NASTEL
Driving Business Transaction Performance ®

Machine Learning
March 2021

Contents

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". At Nastel, we are using Machine Learning to provide 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 Nastel XRay train a model.
- Un-supervised Learning – No model is required.

Components & Properties Required for Machine Learning

The Nastel XRay components that are required to run Machine Learning include:

- Gateway
- DB Writer
- Query Grid & Analytic Grid (part of the Query Grid)
- Compute Grid
- Service
- Prediction Grid

There are a few Nastel AutoPilot properties that will need to be specified in Prediction and Query grids. They are as follows:

- `ML Directory` – (in both the prediction and query grids) as of the writing of this document, this should be set to the system property `{jkool.ml.prediction.data.dir}`
- `Prediction Periods` – (only in the prediction grid) This is the maximum number of predictions that python will forecast when using a timeseries model. As of the writing of this document, this property should be set to the system property `{ml.prediction.periods}`

The system properties just mentioned above should be set in the Tomcat catalina.sh file as follows:

```
-Djkool.ml.prediction.data.dir=<main.dir in ml-python.properties described below>
-Djkool.ml.prediction.periods=<maximum number of predictions>
```

Machine Learning Architecture

Diagrams of the Machine Learning architecture are at the end of this document. It is not necessary for these diagrams to be fully understood in order to install Machine Learning. However, there are a few things that are important to note:

- The Machine Learning code is run in separate Python processes.
- These processes communicate with Nastel XRay via Kafka.
- Data is exchanged between the Machine Learning processes via Kakfa and also via a shared file system. The directory that will be shared between Python and Nastel XRay must be specified in the properties file that will be described later in this document

Install Required Python Environment

Below are instructions on how to get this Python environment installed on a Unix/Linux server and also how to install the required dependencies.

To install Python, follow the instructions in this link:

https://docs.anaconda.com/anaconda/install/linux/

step1:

```
apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1
libxcursor1 libxcomposite1 libasound2 libxi6 libxtst6
```

step:2

download: https://www.anaconda.com/products/individual#linux

or for later versions download from:

https://repo.anaconda.com/archive/

Recommended version: `anaconda3-2020.02-Linux-x86_64.sh`

step3:

Install downloaded file in step2:

```
bash /opt/anaconda3/anaconda3-2020.02-Linux-x86_64.sh
```

Check the name of the file it may change.

Install Python Dependencies and Nastel's Machine Learning Code

Python is an interpreted language. Therefore, no compilation is necessary. All that needs to be done in order to deploy Nastel's Machine Learning Python code to a server, is to copy over files to the server. These files are located on Github in the following location: `https://github.com/Nastel/jKoolMLPython`. Prior to doing this however, you will need to create an anaconda environment and install Python dependencies. To do all of this, follow these steps:

Create a directory structure

- Create a folder on the server you are installing to.
- Within this folder create the following subdirectories: var, lib, config

Create the anaconda environment

- Copy the github file "jkool.yaml" into the lib subdirectory.
- From the anaconda/bin directory, issue the following command which will create an 'envs' subdirectory under lib. This subdirectory contains the environment:

```
conda env create --prefix envs -f <yaml file>
i.e. conda env create –-prefix <directory you created>/lib/envs -f <directory you
created>/lib/jkool.yaml
```

Copy over the properties file and update it

Nastel's Machine Learning Python code is expecting a properties file to be located in the config directory you created. Copy to the config directory the file located on Github in the config directory:

- `ml-python.properties`

It is imperative that you update the values in this properties file to reflect the server you are deploying to. Below is an explanation of the properties in that file. With the exception of the ".dir" entries, most values can remain the way that they are in Github no matter where the code is being deployed to. Ensure slashes in file names are in the proper orientation. Slashes differ between unix and windows.

