Item 243
git_comments:

1. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
2. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
3. * * Returns the predicted number of objects in the queue to occur at the next configured interval (5 mins in the future, e.g.). * * @return milliseconds until backpressure is predicted to occur, based on the number of bytes in the queue.
4. times is the X axis and counts is on the y axis
5. **comment:** TODO: Should change keys used here
   label: code-design
6. * * Returns the predicted time (in milliseconds) when backpressure is expected to be applied to this connection, based on the total number of bytes in the queue. * * @return milliseconds until backpressure is predicted to occur, based on the total number of bytes in the queue.
7. * * Returns the predicted time (in milliseconds) when backpressure is expected to be applied to this connection, based on the number of objects in the queue. * * @return milliseconds until backpressure is predicted to occur, based on the number of objects in the queue.
8. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
9. * * Returns all available predictions
10. * * Returns the predicted total number of bytes in the queue to occur at the next configured interval (5 mins in the future, e.g.). * * @return milliseconds until backpressure is predicted to occur, based on the total number of bytes in the queue.
11. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.

git_commits:

1. **summary:** NIFI-6510 Analytics Framework Introduction (#10)
   message: NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated

query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes

github_issues:

github_issues_comments:

github_pulls:

1. **title:** NIFI - 121
   body: When MergeContent is configured to perform Binary Concatenation, if generating bundle of 1 FlowFile, should just Clone original FLowFile instead of copying data This is more efficient.
   label: code-design
2. **title:** NIFI - 121
   body: When MergeContent is configured to perform Binary Concatenation, if generating bundle of 1 FlowFile, should just Clone original FLowFile instead of copying data This is more efficient.

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
   label: code-design
2. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of

or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

3. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
4. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
5. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data

which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

6. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
7. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information

either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

8. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
9. **summary:** Predictive Analytics for NiFi Metrics
   description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
10. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect

anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

11. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
12. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of

the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

13. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
14. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
15. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this

topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

16. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
17. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with

the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

18. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
19. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
20. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to

help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

21. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
22. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
23. **summary:** Predictive Analytics for NiFi Metrics

description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

24. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
25. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support

both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

26. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
27. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
28. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems).

Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

29. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
30. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where

applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

31. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
32. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
33. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze

these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

34. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
35. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing

model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

36. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
    label: code-design
37. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
    label: code-design
38. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide

meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

39. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

40. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of

the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

41. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
42. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
43. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect

anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

44. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
45. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where

applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

46. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
47. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
48. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems).

Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

49. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
50. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps

later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

51. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
52. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
53. **summary:** Predictive Analytics for NiFi Metrics

description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

54. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

55. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several

key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

56. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
57. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

label: code-design

58. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
    label: code-design
59. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
    label: code-design
60. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users

reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

61. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
62. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with

the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

63. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
64. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
65. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide

meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

66. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

67. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of

the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

68. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
69. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
70. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect

anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

71. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
72. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where

applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design

73. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
label: code-design
74. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}
75. **summary:** Predictive Analytics for NiFi Metrics
description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK

stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

76. **summary:** Predictive Analytics for NiFi Metrics
    description: From Yolanda's email to the list: {noformat} Currently NiFi has lots of metrics available for areas including jvm and flow component usage (via component status) as well as provenance data which NiFi makes available either through the UI or reporting tasks (for consumption by other systems). Past discussions in the community cite users shipping this data to applications such as Prometheus, ELK stacks, or Ambari metrics for further analysis in order to capture/review performance issues, detect anomalies, and send alerts or notifications. These systems are efficient in capturing and helping to analyze these metrics however it requires customization work and knowledge of NiFi operations to provide meaningful analytics within a flow context. In speaking with Matt Burgess and Andy Christianson on this topic we feel that there is an opportunity to introduce an analytics framework that could provide users reasonable predictions on key performance indicators for flows, such as back pressure and flow rate, to help administrators improve operational management of NiFi clusters. This framework could offer several key features: - Provide a flexible internal analytics engine and model api which supports the addition of or enhancement to onboard models - Support integration of remote or cloud based ML models - Support both traditional and online (incremental) learning methods - Provide support for model caching (perhaps later inclusion into a model repository or registry) - UI enhancements to display prediction information either in existing summary data, new data visualizations, or directly within the flow/canvas (where applicable) For an initial target we thought that back pressure prediction would be a good starting point for this initiative, given that back pressure detection is a key indicator of flow performance and many of the metrics currently available would provide enough data points to create a reasonable performing model. We have some ideas on how this could be achieved however we wanted to discuss this more with the community to get thoughts about tackling this work, especially if there are specific use cases or other factors that should be considered.{noformat}

jira_issues_comments:

1. **body:** Commit f3824a0661ae7c32a3b5601ba75e4ec5a11e8f4a in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=f3824a0] NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes
   label: code-design
2. Commit 8a48c779d1e7c63caf404103851ccdff72246a7c in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a48c77] NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d)
3. Commit 6c5eda124b2b77c62ca455d66230a08f4befec5e in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=6c5eda1] NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058)

4. Commit 0bdedd5931f7652a8bf3f89e04d55a11a26b782b in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=0bdedd5] NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b)
5. Commit 0c9e8edffbd1e4270502be9bee9570047f314321 in nifi's branch refs/heads/analytics-framework from Matt Burgess [https://gitbox.apache.org/repos/asf?p=nifi.git;h=0c9e8ed] NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics
6. Commit 586185e4e5b72ffbb39dd4f6970df713a727818b in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=586185e] NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b)
7. Commit 714b3f6a3fdafa4ddcf0740f102a9e87d45da041 in nifi's branch refs/heads/analytics-framework from Matt Burgess [https://gitbox.apache.org/repos/asf?p=nifi.git;h=714b3f6] NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11)
8. Commit c98ae35ddcdbbb56e79fdedd2f39bb21ef8d417e in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=c98ae35] NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058)
9. Commit 59147a387432015cc5869bb0b5b7d0b11ab0b0aa in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=59147a3] NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b)
10. Commit 94b120d102c08741480b4ad04898b287813a1f2d in nifi's branch refs/heads/analytics-framework from Matt Burgess [https://gitbox.apache.org/repos/asf?p=nifi.git;h=94b120d] NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics
11. Commit f093f48bdf7f9d37269f00010764f898b1d7e1fe in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=f093f48] NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b)
12. Commit 6736574284b4bb6985a3d2508a86b54acc389f02 in nifi's branch refs/heads/analytics-framework from Matt Burgess [https://gitbox.apache.org/repos/asf?p=nifi.git;h=6736574] NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11)
13. Commit 93ca7a0c891d41b09f218d7f016e19fdee772201 in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=93ca7a0] NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b)
14. Commit d1e5cc7c13a9f6834ee6c8a69813c23a17dac66e in nifi's branch refs/heads/analytics-framework from Matt Burgess [https://gitbox.apache.org/repos/asf?p=nifi.git;h=d1e5cc7] NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics
15. Commit b3f362bc997e7f9387253cb16d8ba3a9b8f10947 in nifi's branch refs/heads/analytics-framework from Yolanda Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=b3f362b] NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b)
16. Commit 79315c89815f50bc6a08cf4df9fd304bc775c3f8 in nifi's branch refs/heads/analytics-framework from Matt Burgess [https://gitbox.apache.org/repos/asf?p=nifi.git;h=79315c8] NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11)
17. Commit 146af158182f764fa6d88517e81ad0d76c67616b in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=146af15] NIFI-6510 - add an enable/disable property for analytics
18. Commit 7e6eddd0570ecceeb8e14e927cf257c731375fc9 in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=7e6eddd] NIFI-6510 - documentation updates for enable/disable property

