Summary

[

mitosis

1600×636 375 KB

](https://forum.numer.ai/uploads/default/original/1X/dcf9ef8f9e6e5cb24da32fc2ff041bbacfc70df1.jpeg)

We performed a model split to link models to a specific tournament instead of being shared between tournaments. We did this change to simplify the backend data model, which makes it easier to implement and support various functionalities like payments, compute, etc. In the future, this change will let us simplify the API instead of duplicating APIs between tournaments.

Existing models with a Signals submission were copied and converted to Signals models.

Going forward, any models added through signals.numer.ai will be Signals models. Users who sign up through signals.numer.ai will get one Signals model created automatically. Likewise, users who create their account through numer.ai will only get a Tournament model. Signals models can also be made through the API on the createModel

endpoint when tournament: 11

is passed in and Tournament models are created when tournament: 8

is passed in.

We introduced an additional change with model names in this update. Model names are now “owned’ by the account owner. You can re-use model names between tournaments if you already have a model with that name under your account. This also prevents other users from using your model names. For example, I can’t create a model named “integration_test” because it doesn’t belong to my account.

These updates may break existing code depending on how you are currently using the API. Read on for more details.

API Changes

Get models

Getting models that belong to your account now returns models for all tournaments. This change caused issues with the payouts app, specifically for users who had models with the same name for both tournaments. This endpoint now returns an additional tournament field.

query getModels { account { models { name id tournament } } }

Returns:

{ "data": { "account": { "models": [{ "id": "102052af-a3f4-44ea-b4e4-8d419d3ee4e2", "name": "chanes", "tournament": 8 }, { "id": "3c3b0f98-53a8-4670-aa02-982d01b310a8", "name": "chanes", "tournament": 11 }] } } }

Users with one model might now have two models

Users who previously shared one model for both Tournament and Signals will now have two models. This change could break existing code that expected only one model. For example, if you previously uploaded predictions without passing a model_id

:

napi.upload_predictions(file_path=file_path, tournament=tournament_number)

And you now have two models (one for Tournament and one for Signals) you will get this error:

ValueError: Account contains multiple user models - you must specify model_id.

To resolve this issue, pass in the model_id when uploading predictions:

napi.upload_predictions(file_path=file_path, tournament=tournament_number, model_id=model_id)

Users who participated in one tournament do not need to make any changes.

Handling new model_ids

Signals models created in this migration will have a different model_id

than before. However, safeguards were put in place so if an old model_id is used with tournament = 11

or when using a Signals endpoint, the API call will use the new Signals model. This change affects two endpoints: v2ChangeStake

and createSignalsSubmission

.

For example, if I previously shared one model for both Tournament and Signals:

{ "data": { "account": { "models": [{ "id": "11111111-a3f4-44ea-b4e4-8d419d3ee4e2", "name": "chanes" }] } } }

If I wanted to do make a Signals submission I would have done:

createSignalsSubmission(filename: "chanes_signals_upload.csv" modelId: "11111111-a3f4-44ea-b4e4-8d419d3ee4e2") { id }

Now after the model split I have two models:

{ "data": { "account": { "models": [{ "id": "11111111-a3f4-44ea-b4e4-8d419d3ee4e2", "name": "chanes", "tournament": 8 }, { "id": "bbbbbbbb-036c-4216-b2f7-f8ed4beb88e9", "name": "chanes", "tournament": 11
}] } } }

Either of these API calls will work:

createSignalsSubmission(filename: "chanes_signals_upload_prev_model_id.csv" modelId: "11111111-a3f4-44ea-b4e4-8d419d3ee4e2") { id }

createSignalsSubmission(filename: "chanes_signals_upload_new_model_id.csv" modelId: "bbbbbbbb-036c-4216-b2f7-f8ed4beb88e9") { id }

So if the old model_id

is used (11111111-a3f4-44ea-b4e4-8d419d3ee4e2

), it will automatically map the submission to the Signals model with the same model name

(bbbbbbbb-036c-4216-b2f7-f8ed4beb88e9

)

Similarly, for the changeStakeRequest endpoint and using the previous model examples, either of these requests will work:

v2ChangeStake(value: "9000.00", type: "increase", modelId: "11111111-a3f4-44ea-b4e4-8d419d3ee4e2", tournamentNumber: 11) { dueDate requestedAmount status type }

v2ChangeStake(value: "9000.00", type: "increase", modelId: "bbbbbbbb-036c-4216-b2f7-f8ed4beb88e9", tournamentNumber: 11) { dueDate requestedAmount status type }

Both will apply a stake change request of increase 9000.00 to the Signals model bbbbbbbb-036c-4216-b2f7-f8ed4beb88e9

With these safeguards in place, existing code does not need to be changed. However, we recommend users update their code to use the new model_id

.