`main.dir`	Main directory data will be written to. <directory you created>/var/data
`models.dir`	Directory models will be serialized to. <directory you created>/var/data/models
`pp.dir`	Directory pre-processors will be serialized to. <directory you created>/var/data/preprocessors
`exports.dir*`	Shared directory between Nastel XRay and Python <directory you created>/var/data/exports
`config.dir`	Config directory <directory you created>/config
`log.dir`	Log file directory <directory you created>/logs
`kafka.port`	Port that Kafka is running on
`training.pipe`	Training pipe (see architecture diagram)
`querying.pipe`	Querying pipe (see architecture diagram)
`prediction.pipe`	Prediction pipe (see architecture diagram)
`training.result.pipe`	Training result pipe (see architecture diagram)
`querying.result.pipe`	Querying result pipe (see architecture diagram)
`prediction.result.pipe`	Prediction result pipe (see architecture diagram)
`prediction.periods`	Maximum # of predictions to use in the forecast function

***If Python and Nastel XRay do not reside on the same server, you must make this directory shareable.**

Please note that directories specified in this config file that are other than the ones you created manually will be automatically created for you when the machine learning code is kicked off.

The final directory structure for the machine learning code will be as follows:

```
config
lib
   envs
var
   data
       exports
       models
       preprocessors
   logs
```

Copy over config files:

Nastel's Machine Learning Python code is expecting two other files to reside in the config directory. Copy the following files located on Github in the config directory:

- `best-params.json`
- `log.conf`

Update the directories specified in the log.conf file to match your directory structure.

Exports directory must be shareable:

The directory specified in the property exports.dir must be made shareable if Nastel XRay and the Machine Learning code reside on different servers.

Kicking off Machine Learning Processes

When you created the conda environment, an environment directory will have been created under the lib directory. All files on Github ending in ".py" should be copied over to this directory.

You will now need to activate this environment by issuing the following command.

```
conda activate <path to the environment directory>
```

There are three different python processes that must be kicked off for the Machine Learning to run. They are:

1. Training – this is a process that will train the models
2. Querying – this is a process that will perform machine learning queries
3. Real-time Predictions – this is a process that will perform real-time predictions and forecasts.

To kick-off these processes, issue the following commands:

```
1. nohup python.exe  kickoff_train.py &
2. nohup python.exe  kickoff_run_queries.py &
3. nohup python.exe  kickoff_real_time_predictions.py &
```

To verify the processes got kicked off properly, tail the nohup.out file by issuing the following command in a separate terminal window:

```
tail –f nohup.out
```

To check if the processes are running, issue the following command:

```
ps –ef | grep kickoff
```

If there is heavy load, then multiple processes will need to be kicked off. Simply issue the command associated with the slow process again in order to create additional processes. Do this until the desired speed is achieved.

Please note that for training to work, repositories must have an entry in the mlmodel table. This is discussed below.

Log Files

The nohup.out file will contain useful information when diagnosing why a process may not have started. In addition, Nastel's Machine Learning Python code will create and perpetually update two log files. These log files are named: training.log and af.log. They will be located in the log directory (see value of `log.dir`) specified in the properties file.

If the Machine Learning code that is running in Java should encounter an error, the error will be written to the regular Nastel XRay log files.

Data Type and Data Format Restrictions

Please be advised that Machine Learning will only run on "Datasets". The most difficult part of a machine learning engagement will be to ensure that data gets into the proper format for Machine

Learning to run on it. If data is maintained in activities, events, and/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 the Machine Learning. Getting the data into a proper format for Machine Learning can be accomplished by the following mechanisms:

1. The user changes how data is streamed into Nastel XRay so that it is originally streamed into the datasets table in the proper format.
2. The user utilizes Nastel XRay Views, Macro's, or Stored procedures in order to get the data into the proper format and to copy it over to the dataset table.
3. The user either engages Nastel's technical team, or on their own, creates code that massages the data into the proper format and stores it in the datasets table.

Training a model

If a user desires to have Supervised Learning (described above), an entry must exist in the Nastel XRay "mlmodel" table. This entry must be manually inserted into Solr via an upsert statement and a Nastel Support team member should ensure this entry is proper. Below is the list of fields in the mlmodel table. Users need only concern themselves with fields that are marked as Computed: No.