19. Commit f8e743b620854cf8247cf64ae013e98286781f32 in nifi's branch refs/heads/analytics-framework from Rob Fellows [https://gitbox.apache.org/repos/asf?p=nifi.git;h=f8e743b] NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685)
20. Commit 134781d1b2b97185a6c272fb7c78a046dbf1c48d in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=134781d] NIFI-6510 - admin guide updates to further describe model functionality
21. Commit 17c9cc2d5836f171d53ddceb423c7278220ca95e in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=17c9cc2] NIFI-6510 - code quality fixes (if statement and constructor)
22. Commit 69b486a94f6fa8a7bbcceb2941e14b5efd50c347 in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=69b486a] NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold
23. Commit e5738fd557bcd43b9c3fc3b555dd43bfb66c387e in nifi's branch refs/heads/analytics-framework from Andrew I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=e5738fd] NIFI-6510 Extract out predictions into their own DTO
24. Commit 39ff5340f3808abad6f26d28f620a5df25ba6e43 in nifi's branch refs/heads/analytics-framework from Andrew I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=39ff534] NIFI-6510 Optimize imports
25. Commit 7345ee6b55752431e7d926c7bb831147543f9b14 in nifi's branch refs/heads/analytics-framework from Andrew I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=7345ee6] NIFI-6510 Fix formatting
26. Commit 5bb185050c9fd4c29041493f8b41afbdd8a6b3a6 in nifi's branch refs/heads/analytics-framework from Andrew I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=5bb1850] NIFI-6510 Optimize imports
27. Commit de9e2cbb72ca24f979dc89b9ee283686437bc68d in nifi's branch refs/heads/analytics-framework from Andrew I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=de9e2cb] NIFI-6510 Optimize imports
28. Commit fcb31f66b46b232c2bc54af8ba5e44cc71c5fa6c in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=fcb31f6] NIFI-6510 - Notice updates for Commons math and Caffeine
29. Commit 9292005a723c308978894549eb1edf0e3152ac1b in nifi's branch refs/heads/analytics-framework from Rob Fellows [https://gitbox.apache.org/repos/asf?p=nifi.git;h=9292005] NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697)
30. Commit 60d9ce427c2b1af9b291d09e19b47de4edeba753 in nifi's branch refs/heads/analytics-framework from Rob Fellows [https://gitbox.apache.org/repos/asf?p=nifi.git;h=60d9ce4] NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705
31. Commit 6ed10bf6685235d1e52413c75459bb2e5906bcde in nifi's branch refs/heads/analytics-framework from Andrew I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=6ed10bf] NIFI-6510 Rip out useless members
32. Commit 7713be3ce8a7c137808df47a1a1465444d2a77a0 in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=7713be3] NIFI-6510 - dto updates to check for -1 value
33. Commit ab26522390963e4f0189cb77828baeb50a22ab0e in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=ab26522] NIFI-6510 - checkstyle fix
34. Commit 4ff359abfc2b68a52522f722d15b5c2a9f91d288 in nifi's branch refs/heads/analytics-framework from Yolanda M. Davis [https://gitbox.apache.org/repos/asf?p=nifi.git;h=4ff359a] NIFI-6510 - rolled back last change and applied minNonNegative method
35. Commit 16f7c37f05aa722a748d309297c4b0b92675c153 in nifi's branch refs/heads/analytics-framework from Andrew I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=16f7c37] NIFI-6510 Rip out useless members
36. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and

interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

37. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for

checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

38. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction

model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

39. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects *

NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

40. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear

regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

41. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics

framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

42. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

43. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-

6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

44. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out

useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

45. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known.

Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

46. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying

estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

47. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to

account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

48. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize

imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

49. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-

6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

50. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect

property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

51. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement

and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

52. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 -

admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

53. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property *

NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

54. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an

enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

55. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to

back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

56. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data

displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

57. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks.

* add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

58. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. *

Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

59. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial

predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

60. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by:

Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

61. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected

model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

62. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use

mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

63. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-

by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

64. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable

name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

65. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-

6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

66. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from

nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

67. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 -

Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

68. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added

property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

69. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model

fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

70. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that

include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members
label: code-design

71. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses

multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

72. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted interval and incorporated R-squared check Updates to support multiple variables for features, clearing

cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

73. **body:** Commit 8a8b9c1d086ac41b10647f09bdd7d8174a921de5 in nifi's branch refs/heads/master from Andy I. Christianson [https://gitbox.apache.org/repos/asf?p=nifi.git;h=8a8b9c1] NIFI-6510 - Analytics framework (#3681) * NIFI-6510 Implement initial analytic engine * NIFI-6510 Implemented basic linear regression model for queue counts * NIFI-6510 Initial analytics REST endpoint and supporting objects * NIFI-6510 Connect the dots for StatusAnalytics -> API * NIFI-6510 Added poc engine with prediction model caching (cherry picked from commit e013b91) DFA-9 - updated logging and corrected logic for checking if not in backpressure (cherry picked from commit a1f8e70) * NIFI-6510 Updated objects and interfaces to reflect 4 prediction metrics (cherry picked from commit 050e0fc) (cherry picked from commit 9fd365f) * NIFI-6510 adjustments for interface updates, added call to StandardEventAccess, updated interface to use connection id (cherry picked from commit 14854ff) DFA-9 - reduced snapshot interval to 1 minute (cherry picked from commit 36abb0a) * NIFI-6510 Split StatusAnalytics interface into Engine and per-Connection versions * NIFI-6510 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly * NIFI-6510 Revert "DFA-9 Remove redundant connection prediction interfaces as we can just use ConnectionStatusAnalytics directly" This reverts commit 5b9fead1471059098c0e98343fb337070f1c75c1. * NIFI-6510 Added prediction fields for use by UI, still need to be populated * NIFI-6510 Analytics Framework Introduction (#10) * DFA-9 - Initial refactor for Status Analytics - created additional interfaces for models, refactored callers to use StatusAnalytics objects with connection context. Implemented SimpleRegression model. DFA-9 - added logging * DFA-9 - relocated query window to CSA from model, adding the prediction percentages and time interval * DFA-9 - checkstyle fixes * NIFI-6510 Add prediction percent values and predicted interval seconds (cherry picked from commit e60015d) * NIFI-6510 Changes to inject flowManager instead of flow controller, also changes to properly reflect when predictions can be made vs not. (cherry picked from commit 6fae058) * NIFI-6510 Added tests for engine (cherry picked from commit 6d7a13b) * NIFI-6150 Added tests for connection status analytics class, corrected variable names (cherry picked from commit 58c7c81) * NIFI-6150 Make checkstyle happy (cherry picked from commit b6e35ac) * NIFI-6150 Fixed NaN check and refactored time prediction. Switched to use non caching engine for testing * NIFI-6510 Fixed checkstyle issue in TestConnectionStatusAnalytics * NIFI-6510 Adjusted

interval and incorporated R-squared check Updates to support multiple variables for features, clearing cached regression model based on r-squared values Added ordinary least squares model, which truly uses multivariable regression. Refactor of interfaces to include more general interface for variate models (that include scoring support). Ratcheck fixes Added test for SimpleRegression. Minor fix for OLS model fixed test errors fixed checkstyle errors (cherry picked from commit fab411b) * NIFI-6510 Added property to nifi.properties - Prediction Interval for connection status analytics (#11) * NIFI-6566 - Refactor to decouple model instance from status analytics object. Also allow configurable model from nifi.properties NIFI-6566 - changes to allow scoring configurations for model in nifi.properties NIFI-6566 - added default implementation value to NiFiProperties NIFI-6566 - correction to default variable name in NiFiProperties, removed unnecessary init method from ConnectionStatusAnalytics Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3663 * NIFI-6585 - Refactored tests to use mocked models and extract functions. Added check in ConnectionStatusAnalytics to confirm expected model by type * NIFI-6586 - documentation and comments This closes NIFI-6586 Signed-off-by: Andrew I. Christianson <andy@andyic.org> * NIFI-6568 - Surface time-to-back-pressure and initial predictions in the UI * Add multi-line tooltips with detail for connection queue back pressure graphics. * Add estimated time to back pressure to connections summary table. * Add back pressure prediction ticks. * add moment.js to format predicted time to back pressure * tweak summary table headings to match data displayed. re-order connection summary columns * NIFI-6568 - Properly sort the min estimated time to back pressure in the connection summary table. Also added a js doc comment. * NIFI-6510 - add an enable/disable property for analytics * NIFI-6510 - documentation updates for enable/disable property * NIFI-6510 - UI: handle the scenario where backpressure predictions are disabled (#3685) * NIFI-6510 - admin guide updates to further describe model functionality * NIFI-6510 - code quality fixes (if statement and constructor) * NIFI-6510 - log warnings when properties could not be retrieved. fixed incorrect property retrieval for score threshold * NIFI-6510 Extract out predictions into their own DTO * NIFI-6510 Optimize imports * NIFI-6510 Fix formatting * NIFI-6510 Optimize imports * NIFI-6510 Optimize imports * NIFI-6510 - Notice updates for Commons math and Caffeine * NIFI-6510 - UI updates to account for minor API changes for back pressure predictions (#3697) * NIFI-6510 - Fix issue displaying estimated time to back pressure in connection summary table when only one of the predictions is known. Signed-off-by: Matthew Burgess <mattyb149@apache.org> This closes #3705 * NIFI-6510 Rip out useless members * NIFI-6510 - dto updates to check for -1 value * NIFI-6510 - checkstyle fix * NIFI-6510 - rolled back last change and applied minNonNegative method * NIFI-6510 Rip out useless members

label: code-design

74. This feature was merged 9/9/2019