Goal

Modernize the way we deliver models and technical premium to our business partners (e.g., Underwriters, Claims Reps, etc.) Our aim is to deploy these models as an API, such that they can be easily consumed from various downstream applications. In addition to this, we're interested to see a solution with the following components:

- Deployment Governance (e.g., approval process to move to production, version history, etc.)
- Complete, searchable, and auditable provenance of all models
- Model explainability (e.g., which features were important to the prediction and why)
- Model quality monitoring (e.g., model accuracy over time)
- Data quality monitoring (e.g., data drift)
- Holistic view of all models in production (e.g., how many models do we have, what's the aggregate model quality, how many need to be retrained, etc.)
- API telemetry and transaction auditability (e.g., how often is the API used, how performant is it, what were the inputs and outputs in each transaction, etc.)
- Native support for development in open-source languages (e.g., R, Python, etc.) and deployment of varying model types to the solution (e.g., GLM, GBM, CNN, etc.)
- Ability to make pricing transactional data available for analytical purposes

Operating Model – Roles and Personas

What does the operating model look like if we were to use your solution?

For example:

- Data Engineer – builds systems that collect, manage, and convert raw data into usable information.
- Data Scientist – builds models, performs advanced analytics, etc.
- ML Engineer – takes work from data scientists and deploys it to production, and/or may have deep technical ability and machine learning expertise. Focused on model and data quality monitoring, alerting, etc. May have software engineering background.
- MLOps Engineer – manages the CI/CD pipeline, works closely with the ML Engineer. Likely to have DevOps background.

Please describe how each step in the use case aligns with your best practice operating model.

Sample Use Case

To deliver technical premium for Commercial Retail – Auto, you need a combination of pricing inputs from:

1. Manual Premium
    a. Comes from our Policy Admin System, Majesco
    b. Would be API data from Majesco
        i. For the demo, briefly describe how you would ingest this type of data from an internal vendor
2. Experience Mod (tool_name = Experience Rater, file_type = Excel, model_type = Rater)
    a. Relies on historical losses, which comes in the form of PDFs, Excel files, etc.
    b. Sample algorithm:
        i. If Manual Premium < $10,000
            1. If losses > $20,000 then 2
            2. Else .8
        ii. Else
            1. If losses > $1,000,000 then 1.5
            2. Else 1
3. Loss Rating Tool (tool_name = CLRT, file_type = Excel, model_type = Rater)
    a. Relies on (1) and (2)
    b. Only used for large accounts
        i. $500,000+ experience rated manual premium
        ii. This is an example of a business rule we'd like to see handled (i.e., when to use a specific algorithm)
    c. In production, the input to this algorithm is dynamic (e.g., what if analysis), and will come from our Underwriting Workbench (i.e., the UI where the Underwriter works to quote policies).
    d. Example input variables
        i. Historical exposure by year
            1. Auto exposure is vehicle count
        ii. Losses by year
        iii. Claim detail
        iv. Aggregate losses by year
        v. Account information
    e. Please describe how you would handle the dynamic input and output of (3.d) as part of your deployment
    f. Output is technical premium
        1. CLRT Mod * Experience Mod * Manual Premium = Technical Premium
        2. CLRT Mod sample algorithm:
            a. If exposure < 100
                i. If losses > 1000000 then 1.8
                ii. If losses > 500000 then 1.4
                iii. If losses > 250000 then 1.0

iv. If losses > 100000 then 0.9
v. Else 0.8

b. Else
i. If losses > 1000000 then 2.0
ii. If losses > 500000 then 1.5
iii. If losses > 250000 then 1.0
iv. If losses > 100000 then 0.85
v. Else 0.75

4. Discretionary Pricing Guidance (tool_name = NBPT, file_type = Excel, model_type = GLM)
   a. Relies on (1) and (2)
   b. Sample data and schema provided
   c. Please build a GLM with the following model specification
      i. link = "log"
      ii. family = "Tweedie"
      iii. weight = manual_premium
      iv. target = manual_loss_ratio (incurred_loss_and_alae / manual_premium)
      v. features = [
         1. policy_year
         2. risk_state
         3. n_power_units
         4. prior_claim_freq_3yr
         5. easi_snowfall
         6. dnb_credit_score

      ]
   d. Output is technical premium
      i. GLM prediction * Experience Mod * Manual Premium * (1 + Indication) = Technical Premium
      ii. Indications are provided in a sample file
         1. [sample_data].[risk_state] = [indication].[risk_state]
         2. [sample_data].[coverage_type] = [indication].[coverage_type]
         3. Join on these keys to obtain the Indication
5. If the CLRT is used, (3) and (4) are credibility weighted for final technical premium
   a. z * Technical Premium from (3) + (1-z) * Technical Premium from (4)
   b. Assume z = .25
6. Deploy the Technical Premium endpoint and show us a sample request/response using provided data
   a. What's required to deploy the model (e.g., does it need to be conformed to a certain standard, need a specific `predict.*` file, or similar)?
7. Show us the monitoring and governance items we'd like to see (from the Goals section, which is in order of preference), if available. If we run out of time during the demo, please plan to share a write-up and diagram of how this would work within your platform. We understand our use case may not be set up in a way that easily lends itself to this, so please feel free to use a sample demonstration that does.

Further Context and Questions

- The Experience Rater and CLRT represent some rating algorithm in Excel that we want to govern, monitor, and migrate over in the easiest way possible – how would you do this?
    - Would you connect directly to the existing Excel tools, reprogram them into another language, or something else?

- How are business rules incorporated (e.g., (3.d) from the sample use case)?

- For technical premium, the final algorithm needs to combine all the necessary components and output as an API to be used by downstream applications.

    - We want nice to have monitoring and governance tools for models and data in any input or math not coming from Majesco, which is the case for (2), (3), and (4) in the use case.

    - That said, if we were able to use those same monitoring and governance tools on the inputs from Majesco that would be another nice to have.

- The other pricing tool, Majesco, is just a generic input from an internal vendor and we'd just want to see how the platform would pick this up via an API or similar (e.g., (1) from the sample use case. We DON'T need to replicate ISO rating that is already happening via Majesco).

- As a nice to have it would be good to include a call out to an external vendor, via API or similar, to input into the GLM (e.g., Dun & Bradstreet - this is nice to have because ideally the Underwriting Workbench (i.e., the tool the Underwriter works with) would be doing this data collection, but we are wondering if it's supported within your platform).

- We are interested to learn about a "day in the life" of the different roles and personas as they would work within your platform. Our primary interest is in time-to-value (e.g., how long does it take to go from model or rater build to deployment) and iteration-to-value (e.g., how do we do more with the time-to-value pipeline, how easy is it to retrain our model when we know it needs an update, etc.). Please describe how your solution aligns with this interest.

- Does your solution natively support development, staging, and production environments? If not, how do you manage this?

    - If it is natively supported, what is the review and approval process in your solution?

API Requirements and Questions

- Needs to provide a REST API
- Goal is one API per model
    - In the sample use case, please opine on the optimal way to deploy these components (e.g., each tool could be its own API, with a technical premium API that knows how to apply the business rules and make the final premium calculation)
- Needs to be secure by authentication
    - How do you do this?
- Data transfer needs to be encrypted (TLS)
- How is the API schema documented?
- What, if any, data transformations are required to conform to the API?
- Do we need any client specific customization to conform to your API?
- What's the protocol for inbound and outbound API calls?
- What type of infrastructure do you deploy on (e.g., Kubernetes)?

Appendix

High-level Architecture Diagram of Project Scope and Focus