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	Function, 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.	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
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
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
WaitTime	TimeInterval	How long to wait to rebuild the model	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

Once an entry is in this table and the "prediction grid" is started, models will build automatically. The models will also automatically re-build themselves at the "WaitTime" interval specified in the MLModel table. Please note that re-building 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 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. Kick-off model training via the following Train Statement:

```
TRAIN MODEL<model name>
```

ML Analytic Queries

Below is a list of all the Machine Learning queries that can be run along with their syntax

Color key: **optional** **required**

expected

(expct)

compute expected(<model name>, <return all fields in the result set>,<criteria>) where …

- compute expected('SPECIES')
- compute expected('SPECIES', true,"PETAL_LENGTH > 2")

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 activity/event/snapshot/dataset compute correlate(<field1>, <field2>, ...) where ...

get dataset compute correlate() where …

get events compute correlate(PETAL_LENGTH,PETAL_WIDTH,SEPAL_LENGTH,SEPAL_WIDTH)

get dataset compute correlate() – *ML will deduce fields*

featureSuggestion –no models involved

(fsuggestion,fsug)

get activity/event/snapshot/dataset compute featuresuggestion(<field1>, <field2> ....<target>) where <criteria>

get dataset compute featuresuggestion(<target>) where <criteria>

get event compute featuresuggestion(PETAL_LENGTH, PETAL_WIDTH, SEPAL_LENGTH, SEPAL_WIDTH,SPECIES)

get dataset compute featuresuggestion(SPECIES) – *ML will deduce fields*

clusters – no models involved

(cl)

get event/activity/snapshot/dataset compute clusters(<numberOfClusterColumns>, <field1>, <field2> ..., <additionalreportingcolumn1>, <additionalreportingcolumn2>, ...,<number of clusters or auto (in quotes)>,<recluster boolean>) where* <criteria>

get events compute clusters(3, PETAL_LENGTH, PETAL_WIDTH, SEPAL_LENGTH, SEPAL_WIDTH,'auto',true)

get events compute clusters(3, PETAL_LENGTH, PETAL_WIDTH, SEPAL_LENGTH, SEPAL_WIDTH,'3',true)

clusterDetails

(cld)

get event/activity/snapshot/dataset compute clustersdetails(<topic id>)

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

clusters3d – no models involved, used to build a 3D plotly chart

(cl3d)

get event/activity/snapshot/dataset compute clusters3d(<field1>, <field2>,<field3>, <additionalreportingcolumn1>, <additionalreportingcolumn2>, ... ,< number of clusters or auto (in quotes) >) ,<recluster boolean>) where* <criteria>

get events compute clusters3d(PETAL_LENGTH, PETAL_WIDTH, SEPAL_LENGTH, SEPAL_WIDTH,'auto',true)

get events compute clusters3d(PETAL_LENGTH, PETAL_WIDTH, SEPAL_LENGTH, SEPAL_WIDTH,'3',true)

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

holtwinterspredication – no models involved, indicate a time unit of '1' always

(holtwinters, hw)

get dataset compute holtwinterspredication(<aggregated field>, <periods in forecast>) where <criteria> group by <field> bucketed by 1 <time unit>

get dataset compute holtwinterspredication(avg(syn_raidPort_totalIOPS), 24) where name = 'DiskManagementAggregated' group by properties('starttime') bucketed by 1 hour show as linechart

extrapolate – no models involved, indicate a time unit of '1' always

(extrap)

get dataset compute extrapolate(<aggregated field>, <number of forecasts>) where <criteria> group by <field> bucketed by 1 <time unit>

get dataset compute extrapolate(avg(syn_raidPort_totalIOPS), 10) where name = 'DiskManagementAggregated' group by properties('starttime') bucketed by hour show as linechart

anomaly – no models involved

get dataset compute anomaly(<field>) where <criteria>

get dataset compute anomaly(CourseCost)

Explanation of queries

For the first three queries, one needs to understand that models are built with data, called *independent variables*, that are used to predict what another piece of data, called the *target*, should be based on the independent variables. For instance, if you want to predict how much weight someone will lose, weight would be the target and things like calorie intake, 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, trends can be determined during certain periods of time. For instance, if we are forecasting sales, sales will always spike during holiday 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: Un-supervised learning, no model. 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. However, the forecasts will most likely not be as accurate as forecasts that use a model.

Extrapolate: Un-supervised learning, no model. 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? 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 : Un-supervised learning, no model. 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: Un-supervised learning, no model. Given a target, will tell you the best fields to use as independent variables. In other words, which fields mostly affect the target.

