



XRay

Machine Learning Guide

Version 1.6

Document Number: ML003

Document Title: XRay Machine Learning Guide

Document Release Date: December 2023

Document Number: ML003

Published by:

Research & Development

meshIQ

88 Sunnyside Blvd, Suite 101

Plainview, NY 11803

Copyright © 2023 meshIQ. All rights reserved. No part of the contents of this document may be produced or transmitted in any form, or by any means without the written permission of meshIQ.

Confidentiality Statement: The information within this media is proprietary in nature and is the sole property of meshIQ. All products and information developed by meshIQ are intended for limited distribution to authorized meshIQ employees, licensed clients, and authorized users. This information (including software, electronic and printed media) is not to be copied or distributed in any form without the expressed written permission from meshIQ.

Acknowledgements: The following terms are trademarks of meshIQ in the United States or other countries or both: AutoPilot, AutoPilot M6, M6 Web Server, M6 Web Console, M6 for WMQ, MQControl, Navigator, XRay.

The following terms are trademarks of the IBM Corporation in the United States or other countries or both: IBM, MQ, WebSphere MQ, WIN-OS/2, AS/400, OS/2, DB2, Informix, AIX, and z/OS.

Java, J2EE, and the Java Logos are trademarks of Sun Microsystems Inc. in the United States or other countries, or both.

InstallAnywhere is a trademark or registered trademark of Flexera Software, Inc.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>), including Derby Database Server. The Jakarta Project" and "Tomcat" and the associated logos are registered trademarks of the Apache Software Foundation.

Intel, Pentium and Intel486 are trademarks or registered trademarks of Intel Corporation in the United States, or other countries, or both.

Microsoft, Windows, Windows NT, Windows XP, the Windows logos, Microsoft SQL Server, and Microsoft Visual SourceSafe are registered trademarks of the Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Mac, Mac OS, and Macintosh are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.

"Linux" and the Linux Logos are registered trademarks of Linus Torvalds, the original author of the Linux kernel. All other titles, applications, products, and so forth are copyrighted and/or trademarked by their respective authors.

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates.

Other company, product, and service names may be trademarks or service marks of others.

Contents

- CHAPTER 1: INTRODUCTION..... 1**
 - 1.1 HOW THIS GUIDE IS ORGANIZED 1
 - 1.1 HISTORY OF THIS DOCUMENT 1
 - 1.2 USER FEEDBACK..... 1
 - 1.4 RELEASE NOTES 1
 - 1.3 INTENDED AUDIENCE 2
 - 1.4 TECHNICAL SUPPORT 2

- CHAPTER 2: SETUP 3**
 - 2.1 DATA TYPE AND DATA FORMAT RESTRICTIONS 3
 - 2.2 HOW TO GET SET UP FOR MACHINE LEARNING 3
 - 2.3 TRAINING A MODEL 3

- CHAPTER 3: QUERIES 7**
 - 3.1 ML ANALYTIC QUERIES 7
 - 3.2 EXPLANATION OF QUERIES 11
 - 3.3 REAL-TIME PREDICTIONS AND FORECASTS..... 12

- CHAPTER 4: TROUBLESHOOTING 14**
 - 4.1 WHAT CAN GO WRONG WHILE TRAINING? 14
 - 4.2 WHAT CAN GO WRONG WHILE RUNNING MACHINE LEARNING QUERIES?..... 16
 - 4.3 WHAT CAN GO WRONG WITH REAL-TIME PREDICTIONS?..... 21

Chapter 1: Introduction

Machine learning is defined as “the use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyze and draw inferences from patterns in data.” Machine Learning provides insights into data that would not normally be seen by simply viewing the data. There are two types of Machine Learning that can be run:

Supervised Learning – Requires a Machine Learning “model.” A Machine Learning model is a set of statistical equations that can detect patterns in data. The model gets created by being trained (or learning) via a set of historical data. There is a section in this document that discusses how to have XRay train a model.

Unsupervised Learning – No model is required.

That was a quick introduction to Machine Learning. There is a myriad of information on the internet that will explain Machine Learning in great detail if you wish to learn more about it. The purpose of the remainder of this document is to provide the technical information necessary to get meshIQ’s Machine Learning Solution up and running and to begin using it.

