Item 112
git_comments:

git_commits:

1. **summary:** [SPARK-8764] [ML] string indexer should take option to handle unseen values
   message: [SPARK-8764] [ML] string indexer should take option to handle unseen values As a precursor to adding a public constructor add an option to handle unseen values by skipping rather than throwing an exception (default remains throwing an exception), Author: Holden Karau <holden@pigscanfly.ca> Closes #7266 from holdenk/SPARK-8764-string-indexer-should-take-option-to-handle-unseen-values and squashes the following commits: 38a4de9 [Holden Karau] fix long line 045bf22 [Holden Karau] Add a second b entry so b gets 0 for sure 81dd312 [Holden Karau] Update the docs for handleInvalid param to be more descriptive 7f37f6e [Holden Karau] remove extra space (scala style) 414e249 [Holden Karau] And switch to using handleInvalid instead of skipInvalid 1e53f9b [Holden Karau] update the param (codegen side) 7a22215 [Holden Karau] fix typo 100a39b [Holden Karau] Merge in master aa5b093 [Holden Karau] Since we filter we should never go down this code path if getSkipInvalid is true 75ffa69 [Holden Karau] Remove extra newline d69ef5e [Holden Karau] Add a test b5734be [Holden Karau] Add support for unseen labels afecd4e [Holden Karau] Add a param to skip invalid entries.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-8764][ML] string indexer should take option to handle unseen values
   body: As a precursor to adding a public constructor add an option to handle unseen values by skipping rather than throwing an exception (default remains throwing an exception),

github_pulls_comments:

1. [Test build #36715 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/36715/console) for PR 7266 at commit [`d69ef5e`](https://github.com/apache/spark/commit/d69ef5e183ec6a709c903316ca7e4d9c11b80726). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
2. [Test build #36848 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/36848/console) for PR 7266 at commit [`75ffa69`](https://github.com/apache/spark/commit/75ffa6986dc18bae6ee3a749e7b5e33b413cde2f). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
3. [Test build #36964 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/36964/console) for PR 7266 at commit [`aa5b093`](https://github.com/apache/spark/commit/aa5b093d3d36d539cff454b149b148d857f816e8). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
4. LGTM
5. cc @jkbradley for review since created the jira.
6. [Test build #37544 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/37544/console) for PR 7266 at commit [`100a39b`](https://github.com/apache/spark/commit/100a39bbda0cd1d046010f26d5b7dd5272ca1ae4). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class UnresolvedAttribute(nameParts: Seq[String]) extends Attribute` - `abstract class Star extends LeafExpression with NamedExpression` - `case class UnresolvedAlias(child: Expression) extends UnaryExpression with NamedExpression` - `case class SortOrder(child: Expression, direction: SortDirection) extends UnaryExpression` - `trait AggregateExpression extends Expression` - `trait PartialAggregate extends AggregateExpression` - `case class Min(child: Expression) extends UnaryExpression with PartialAggregate` - `case class Max(child: Expression) extends UnaryExpression with PartialAggregate` - `case class Count(child: Expression) extends UnaryExpression with PartialAggregate` - `case class Average(child: Expression) extends UnaryExpression with PartialAggregate` - `case class Sum(child: Expression) extends UnaryExpression with PartialAggregate` - `case class SumDistinct(child: Expression) extends UnaryExpression with PartialAggregate` - `case class First(child: Expression) extends UnaryExpression with PartialAggregate` - `case class Last(child: Expression) extends UnaryExpression with PartialAggregate` - `trait Generator extends Expression` - `case class Explode(child: Expression) extends UnaryExpression with Generator` - `trait NamedExpression extends Expression` - `abstract class Attribute extends LeafExpression with NamedExpression` - `case class PrettyAttribute(name: String) extends Attribute` - `abstract class LeafNode extends LogicalPlan` - `abstract class UnaryNode extends LogicalPlan`
7. jenkins, retest this please.
8. @jkbradley if you have a chance to look at this PR too its in the same class/file as the last one.
9. [Test build #186 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SlowSparkPullRequestBuilder/186/console) for PR 7266 at commit [`100a39b`](https://github.com/apache/spark/commit/100a39bbda0cd1d046010f26d5b7dd5272ca1ae4). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
10. [Test build #39348 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/39348/console) for PR 7266 at commit [`100a39b`](https://github.com/apache/spark/commit/100a39bbda0cd1d046010f26d5b7dd5272ca1ae4). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
11. I was taking another look at it, and I like the setup. But one thing I had not thought of was a third option: creating a new label/index which all unseen values are mapped to. That would let users avoid filtering out rows, and instead replace the bad/unseen values with a default value. Rather than putting that in this PR, could you modify the Param to be a String, so that we can specify other options in the future?