Clusters: Un-supervised learning, no model. Clusters data into bunches. Data within a bunch are strongly related to each other.

AnomalyDetection: Given a numeric field, this function will highlight rows in which that field is out of the typical range of values for that field. 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.

Real-time Predictions and Forecasts

When the real-time 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 Python for either a prediction or a forecast (depending on the model type).

After Python 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 if 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.

Troubleshooting

Please see the following trouble shooting charts that provide how to resolve errors that may occur.

What can go wrong while training?

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.	Contact the technical team. Resolution may involve: correcting some code, adjusting machine learning parameters, modifying the criteria that obtains training data, updating ivs default values. After correcting, the jobstatus in the mlmodel table will need to be set to 'COMPLETED' in order to re-initiate training.
Training fails entirely.	Data is not sufficient enough to train with. For instance, there may be too little data, 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.	Contact the technical team. Resolution may involve: modifying the criteria that obtains training data. Adding default values to the independent variables. After correcting the jobstatus in the mlmodel table will need to be set to 'COMPLETED' in order to re-initiate training.
Training fails entirely.	Fields in the mlmodel table are improperly specified.	Check Nastel XRay logs. Check python training log. Check status and exception in the mlmodel table.	Contact the technical team. Resolution will involve correcting the fields in the mlmodel table. After correcting the jobstatus in the mlmodel table will need to be set to 'COMPLETED' in order to re-initiate training.
Training fails entirely.	Kafka goes down and the messages to re-train 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 re-initiate training.
Training fails entirely.	Nastel XRay Prediction Grid fails. This can be caused by a coding issue that doesn't account for something in the data.	Check Nastel XRay logs. Check status and exception in the mlmodel table.	Contact the technical team. Resolution will involve correcting java code. After correcting the jobstatus in the mlmodel table will need to be set to 'COMPLETED' in order to re-initiate 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 may fail but Tensor Flow operates on the data without issue.	View the python training log. View the log and job tables in solr. This will state which model(s) failed.	Contact the technical team. Resolution may not be an issue as some models simply perform better than others on certain types of data. However, it may be correctable by adjusting machine learning parameters or modifying the criteria that obtains the training data or specifying default values.

What can go wrong while running Machine Learning Queries?

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 error message the query is giving. If it had no instructions on how to correct, there may be a coding error.	If no instructions given on how to correct, contact the technical team. Python and Nastel XRay 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 error message.	Re-issue query with all ivs specified in proper order
Any query with a java stack trace but no understandable explanation as to what went wrong.	Error in Nastel XRay code.	See error message the query is giving. If it had no instructions on how to correct, it may be a coding error.	Contact the technical team. Python and Nastel XRay 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 them 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 error message the query is giving. It will be very specific as to what data it cannot deal with. This will never happen with regards 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 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 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		Correct the query. Forecast on a record with a date after the latest date used when training the model (mlmodel.maxdatadate)

	used when training the model. In this example, it needs to be after 7/1.		
Forecast – Error message states "… not enough forecasts have been returned".	The number of forecasts requested exceed either: the data available if regressors are indicated or the maximum number of forecasts specified in the prediction.periods field in the ml.properties file.	See the error message	Correct the query or increase prediction.periods and restart the kickoff_run_queries.py 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 it's been issued 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 it 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, re-train the model, re-run 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	Re-run the getclusters query and then re-run 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.
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.

What can go wrong with real-time predictions?

Error	Cause	To Diagnose	To Resolve
Predictions and anomalies stop getting updated.	Possible cause are: Kafka is down, Python process(es) down, Prediction Grid down. Something about the data changed that is causing it to no longer work properly with the model.	View all log files and the log table. Look for exceptions.	Contact the technical team. 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.	Contact the technical team.

Updating Python code and dependencies

After the initial install, when new versions of Python are released, a zip file will be delivered. Simply unzip this file into the ML directory, for example: /opt/nastel/ml directory (this zip file is a download from GitHub).

If python dependencies require updating, you will be provided with a new .yml file. To update the environment issue the following commands:

```
conda deactivate
conda env update –name <path to the environment directory> --file <yaml file>
conda activate <path to the environment directory>
```