1.1 How this Guide is Organized

[Chapter 1:](#) Introduction

[Chapter 2:](#) Setting up your system for Machine Learning and model training.

[Chapter 3:](#) Learn about the various Machine Learning queries.

[Chapter 4:](#) Troubleshooting common errors encountered during model training and while running Machine Learning.

1.1 History of this Document

Table 1. Document History			
Release Date	Document Number	Version	Summary
March 2021	ML001	1.3	Initial release
December 2023	ML003	1.6	Updates for v1.6

1.2 User Feedback

meshIQ encourages all users and administrators to submit comments, suggestions, corrections, and recommendations for improvement for all documentation. Please send your comments via e-mail to: support@meshiq.com. You will receive a response, along with status of any proposed change, update, or correction.

1.4 Release Notes

For product release notes, see the [XRay release notes page](#) in the Resource Center.

1.3 Intended Audience

This guide is intended for systems administrators and operating engineers responsible for the installation and administration of the XRay environment.

1.4 Technical Support

Use one of the following methods for technical support:

- **Call:** 800-963-9822 ext. 1
If you are calling from outside the United States: 001-516-801-2100
- **Email:** mysupport@meshiq.com
- **Resource center:** <https://customers.meshiq.com>
- **Automated support system:** <http://mysupport.meshiq.com> (user ID and password required)

Chapter 2: Setup

2.1 Data Type and Data Format Restrictions

Please be advised that several Machine Learning functions will only run on “Datasets.” Probably the most difficult part of preparing to run Machine Learning will be preparing the data in the proper format. If data is maintained in activities, events, or snapshots, then a mechanism must be in place to move this data into datasets. Also, even if data is originally stored in a dataset, there is a high likelihood that the data will need to be manipulated in order to work properly with Machine Learning. Getting the data into a proper format for Machine Learning can be accomplished by the following mechanisms:

1. Change how data is streamed so that it is originally streamed into the datasets table in the proper format.
2. Use Views, Macros, or Stored procedures in order to get the data into the proper format and to copy it over to the dataset table.
3. Engage meshIQ’s technical team or another technical team to create code that massages the data into the proper format and stores it in the datasets table.

2.2 How to Get Set up for Machine Learning

1. Ensure you have at least one month’s worth of data.
2. Create an entry in the mlmodel table (see details in the next section of this document). This will initiate training.
3. Monitor the training:
 - The jobstatus field in the mlmodel table should change to “SCHEDULED” and then to “COMPLETED” if it is successful. If there is a failure, it will change to “FAILED”.
 - The jobs table will show model progress.
4. After the models have finished building, you can automatically create a Machine Learning Dashboard by doing the following:
 - Issue the query ‘get active models’.
 - Right-click on the results
 - Choose ‘Automate Dashboard’.

2.3 Training a model


If you wish to have Supervised Learning (described above), an entry must exist in the jKool “mlmodel” table. The XRay UI has a screen devoted to populating this table. It is available from the main menu. Select  > Admin Settings > ML Models. See the [ML Model Definitions \(Machine Learning\)](#) article in the Resource Center for more information. These screens will allow you to specify the following fields that are marked with “Computed” set to “No”. Fields with “Computed” set to “Yes” will be automatically populated by the system.

Table 2: mlmodel Table			
Field	Type	Description	Computed
Accuracy	Decimal	Accuracy of the model	Yes
Active	Boolean	True if the model is in use	No
AdminRole	String[]	Admin field – people with administrative rights to this table	No
AnomalyMargin	Decimal	Margin of error for a numeric value to be considered anomaly	No
AnomalyMarginType	Enum	Percent, Numeric	No
Criteria	String	Where condition	No
DatasetName	String	Datasets model can be run on	No
Description	String	Model description	No
Exception	String	Exceptions while building the model	Yes
EndTime	Timestamp	Time the model finished executed. This fields is informational only.	Yes
ElapsedTime	Timeinterval	Time it took to build the model	Yes
IndepVars	String[]	Independent Variables	No
IndepVarsFinal	String[]	Independent Variables after elimination due to zero variance	Yes
JobStatus	Enum	SCHEDULED,RUNNING,PAUSED,COMPLETED,FAILED,CANCELED	Yes
LogTaken	Boolean	If a log was taken while building (take log for predictions)	Yes
LookbackPeriod	String	Signifies the amount of data that will be used for training. Enter a string in the format that would be used in a JKQL query (i.e. last year, last 7 months). If this field is blank, the default will come from global.properties.	
Name	String	Name of the model	No
OptModelType	String	Type of model that performed the best	Yes
Owner	String	Admin field – the owner of this table	No
ReportingFields	String[]	Additional fields to return in query	No
Schedule	String	How often a model gets trained. Please	No

Table 2: mlmodel Table

Field	Type	Description	Computed
		see the JKQL User's Guide for details on using this field.	
Starttime	Timestamp	Time the model began building	Yes
TSField	String	If time-series, field we use to group and bucket	No
TSeries	Boolean	Indicates if a time-series model	No
TSTimeInterval	Enum	Minute,Hour,Day,Week,Month,Year	No
TargetVar	String	Target field	No
UpdateTime	Timestamp	Admin field - time the model entry was updated	Yes
UserRole	String[]	Admin field - users of the table	No
MaxDataDate	Timestamp	Necessary for time series model only. It is the maximum date of the data used to train the model. Forecasts will predict forward from this date.	Yes

The models will automatically rebuild themselves at the "Schedule" interval specified in the MLModel table. Please note that rebuilding is necessary because things change over time and the model needs to "re-learn." The log table and job table can be monitored for any errors that might have occurred during training and to see training progress. Also, if a train failed, the "jobstatus" field in the mlmodel table will be set to "FAILED."

In addition to models re-training themselves automatically, a train can be kicked off manually via a Train Statement (syntax below). This train statement will have no effect on the automatic training. For instance, if a model is scheduled to be rebuilt the first of every month and it was automatically built February 1st, then manually built February 10th, the next automatic build will be March 1st, not March 10th.

TRAIN MODEL '<model name>'

To determine the time models are scheduled to be automatically built, enter the following statement:

GET ACTIVE MODELS

With the exception of the following fields, all other fields returned in the 'get active models' query come from the mlmodel table: JobStatus, NextExecutedTime, Statistics. The difference between the JobStatus field in the 'get active model' query and the jobstatus field in the 'get mlmodels' query is described below.

If no schedule exists in the mlmodel table, then the model can only be trained manually, and you will see 'COMPLETED' jobstatus values in the 'get active model' query. However, if

the schedule field is populated in the mlmodel table, 'COMPLETED' will never be in the jobstatus field value for those rows in the *'get active model'* query. This is because even if a model just finished training, it is still scheduled for the 'next time' it will train.

Chapter 3: Queries

3.1 ML Analytic Queries

Below is a list of all the Machine Learning queries that can be run, along with their syntax. The XRay UI will automatically build a dashboard containing viewlets of these queries after you perform the following actions:

- Issue the query 'get active models'.
- Right-click on the results.
- Choose 'Automate Dashboard'.

These queries are meant to work in conjunction with the new 1.5 query "piping." Please see the JKQL User's Guide for details on piping.

optional required

expected

(expct)

<get statement> | compute expected(<model name>) where ...

- get datasets fields PETAL_LENGTH, PETAL_WIDTH, SEPAL_LENGTH, SEPAL_WIDTH where PETAL_LENGTH > 2 | compute expected('SPECIES') where SPECIES='versicolor'

featureSelection (any criteria is irrelevant)

(fselection, fsel)

compute featureselection(<model name>)

- compute featureSelection('SPECIES')

forecast (criteria of only one row required)

(fcst)

compute forecast(<model name>, <id>)

compute forecast(<model name>, <number of forecasts>)

- compute forecast('MaxResponseTime', "e62646e5-9765-11e9-91d4-7629afde2223")
- compute forecast('MaxResponseTime', 30)

whatif (any criteria is irrelevant)

(wi)

compute whatif(<model name>, <list of ivs with values>)

- compute whatif('SPECIES', 'PETAL_LENGTH=1.4', 'PETAL_WIDTH=0.2', 'SEPAL_LENGTH=4.9', 'SEPAL_WIDTH=3')

correlate - no models involved

(corr)

<get statement> | compute corr(<min corr>) where <criteria>

- get datasets fields PETAL_LENGTH, PETAL_WIDTH, SEPAL_LENGTH, SEPAL_WIDTH | compute corr(.5)

Please note that any non-numeric fields that were specified in the get statement will be ignored.