12. [Test build #39952 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/39952/console) for PR 7266 at commit [`414e249`](https://github.com/apache/spark/commit/414e249de422d718d33bb9e1b04525704f3e8a30). - This patch **fails Scala style tests**. - This patch merges cleanly. - This patch adds no public classes.
13. [Test build #39957 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/39957/console) for PR 7266 at commit [`7f37f6e`](https://github.com/apache/spark/commit/7f37f6e0cd5cc377f2d9efae0ba1e3906fec58ea). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
14. Those 2 small items are the only issues I see.
15. [Test build #40054 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/40054/console) for PR 7266 at commit [`045bf22`](https://github.com/apache/spark/commit/045bf22ab1852bd2f3d2403a2202a558f9eda911). - This patch **fails Scala style tests**. - This patch merges cleanly. - This patch adds no public classes.
16. [Test build #40059 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/40059/console) for PR 7266 at commit [`38a4de9`](https://github.com/apache/spark/commit/38a4de9ea4a04e3765b81b54479e80157630a13d). - This patch **fails to build**. - This patch merges cleanly. - This patch adds no public classes.
17. Catalyst failure seems likely unrelated, jenkins retest this please.
18. [Test build #258 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SlowSparkPullRequestBuilder/258/console) for PR 7266 at commit [`38a4de9`](https://github.com/apache/spark/commit/38a4de9ea4a04e3765b81b54479e80157630a13d). - This patch **fails to build**. - This patch merges cleanly. - This patch adds no public classes.
19. [Test build #40068 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/40068/console) for PR 7266 at commit [`38a4de9`](https://github.com/apache/spark/commit/38a4de9ea4a04e3765b81b54479e80157630a13d). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
20. LGTM, merging with master. Thanks!
21. For me it would be usefull(when working with trees) if string indexer was mapping unseen labels to maximum known label+1. The row would not be skipped if it contained some non-used feature for classification.
22. @miro-balaz : This probably isn't the best place for a new feature request - but if you head over to the ASF JIRA you can create a new ticket and cc the people who worked on this.
23. thank you for directions On Monday, 12 September 2016, Holden Karau notifications@github.com wrote: > @miro-balaz https://github.com/miro-balaz : This probably isn't the > best place for a new feature request - but if you head over to the ASF JIRA > you can create a new ticket and cc the people who worked on this. > > — > You are receiving this because you were mentioned. > Reply to this email directly, view it on GitHub > https://github.com/apache/spark/pull/7266#issuecomment-246216639, or mute > the thread > https://github.com/notifications/unsubscribe-auth/ALaD0wJm0im_ycl7fyxq-c97bY3RBCXkks5qpKJQgaJpZM4FT5k- > .
24. When "skip" is chosen as the way to handle Unseen labels, is there a way to know which rows were skipped?

github_pulls_reviews:

1. remove newline
2. I think that `HasSkipInvalid` should be mixed-in with `StringIndexerModel` rather than `StringIndexerBase`. Skipping invalid is really a parameter that can be set for the resulting model and is not used by `StringIndexer#fit` except in `copyValues`
3. Why?
4. Remove if `HasSkipInvalid` is localized to `StringIndexerModel`
5. I'm probably just missing something but looking at the other params it seems that they are able to be set on the trainer and the mode so I figured I should keep with that style.
6. I don't think the two different exceptions add any new information (the user already knows if the model is configured to skip invalid or not). Also, this branch should never execute because of L152. Perhaps we can collapse this into a single exception?
7. Do you have a specific example in mind? My suggestion was because `SkipInvalid` has nothing to do with the fitting routine itself; rather, it affects how the model is evaluated. Looking at [other models](https://github.com/apache/spark/blob/master/mllib/src/main/scala/org/apache/spark/ml/regression/GBTRegressor.scala#L91), the setters for params exist exclusively in the `Predictor` or in the `PredictionModel` but not both.
8. Discussion on L69.
9. Looking at the other params in the StringIndexer model as well as LogisticRegression (e.g. setThreshold) it seems that when a param makes sense to be set on the model it is exposed in both the model and the trainer.
10. That makes sense, thanks for pointing it out me!
11. OK as is, sorry for my mixup
12. typo
13. Please make docs more explicit. It's very helpful to users. Here, it'd be nice to state the options and what they mean: "Options: skip (filter out rows with bad values) or error (throw an error). Note that skipping rows means the Transformer can output a smaller dataset."
14. I'd create 1 extra record with value "b" here to make sure that StringIndexer chooses "b" to have index 0.

jira_issues:

jira_issues_comments: