Classification workflows conversational agent example runs

Anton Antonov
MathematicaForPrediction at GitHub
MathematicaForPrediction at WordPress
ConversationalAgents at GitHub
May 2018

Introduction

This (laconic) notebook gives basic demonstrations of the functionalities developed in the project “Classification workflows conversational agent”.

Load preliminary code

Load the packages for the ClCon monad, monad tracing, functional parsers, and data obtaining:

In[28]:=
```
Import["https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/MonadicProgramming/MonadicContextualClassification.m"]
Import["https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/MonadicProgramming/MonadicTracing.m"]
Import["https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/FunctionalParsers.m"]
Import["https://raw.githubusercontent.com/antononcube/MathematicaVsR/master/Projects/ProgressiveMachineLearning/Mathematica/GetMachineLearningDataset.m"]
```

Load project code

Get and run the parsers specification and generation code:

In[32]:=
```
Clear["ebnf*"]
Import["https://raw.githubusercontent.com/antononcube/ConversationalAgents/master/EBNF/ClassifierWorkflowsGrammar.m"]
Names["ebnf*"]
LeafCount /@ res
```

Out[34]=
```
{ebnfClassifierEnsembleMaking, ebnfClassifierMaking, ebnfClassifierQuery, ebnfClassifierTesting, ebnfCommand,
 ebnfDataLoad, ebnfDataOutliers, ebnfDataStatistics, ebnfDataTransform, ebnfPipelineCommands, ebnfROCPlot, ebnfSplitting, ebnfVerification}
```

Out[35]=
```
{628, 763, 897, 466, 424, 960, 531, 905, 4465, 1417, 491, 843, 25}
```

Load the translator package:

In[36]:=
```
Import["https://raw.githubusercontent.com/antononcube/ConversationalAgents/master/Projects/ClassficationWorkflowsAgent/Mathematica/ClConTranslator.m"]
```

Get data

Get the Titanic data (from WL’s repository):

In[37]:=
```
dsTitanic = GetMachineLearningDataset["Titanic"];
```

Generate a classification pipeline

Generate a ClCon pipeline from a sequence of natural language commands:

In[38]:=
```
clCommands = {
   "load the titanic data",
   "split the data with 65 percent for training", "summarize data", "train a random forest classifier",
   "show classifier information",
   "display classifier training time", "show accuracy, precision, recall, and area under roc curve", "display confusion matrix plot",
   "compute the variable importance estimates"};
pl = ToClConPipelineFunction[clCommands]
```

Out[39]=
```
Function[{x, c}, (((((((ClConUnit[x, c] ⟹ ClConSplitData[0.65]) ⟹ ClConEchoFunctionValue[summaries:, (Multicolumn[#1, 5] &) /@ RecordsSummary /@ #1 &]) ⟹ ClConMakeClassifier[RandomForest]) ⟹
            ClConEchoFunctionContext[classifier info:, If[AssociationQ[#1[classifier]], ClassifierInformation /@ #1[classifier], ClassifierInformation[#1[classifier]]] &]) ⟹
          ClConEchoFunctionContext[classifier property "TrainingTime" :, If[AssociationQ[#1[classifier]], (ClassifierInformation[#1, TrainingTime] &) /@ #1[classifier],
             ClassifierInformation[#1[classifier], TrainingTime]] &]) ⟹
        Function[{x$, c$}, ClConUnit[x$, c$] ⟹ ClConClassifierMeasurements[{Accuracy, Precision, Recall, AreaUnderROCCurve}] ⟹ ClConEchoValue]) ⟹
      Function[{x$, c$}, ClConUnit[x$, c$] ⟹ ClConClassifierMeasurements[{ConfusionMatrixPlot}] ⟹ ClConEchoValue]) ⟹
    Function[{x, c}, ClConUnit[x, c] ⟹ ClConAccuracyByVariableShuffling[] ⟹ ClConEchoValue]]
```

Run the generated pipeline

Run the generated pipeline over the Titanic data:

In[40]:=
```
ClConUnit[dsTitanic] ⟹ pl;
```

» summaries: ⟨| trainingData → …, testData → … |⟩

trainingData →

1 id		2 passengerClass		3 passengerAge		4 passengerSex		5 passengerSurvival	
Min	1	3rd	468	Min	-1	male	547	died	525
1st Qu	323	1st	213	1st Qu	10	female	303	survived	325
Mean	656.116	2nd	169	Median	20				
Median	667.5			Mean	23.3659				
3rd Qu	988			3rd Qu	40				
Max	1306			Max	80				

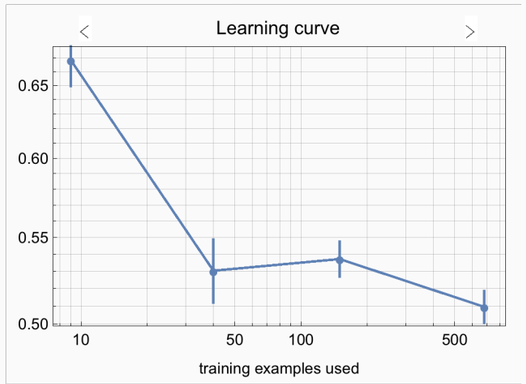
testData →

1 id		2 passengerClass		3 passengerAge		4 passengerSex		5 passengerSurvival	
Min	6	3rd	241	Min	-1	male	296	died	284
1st Qu	337.75	1st	110	1st Qu	10	female	163	survived	175
Mean	631	2nd	108	Median	20				
3rd Qu	652.932			Mean	23.8911				
Max	977.5			3rd Qu	40				
	1309			Max	70				

» classifier info:

Classifier information

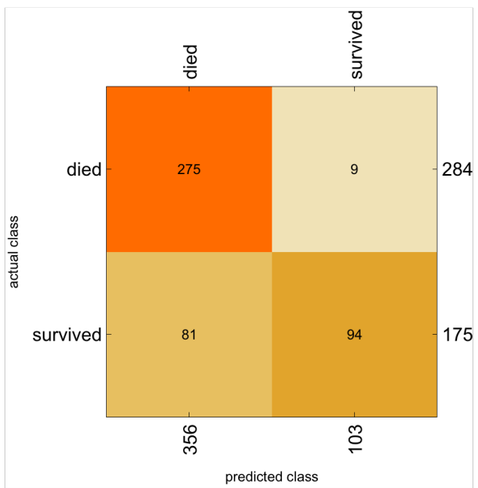
Input type	Mixed (number: 4)
Classes	died, survived
Method	RandomForest
Accuracy	77.2% ± 1.0%
Loss	0.510 ± 0.0088
Single evaluation time	4.21 ms/example
Batch evaluation speed	40.4 examples/ms
Classifier memory	257. kB
Training examples used	850 examples
Training time	1.82 s



» classifier property "TrainingTime" : 1.82636 s

» value: ⟨|Accuracy → 0.803922, Precision → ⟨|died → 0.772472, survived → 0.912621|⟩, Recall → ⟨|died → 0.96831, survived → 0.537143|⟩, AreaUnderROCCurve → ⟨|died → 0.866398, survived → 0.824326|⟩|⟩

» value: ⟨|ConfusionMatrixPlot →



|⟩

» value: ⟨| None → 0.803922, id → 0.786492, passengerClass → 0.801743, passengerAge → 0.799564, passengerSex → 0.614379 |⟩

Trace run and ode-command table

Run the generated pipeline through TraceMonad in order to obtain tabulated correspondence between (1) the generated ClCon pipeline components and (2) the natural language commands used to generate them:

In[41]:=
```
(p =
   ClConUnit[dsTitanic] ⟹
    ToClConPipelineFunction[clCommands, "Trace" → True]) ⟹
 TraceMonadTakeGrid[]
```

» summaries: ⟨| trainingData →

1 id		2 passengerClass	3 passengerAge		4 passengerSex		5 passengerSurvival	
Min	1	3rd 459	Min	-1	male	549	died	525
1st Qu	328	1st 209	1st Qu	10	female	301	survived	325
Median	644.5	2nd 182	Median	20				
Mean	649.839		Mean	23.8612				
3rd Qu	969		3rd Qu	40				
Max	1309		Max	80				

