outrageously
AMBITIOUS

Module 5:
Model Lifecycle Management

Duke
PRATT SCHOOL *of* ENGINEERING

Module 5 Objectives:

At the conclusion of this module, you should be able to:

1) Assess & mitigate the main risks of ML models in production
2) Identify key elements of an ML system to monitor and determine model retraining strategies
3) Describe the benefits and best practices of model versioning

outrageously **AMBITIOUS**

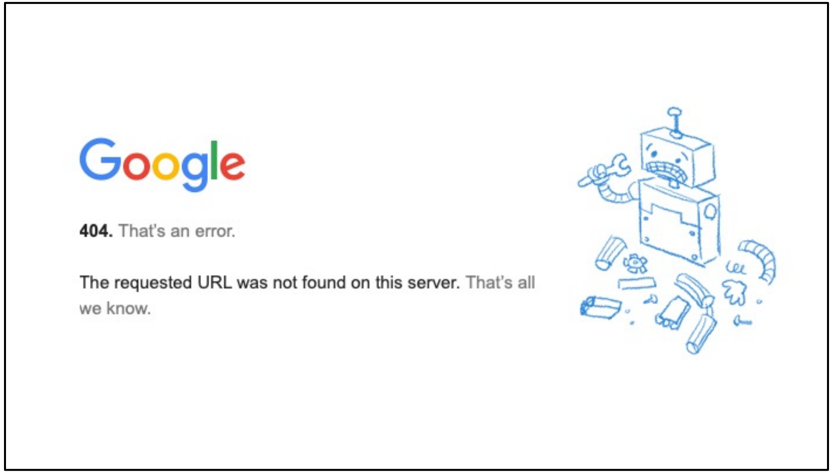
ML System Failures

ML system failures

- ML-based products are subject to the normal failure causes of software products, plus additional risk factors
- Google study found that majority of its ML failures were not due to the model itself
- However, model failures are particularly dangerous because they can be more difficult to detect

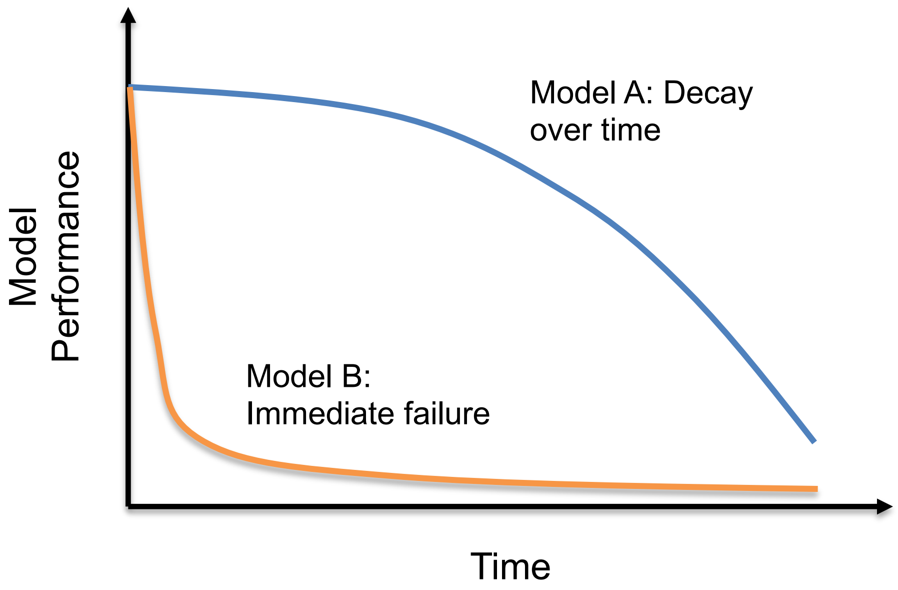
Software vs. model failures

Software failure



Model failure

Model decay


Model A: Decay over time
Model B: Immediate failure
Model Performance
Time

Model issues in production

- Training – serving skew
- Excessive latency
- Data drift
- Concept drift

Training – serving skew

- Mismatch between the training data and the input data while in production
- Often results from training on artificially constructed or cleaned dataset
- Examples:
  - Computer vision model trained on high-resolution imagery with perfect lighting
  - NLP model trained on limited subset of questions fails to anticipate variety of questions asked
- Typically manifests itself immediately after moving into production