<min corr> is between 0 and 1. It denotes the minimum value that a row or column must have in order to be show. This excludes the 1 on the diagonal. For instance, in the below table the second row and the first column would be eliminated if the minimum value was .5.

.33	.70	.12	.8	1
.12	.22	.01	1	.08
.1	.5	1	.4	.3
.4	1	.3	.91	.88
1	.56	.5	.45	.2

featureSuggestion -no models involved
(fsuggestion, fsug)

<get statement> | compute **featuresuggestion**(<target>) where <criteria>

- get dataset fields Position,map('Organization'),Department | compute featuresuggestion(ExpenseTotal)

clusters - no models involved
(cl)

<get statement> | compute **clusters**(<numberOfClusterColumns, <number of clusters or auto (in quotes)>,<recluster boolean>) where* <criteria>

- get dataset fields SEPAL_WIDTH, SEPAL_LENGTH, PETAL_WIDTH, PETAL_LENGTH | compute clusters('4','3',true)
- <number of clusters or auto (in quotes)> - default is 'auto'
<recluster boolean> - default is false

clusterDetails
(cld)

compute **clustersdetails**(<topic id>)

- compute clusterdetails('09721689-e598-45a6-addc-02a46cd2ebea-1')

clusters3d - no models involved, used to build a 3D plotly chart
(cl3d)

<get statement> | compute **clusters3d**(<number of clusters or auto (in quotes)>,<recluster boolean>) where* <criteria>

- get dataset fields SEPAL_LENGTH, PETAL_WIDTH, PETAL_LENGTH, SEPAL_WIDTH | compute clusters3d('auto',true)
- <number of clusters or auto (in quotes)> - default is 'auto'
<recluster boolean> - default is false

Number of cluster columns will always be "3" in this function. That is why a the user does not specify. In this example, clusters will be built using PETAL_LENGTH, PETAL_WIDTH,

SEPAL_LENGT, SEPAL_WIDTH is an additional reporting field that will display if a point in a plotly cluster is clicked up.

*** If the cluster rebuild flag is set to false, it's imperative that the filter criteria in the get statement matches what was used to originally build the clusters and that the data the cluster was originally build upon has not changed. Otherwise, results will be inaccurate.**

holtwintersprediction - no models involved, indicate a time unit of '1' always
(holtwinters, hw)

<aggregated get statement> | compute holtwintersprediction(<periods in forecast>)
where <criteria>

- get dataset fields avg(cacheUsage) group by map('starttime') bucketed by 1 hour | compute holtwintersprediction(5)

extrapolate - no models involved, indicate a time unit of '1' always
(extrap)

<aggregated get statement> | compute extrapolate(<number of forecasts>) where
<criteria>

- get dataset fields avg(cacheUsage) group by map('starttime') bucketed by 1 hour | compute extrapolate(10)

anomaly - no models involved

<get statement fields are <anomaly field> and optional <time field> | compute anomaly(<anomaly algorithm*>) where <criteria>

***Anomaly algorithms include: IsolationForest, LocalOutlierFactor, EllipticEnvelope
EllipticEnvelope or null if you wish for it to take the default (LocalOutlierFactor)**

- get event fields DELIVERYFEE, starttime where DELIVERYFEE is not null |
compute anomaly('LocalOutlierFactor')

expectedanomaliesinrange
(EAIR)

compute expanomaliesinrange

(<anomaly starttime>,<anomaly endtime>, <dataset name>) where <criteria>

compute anomalydetectioninrange

(<prior period>, <dataset name>) where <criteria>

- compute expanomaliesinrange('2015-01-01 00:00:00','2022-09-01 00:00:00','Iris')
- compute EAIR('PRIOR_DAY','Iris')

Dates should be in the format: YYYY-MM-DD hh:mm:ss

<prior period> can be: PRIOR_DAY, PRIOR_HOUR, PRIOR_MINUTE, PRIOR_WEEK

allforecastanomaliesinrange
(FAIR)

compute fcanomaliesinrange

```
(<anomaly starttime>,<anomaly endtime>, <dataset name>) where <criteria>  
compute FCAIR
```

```
(<prior period>, <dataset name>) where <criteria>
```

- `compute allforecastanomaliesinrange('2015-01-01 00:00:00','2020-09-01 00:00:00','USDTUSD0xdac17f958d2ee523a2206206994597c13d831ec7Daily')`
- `compute FCAIR('PRIOR_DAY','USDTUSD0xdac17f958d2ee523a2206206994597c13d831ec7Daily')`

<prior period> can be PRIOR_DAY, PRIOR_HOUR, PRIOR_MINUTE, PRIOR_WEEK

rogueedges - no models involved, must run on activities only

```
<get statement> compute | rogueEdges('') where <criteria>
```

- `get activity where activityid = '7305a931-6cc4-11e7-a542-600292390f02' | compute rogueEdges('')`

Please note: It is easiest to see rogue edge functions results visually with the UI. The rogue edge will be colored red.

3.2 Explanation of queries

For the first three queries, one needs to understand that models are built with data called *independent variables* (IVs) that are used to predict what another piece of data, called the *target*, should be. So for instance, if you want to predict how much weight someone will lose, weight would be the target and things like calorie intake and exercise would be the independent variables.

Expected: Supervised Learning. It uses a model. Given rows of data, it will take the independent variables; it will tell you what it expects each target value to be.

Feature Selection: Supervised Learning. It uses a model. Given a model, it will tell you the most significant independent variables that are determining the target. It will weigh the independent variables.

What-if: Supervised Learning. It uses a model. Given one set of independent variables, will tell you what the target is expected to be. So for instance: if someone exercises an hour a day and eats 1000 calories a day, how much weight will they lose? With this query, you provide hypothetical independent variables.

Forecast: Supervised Learning. It uses a model. Uses a 'time-series model' to forecast several periods into the future. This forecast will take seasonality into consideration when forecasting into the future. What we mean by seasonality is this ... trends can be determined during certain periods of time. For instance, if we are forecasting sales, sales will always spike during Xmas seasons. Or certain business may experience higher than normal hits to their website on Saturday and Sunday. Forecasting models will learn the seasonality and take it into consideration when making forecasts.

Holt Winters: Unsupervised Learning. No model is used. This is a type of forecast that will not give results as well as the forecast function that uses a model. It will forecast into the future and also take seasonality into consideration. Although it may be advantageous to do a forecast without having to build a model, please note that Holt Winters forecasts will not be as accurate as forecasts that use a model.

Extrapolate: Unsupervised Learning. No model is used. Extrapolate is a "straight-line" forecast. Here is an easy-to-understand example ... a company is running low on disk storage. They want to know that if the problem continues ... at what point of time in the future will they be completely out of disk storage? So the extrapolate function will determine a straight line that takes the most current trend and extends it into the future. It will not consider the trends in the past.

Correlate : Unsupervised Learning. No model is used. Given many rows of data, will tell you the connected-ness (correlation) of all of the different fields of data. Strongly positive numbers are highly correlated in the same direction. Strongly negative numbers are highly correlated in opposing directions.

Feature Suggestion: Unsupervised Learning. No model is used. Given a target, will tell you the best fields to use as independent variables. In other words, which fields mostly affect the target.

Clusters: Unsupervised Learning. No model is used. Clusters data into bunches. Data within a bunch are strongly related to each other.

AnomalyDetection: Unsupervised Learning. No model is used. Given a numeric field, this function will flag rows in which that field is out of the typical range of values for that field. So for instance, if typical values of a field are between 1 and 10, and we get a value of 15,000, the 15,000 would be considered an anomaly. This query will accept an optional time field. The only purpose of this time field is to return it in the result set. This is useful when running on timeseries data if one wishes to see when the anomaly occurred. Choices for the anomaly algorithm are listed below:

- EllipticEnvelope
- LocalOutlierFactor (default)
- IsolationForest.
- null if you wish it to take the default

FCAnomaliesInRange: This query will find all anomalies within a specified time period. It differs from the above query in that it will use “timeseries models” for anomaly detection. For all models that pertain to the dataset the query is running on (the get statement should have the dataset in the where clause), it will run a forecast. Any forecast that is flagged as anomalous (determined because the actual value is outside of the upper and lower confidence limits determined by the model) will be returned. Because this query uses various models that run on various targets, it will return several types of anomalies that occurred within the specified timeframe.

ExpAnomaliesInRange: This query will also find all anomalies within a specified time period. It differs from the above query in that it will use regular models, not timeseries models for anomaly detection. What it will do is as follows: For all models that pertain to the dataset the query is running on (the get statement should have the dataset in the where clause), it will run an expected query. Any expected value that differs from the actual value will be returned. Because this query uses various models that run on various targets, it will return several types of anomalies that occurred within the specified timeframe.

RogueEdges: Using the results returned by the get statement of this query, this query will 1) find the relative topology of the highest parent of the activity associated with the longest running activity, 2) find the average elapsed time on each from/to edge of the relative topology, and 3) flag any edge that greatly (by a specified parameter) exceeds that average. The UI will color this edge red when drawing the topology map.

3.3 Real-time Predictions and Forecasts

When the real-time Machine Learning analytic grid is running, as data is streamed into a dataset, the real-time grid will check to see if the dataset it is being streamed into has a model associated with it. If it does, the real-time grid will then check to see if the data

passes the criteria specified in the mlmodel table for that model. If it does pass the criteria, then it will be sent off to Machine Learning for either a prediction or a forecast (depending on the model type).

After Machine Learning makes the prediction or the forecast, the real-time grid will then update the streamed data with the prediction or forecast along with the confidence of the prediction or forecast. It will create two properties: predicted-<target> and confidence-<target>, where <target> is the target entry in the mlmodel table.

The real-time grid will also determine whether the streamed data is anomalous. If it is, it will update a property called anomaly-<target>, where <target> is the target entry in the mlmodel table.

Anomalies will be determined as follows:

Non-timeseries string targets: if the streamed value does not match the predicted (expected) value, then it is marked as anomalous.

Non-timeseries numeric target: if the streamed value either exceeds or falls below the expected value by more than the amount specified in the anomaly margin field for the mlmodel entry, it is marked as anomalous.

Timeseries: if the streamed value falls either above the high or below the low forecasted values, it is marked as anomalous.

Chapter 4: Troubleshooting

Although we have spent a great deal of time and effort accounting for all different type of error scenarios, errors may still occur. Please see the following troubleshooting charts that provide how to resolve errors that may occur.

4.1 What can go wrong while training?

Table 3. Troubleshooting			
Error	Cause	To Diagnose	To Resolve
Training fails entirely.	Python can fail due to an unaccounted-for situation in the code or data. It can also fail because there is something about the data that makes it impossible to train.	View the jobstatus and exception in the mlmodel table. View the Python training log. View the log and job tables in Solr.	Reach out to meshIQ Technical Support. Resolution may involve one of the following: correcting some code, adjusting Machine Learning parameters, modifying the criteria that obtains training data, or updating IVs' default values. After you make corrections, you must set the jobstatus in the mlmodel table to 'COMPLETED' in order to reinitiate training.
Training fails entirely.	Data is not sufficient enough to train with. For instance, there may be too little data, or there may be too much missing data.	View the jobstatus and exception in the mlmodel table. View the Python training log. View the log and job tables in Solr.	Reach out to meshIQ Technical Support. Resolution may involve one of the following: modifying the criteria that obtains training data or adding default values to the independent variables. After you make corrections, you must set the jobstatus in the mlmodel table to 'COMPLETED' in