, testData →

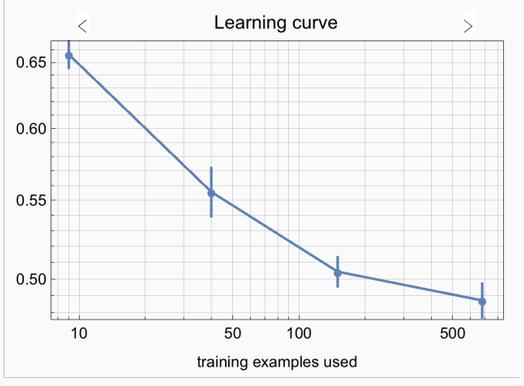
1 id		2 passengerClass	3 passengerAge		4 passengerSex		5 passengerSurvival	
Min	8	3rd 250	Min	-1	male	294	died	284
1st Qu	328	1st 114	1st Qu	0	female	165	survived	175
Mean	664.558	2nd 95	Median	20				
Median	681		Mean	22.9739				
3rd Qu	1001.5		3rd Qu	40				
Max	1306		Max	80				

|⟩

» classifier info:

Classifier information

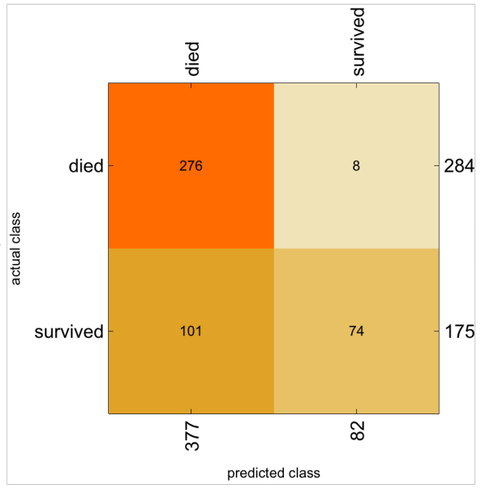
Input type	Mixed (number: 4)
Classes	died, survived
Method	RandomForest
Accuracy	79.7% ± 1.0%
Loss	0.487 ± 0.010
Single evaluation time	4.3 ms/example
Batch evaluation speed	39.3 examples/ms
Classifier memory	253. kB
Training examples used	850 examples
Training time	2.36 s



» classifier property "TrainingTime" : 2.36569 s

» value: ⟨| Accuracy → 0.762527, Precision → ⟨| died → 0.732095, survived → 0.902439 |⟩, Recall → ⟨| died → 0.971831, survived → 0.422857 |⟩, AreaUnderROCCurve → ⟨| died → 0.826781, survived → 0.765614 |⟩ |⟩

» value: ⟨| ConfusionMatrixPlot →



|⟩

» value: ⟨| None → 0.762527, id → 0.753813, passengerClass → 0.771242, passengerAge → 0.751634, passengerSex → 0.616558 |⟩

Out[41]=

`ClConUnit[x, c] ⟹`	
`  ClConSplitData[0.65`] ⟹`	split the data with 65 percent for training
`  ClConEchoFunctionValue["summaries:", (Multicolumn[#1, 5] &) /@ RecordsSummary /@ #1 &] ⟹`	summarize data
`  ClConMakeClassifier["RandomForest"] ⟹`	train a random forest classifier
`  ClConEchoFunctionContext["classifier info:",` `  If[AssociationQ[#1["classifier"]], ClassifierInformation /@ #1["classifier"], ClassifierInformation[#1["classifier"]]] &] ⟹`	show classifier information

`  ClConEchoFunctionContext["classifier property "TrainingTime" :",`
`  If[AssociationQ[#1["classifier"]], (ClassifierInformation[#1, "TrainingTime"] &) /@ #1["classifier"],`
`  ClassifierInformation[#1["classifier"], "TrainingTime"]] &] ⟹`
	display classifier training time

`  Function[{x$, c$}, ClConUnit[x$, c$] ⟹` `  ClConClassifierMeasurements[{"Accuracy", "Precision", "Recall", "AreaUnderROCCurve"}] ⟹ ClConEchoValue] ⟹`	show accuracy, precision, recall, and area under roc curve
`  Function[{x$, c$}, ClConUnit[x$, c$] ⟹ ClConClassifierMeasurements[{"ConfusionMatrixPlot"}] ⟹ ClConEchoValue] ⟹`	display confusion matrix plot
`  Function[{x, c}, ClConUnit[x, c] ⟹ ClConAccuracyByVariableShuffling[] ⟹ ClConEchoValue]`	compute the variable importance estimates