Excessive latency

- Latency in generating predictions can vary based on
  - Volume of input data
  - Data pipeline
  - Choice of model
- For online & edge models, low latency is critical:
  - Unlocking a phone
  - Autonomous driving systems

Data drift

- Model is trained on a static training set, but the environment changes over time
  - Population shifts
  - Adversarial reactions e.g. identifying spam
- Results in shifting distribution of feature(s)
- Changes can occur quickly or slowly over time
- Model performance degrades on the edges of feature space

Data drift example

Shifting demand for rental bicycles

Concept drift

- Distributions of data may stay same, but the patterns that the model learned no longer apply
- Results from shifts in the relationships between inputs and outputs
  - Consumer preferences
  - Human behavior shifts

Concept drift example

Detecting fraud: one-way flight tickets

outrageously
AMBITIOUS

ML System Monitoring

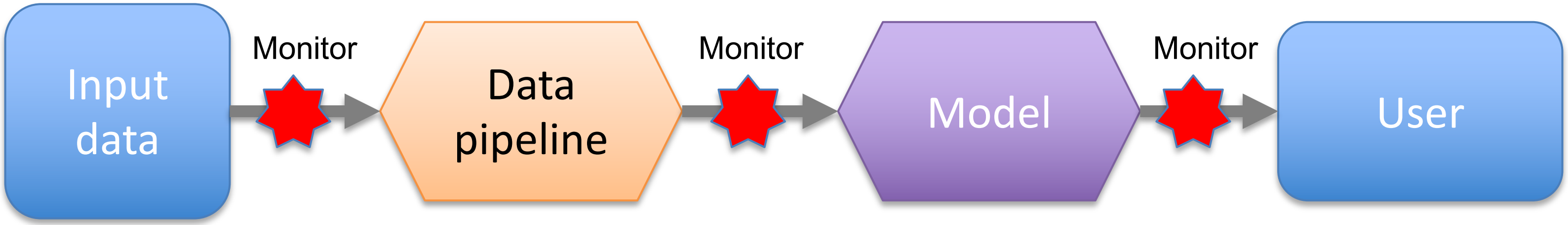
Duke
PRATT SCHOOL *of* ENGINEERING

ML system monitoring

- ML models often fail silently – issues are not obvious
- Proper monitoring can diagnose latent issues before they cause noticeable disruption to users
- ML monitoring should accompany standard software monitoring best practices

What to monitor

- Input data – data drift
- Data pipeline
- Model outputs
- Target labels (if available) – concept drift



Input data monitoring

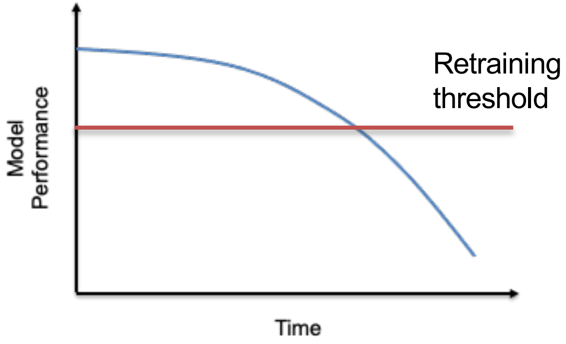
- Basic quality checks
  - Correct schema, encoding
  - Expected volume of data
  - Missing data
- Distribution of input data
  - Visualizations or statistical tests to flag potential data drift
- Correlation of features to targets
  - Possible concept drift issues
- Periodic manual audits

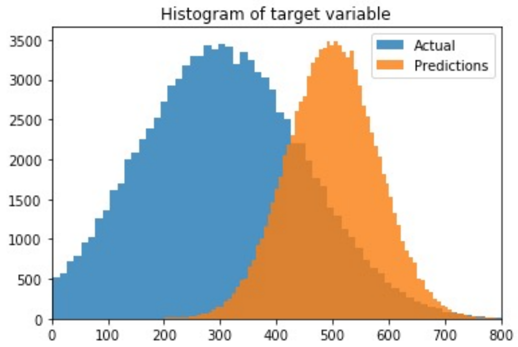
Data pipeline monitoring

- Disparities can arise between processing of training data and processing of live input data
  - During training, processing applied in batch
  - Production processing of streaming data
- Check distributions of data pre- and post-processing
- Check values of features prior to modeling
  - Are continuous features within ranges?
  - Are categorical features present in training data?