Table 3. Troubleshooting

Error	Cause	To Diagnose	To Resolve
			order to reinitiate training.
Training fails entirely.	Fields in the mlmodel table are improperly specified.	Check jKool logs. Check Python training log. Check status and exception in the mlmodel table.	Reach out to meshIQ Technical Support. Resolution will involve correcting the fields in the mlmodel table. After you make corrections, you must set the jobstatus in the mlmodel table to 'COMPLETED' in order to reinitiate training.
Training fails entirely.	Kafka goes down and the messages to retrain will not be received. The jobstatus in the mlmodel table will remain in a 'SCHEDULED' state.	Check the starttime, endtime, and jobstatus in the mlmodel table.	Restart Kafka. Update the jobstatus to 'COMPLETED' in order to reinitiate training.
Training fails entirely.	jKool Prediction Grid fails. This can be caused by a coding issue that doesn't account for something in the data.	Check jKool logs. Check status and exception in the mlmodel table.	Reach out to meshIQ Technical Support. Resolution will involve correcting Java code. After you make corrections, you must set the jobstatus in the mlmodel table to 'COMPLETED' in order to reinitiate training.
Certain types of models can fail.	Sometimes certain models have trouble handling training data, but other models can deal with the data without issue. For instance Random Forest	View the Python training log. View the log and job tables in Solr. This will state which model(s) failed.	Reach out to meshIQ Technical Support. Resolution may not be feasible, as some models simply perform better

Table 3. Troubleshooting			
Error	Cause	To Diagnose	To Resolve
	may fail but Tensor Flow operates on the data without issue.		than others on certain types of data. However, it may be correctable by adjusting Machine Learning parameters, modifying the criteria that obtains the training data, or specifying default values.

4.2 What can go wrong while running Machine Learning Queries?

Table 4. Machine Learning Query Troubleshooting			
Error	Cause	To Diagnose	To Resolve
Any query with an error message stating that the function arguments are improperly specified.	Queries need to be properly formatted.	An error message will appear in the UI.	Follow instructions in the error message on how to correct the query syntax. Also, see the documentation on how to write the query.
Any query with an error message that states: "failed in Python."	Error in Python code.	See the error message the query is giving. If it had no instructions on how to correct, there may be a coding error.	If no instructions are given on how to correct, reach out to meshIQ Technical Support. Python and jKool logs should be checked.
What-if – Error states: "...Please ensure that all IVs have been specified and that they are in the proper order."	All IVs specified in the mlmodel for the model being queried must be in the query and in the same order.	See the error message.	Reissue query with all IVs specified in proper order
Any query with a Java stack trace but no	Error in jKool code.	See the error message the	Reach out to meshIQ Technical Support.

Table 4. Machine Learning Query Troubleshooting

Error	Cause	To Diagnose	To Resolve
understandable explanation as to what went wrong.		query is giving. If it had no instructions on how to correct, it may be a coding error.	Python and jKool logs should be checked.
Expected or What-if Query - Error message states: "... failed in Python: <field> has value <value> that was not used to train the model."	This can happen if data has been passed in that the model had not been trained with. This will be specified in the error message returned by the query. For instance, if we trained with the colors blue and green, and the color orange was passed in via the query, then the Machine Learning will not know what to do with it.	See the error message the query is giving. It will be very specific as to what data it cannot deal with. This will never happen with regard to numeric data. It will only happen with string data.	Constrain the query to not work on the problematic record(s).
Expected - Error message states: "Please ensure all model fields are specified in the mlmodel table" or "Null Independent Variables."	The mlmodel table was not populated properly for that model.	See the error message.	Correct the entry in the mlmodel table.
Query is hanging.	This can happen if either Kafka or Python process(es) are down.	Query will be hanging and eventually time out.	Ensure Kafka is up and running. Ensure the Python process(es) are up and running. Check log files.
Expected, Forecasting Queries, What-if - Error message states: "Model does not exist."	User had entered a query against a model that does not exist in the mlmodel table.	See the error message.	See all available models for that repository and correct the query.
Expected - Error stating "No rows meet the criteria for this prediction."	No data can be found to predict.	See the error message.	Look at the criteria for the entry in the mlmodel table. Also look at any additional criteria that was

