitLearn function *train_test_split()*
- 70% of the data went into training and 30% went into testing
- I am using the seed: 100 for this test.
- *kdd10_x_train, kdd10_x_test, kdd10_y_train, kdd10_y_test = train_test_split(kddcup_10_x,kddcup_10_y,test_size=0.7, random_state=100)*

How you tuned the parameters of the training method (if any)

I didn't tune the parameters for the Kddcup Data Set (10%), I did encode the second, third, and fourth columns using the SkitLearn function *LabelEncoder()*

For the labels, I modify the values like:

- If the value was "normal." value = "1"
- If the values were anything else value = "2"

Naive Bayes Classifier Model

- **The performance/error on the training dataset:** 92.17513342824563 %
- **The size of the model:** 1.991 kB
- **The testing time:** 5.633944 seconds
- **The training time:** 13.381048 seconds

Notes:

- For this model, I found that the best function is GaussianNB() so that's the one I am using.

Decision Trees Model

- **The performance/error on the training dataset:** 99.96761286578907 %
- **The size of the model:** 21.419 kB
- **The testing time:** 4.861191 seconds

- **The training time:** 19.511176 seconds

Notes:

- For this model, I am using the ScikitLearn function *DecisionTreeClassifier()*

Support Vector Machine Model

- **The performance/error on the training dataset:** 98.44001970217331 %
- **The size of the model:** 1.057 kB
- **The testing time:** 5.581422 seconds
- **The training time:** 18.411336 seconds

Notes:

- For this model, I am using the ScikitLearn function *LinearSVC()* the function *SVC() takes a lot of time to work.*

Neural Networks Model

- **The performance/error on the training dataset:** 99.5411822653451 %
- **The size of the model:** 143.076 kB
- **The testing time:** 5.428541 seconds
- **The training time:** 83.833982 seconds

Notes:

- For this model, I am using the ScikitLearn function *MLPClassifier()* and I am changing the max_iter value to 1000
- MLPClassifier(max_iter = 1000)

Kddcup Data Set (10%)					
Model	**Function Used**	**Performance**	**Size**	**Testing Time**	**Training Time**
Naive Bayes	GaussianNB()	92.17513342824563 %	1.991 kB	5.633944 seconds	13.381048 seconds
Decision Trees	DecisionTreeClassifier()	99.96761286578907 %	21.419 kB	4.861191 seconds	19.511176 seconds
Support Vector Machine	LinearSVC()	98.44001970217331 %	1.057 kB	5.581422 seconds	18.411336 seconds
Neural Networks	MLPClassifier()	99.5411822653451 %	143.076 kB	5.428541 seconds	83.833982 seconds

Kddcup Data Set (100%)

I was only able to use the Naive Bayes and Decision Trees models due to the size of this data set. I left the code running all night and it never finished for the other models.

How you partitioned the data into training and test datasets

- I divided the data using the MLJ function *partition()*
- 70% of the data went into training and 30% went into testing
- I am using partition because *train_test_split()* takes a lot of time
- train, test = partition(1:length(kddcup_y), 0.70, shuffle = true)

How you tuned the parameters of the training method (if any)

I didn't tune the parameters for the Kddcup Data Set (100%), I did encode the second, third, and fourth columns using the SkitLearn function *LabelEncoder()*

For the labels, I modify the values like:

- If the value was "normal." value = "1"
- If the values were anything else value = "2"

Naive Bayes Classifier Model

- **The performance/error on the training dataset:** 94.23012407376785 %
- **The size of the model:** 1.991 kB
- **The testing time:** 71.944270 seconds
- **The training time:** 175.422732 seconds

Notes:

- For this model, I found that the best function is GaussianNB().

Decision Trees Model

- **The performance/error on the training dataset:** 99.99407973575207 %
- **The size of the model:** 45.899 kB
- **The testing time:** 67.061262 seconds
- **The training time:** 232.248612 seconds

Notes:

- For this model, I am using the ScikitLearn function *DecisionTreeClassifier()*

Kddcup Data Set (100%)					
Model	**Function Used**	**Performance**	**Size**	**Testing Time**	**Training Time**
Naive Bayes	GaussianNB()	94.23012407376785 %	1.991 kB	71.944270 seconds	175.422732 seconds
Decision Trees	DecisionTreeClassifier()	99.99407973575207 %	45.899 kB	67.061262 seconds	232.248612 seconds

Extra Information:

Necessary Packages:

- PyCall
- CSV
- DataFrames
- ScikitLearn
- MLJ (Only if running the KddFullModel() function)

All of the necessary Packages can be downloaded inside the program, just uncomment the necessary packages at the beginning of the code.

I created a function that would run the Kddcup Full Data Set but it takes a lot of time to run, as commented at the end of the code, just run if there is extra time to spare.

My code assumes that the Data Sets are in the same main folder as the code, inside another folder called “Datasets” each dataset being inside a folder with its respective name: “Car”, “Abalone”, “Madelon” and “Kddcup”

For example, Car Dataset files are in "Datasets/Car”

My code was run using seed: 100 when splitting the data, the seed is not necessary and can be erased but I used it to get consistent values.