Model output monitoring

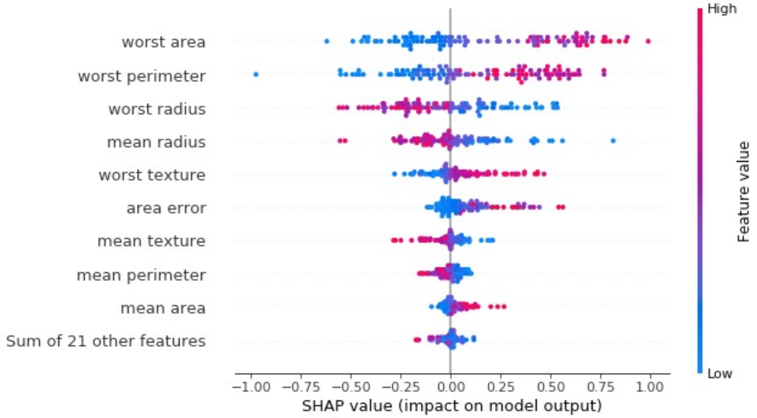
- Evolution of model performance metrics over time
  - Performance threshold to initiate retraining
- Distance between distribution of predicted labels and distribution of observed labels
  - Identify possible bias or concept drift





Model auditing

- Monitor model performance across demographic groups for bias
- Inspect the impact of features to ensure logical reasoning for predictions
  - **LIME (Local Interpretable Model-Agnostic Explanations)** – locally approximates a complex model with a simpler, interpretable surrogate model
  - **SHAP (Shapley Additive Explanations)** – assigns value to each feature based on its ability to change the prediction from the expected value

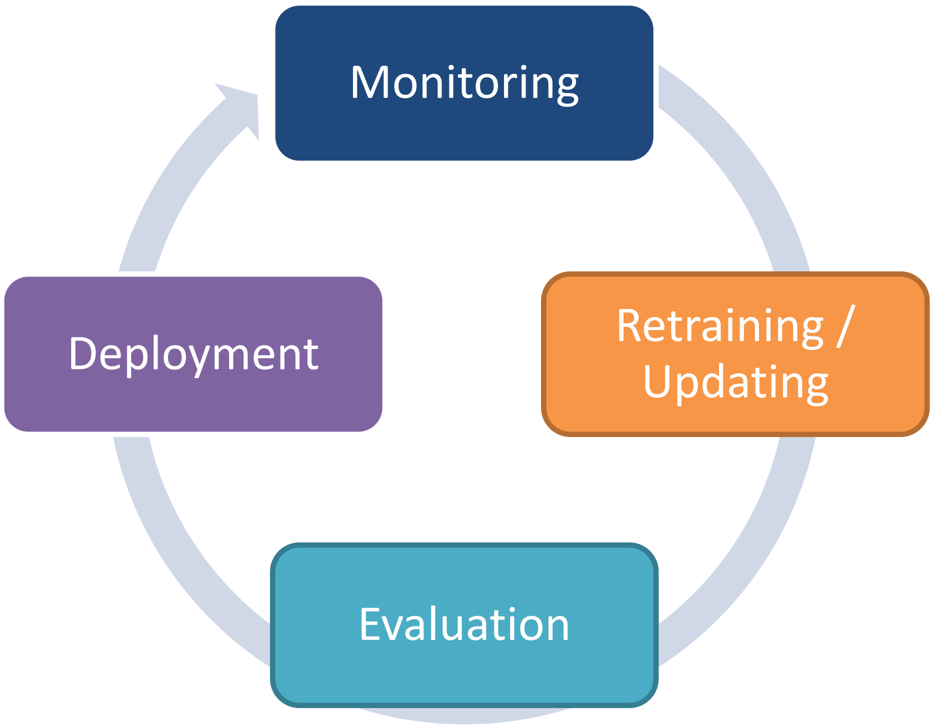


Model Maintenance

outrageously AMBITIOUS

Duke
PRATT SCHOOL of ENGINEERING

Model maintenance cycle



Retraining vs updating models

Retraining

- Using data collected since previous training to re-train model weights
- Enables model to adapt to changes in environment
- Can be done on a schedule or triggered

Updating

- Uses new data to re-do the modeling process
- Allows for adjustments to model form
- Can lead to identification of higher quality model
- Pruning eliminates unnecessary features

Why retrain models?