Table 4. Machine Learning Query Troubleshooting

Error	Cause	To Diagnose	To Resolve
			entered. Determine why the criteria is constraining the data so much that no records are returned.
Correlate – Error stating “No rows meet the criteria for this correlation.”	No data can be found to correlate.	See the error message.	Correct constraints. Check the raw data to see if it exists. Ensure the time period is correct (i.e., it may be set to something like ‘the last month’).
Forecast – Error stating “Error forecasting. No data to forecast.”	No data can be found to forecast.	See the error message.	Look at the criteria for the entry in the mlmodel table. Also look at any additional criteria that was entered. Determine why the criteria is constraining the data so much that no records are returned.
Forecast – Error message states: “Please choose a record that began after the model update time.”	This happens when forecasting on a single record. You can only forecast after the latest date used when training a time-series model. For instance, if the model was trained with data from 1/1/ to 6/30, then you can only forecast beginning 7/1. The record specified has to have a time that is after the latest date used when training the model. In this example, it needs to be after 7/1.		Correct the query. Forecast on a record with a date after the latest date used when training the model (mlmodel.maxdatadate).
Forecast – Error message states: “... not enough forecasts have been returned.”	The number of forecasts requested exceeds one of the following: <ul style="list-style-type: none"> • the data available, if regressors are 	See the error message.	Correct the query or increase prediction.periods and restart the kickoff_run_queries.py

Table 4. Machine Learning Query Troubleshooting

Error	Cause	To Diagnose	To Resolve
	indicated, or <ul style="list-style-type: none"> the maximum number of forecasts specified in the prediction.periods field in the ml.properties file. 		for it to take effect.
Forecast – “Error forecasting. An active timeseries model does not exist for this model name”	A forecast query has been issued against a model that does not exist in that repo or against a non-time-series model.	See the error message.	Issue query against a time-series model.
Forecast – “Please ensure that all time-series fields are specified in the mlmodel table”	A forecast query has been issued that is either against a model that is not a time-series model or that is a time-series model but has some entries missing in the mlmodel table.	See the error message.	Correct the entry in the mlmodel table, retrain the model, rerun the query.
ClusterDetails – Error states: “Cluster details do not exist for this id. It’s possible that the cluster cache expired. Please re-run clustering and try again.”	When a cluster is run, the details about it are only cached for about 5 minutes. So the cache may expire.	See the error message.	Rerun the getclusters query and then rerun clusterdetails.
Clusters or Cluster3D – Error states: “No rows meet the criteria for this clustering.”	The cluster is constrained too much or and no rows are being returned for it to work on or no rows exist for it to run on.	See the error message	Correct the constraint. Check the raw data to see if it exists. Ensure the time period is correct (i.e., it may be set to something like ‘the last month’).
FeatureSelection – Error states: “This model does not exist.”	The query is being run against a model that either does not exist or is not active.	See the error message.	Correct the query.

Table 4. Machine Learning Query Troubleshooting

Error	Cause	To Diagnose	To Resolve
FeatureSelection – Error states: “Feature Selection is unavailable for timeseries models.”	Feature selection can only be run against non-time-series models.	See the error message.	Correct the query.
FeatureSelection – Error states: “Feature Selection cannot be run on these models.”	Certain models do not lend themselves to feature selection. So if the optimal model that Python chose for the data is one of these, feature selection cannot be run on it.	See the error message.	The query cannot be run for this model.

4.3 What can go wrong with real-time predictions?

Table 5. Real-time Predication Troubleshooting			
Error	Cause	To Diagnose	To Resolve
Predictions and anomalies stop getting updated.	Possible causes: Kafka is down, Python process(es) down, Prediction Grid down. Something about the data changed that is causing it not to work properly with the model any longer.	View all log files and the log table. Look for exceptions.	Reach out to meshIQ Technical Support. Programs need to be put in place that will update records that were not marked.
Predictions and/or anomalies are incorrect.	Programming issue	Analyzing updates.	Reach out to meshIQ Technical Support.