- Improve model performance with additional data
- Update model to reflect changing environment - reduce impact of data and concept drift
- Reduces threat of adversarial actors
- Recent data may be more important/relevant than old data

Scheduled retraining

- Retraining is done periodically on a fixed time schedule – e.g. days, weeks, months
- Requires knowledge of model's decay rate to ensure retraining prior to significant degradation
- Common when manual processes are involved in retraining – e.g. data collection

Triggered retraining

- Retraining is initiated when model performance degrades below a set threshold
- Ensures model stays fresh and responsive to changing environment
- Requires fully automated processes for model retraining

Continuous learning

- Model is trained on each new datapoint / batch as it comes in
- Primary use cases:
  - Very large datasets that make batch retraining difficult
  - Applications that require real-time responsiveness to quickly changing environment (e.g. social media)

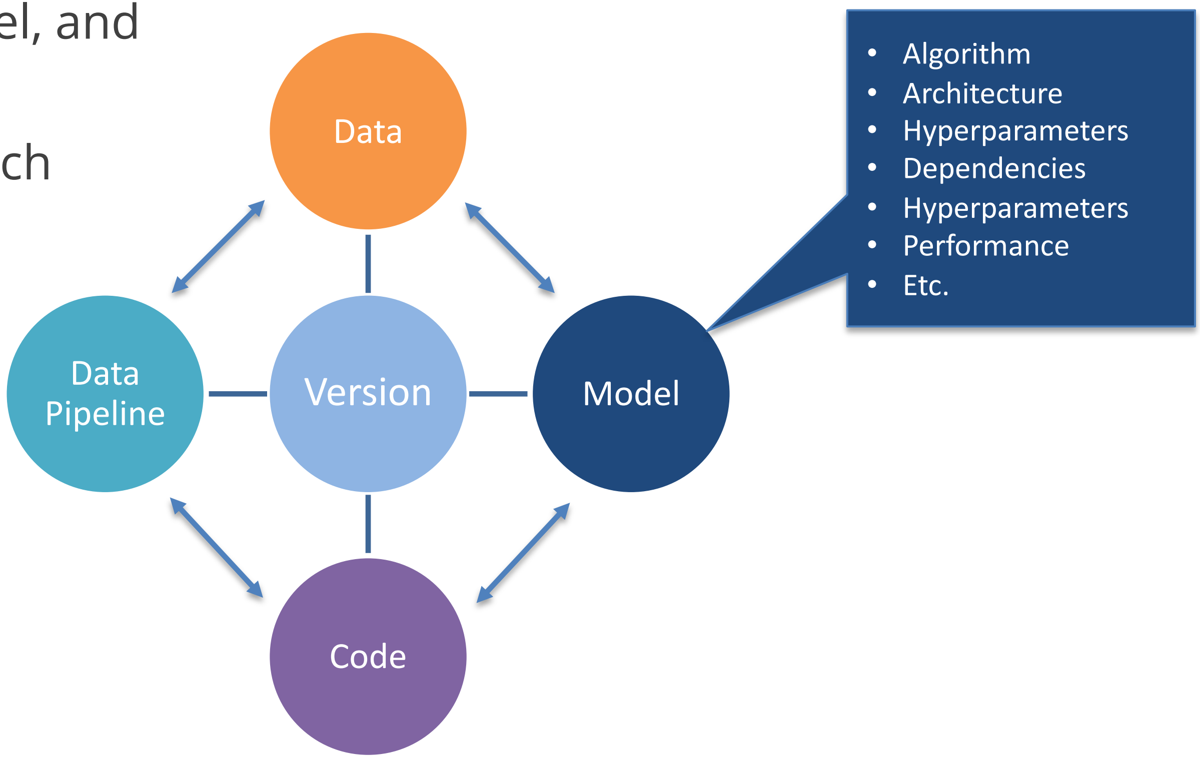
outrageously
AMBITIOUS

Model Versioning

Duke
PRATT SCHOOL of ENGINEERING

Versioning for ML systems

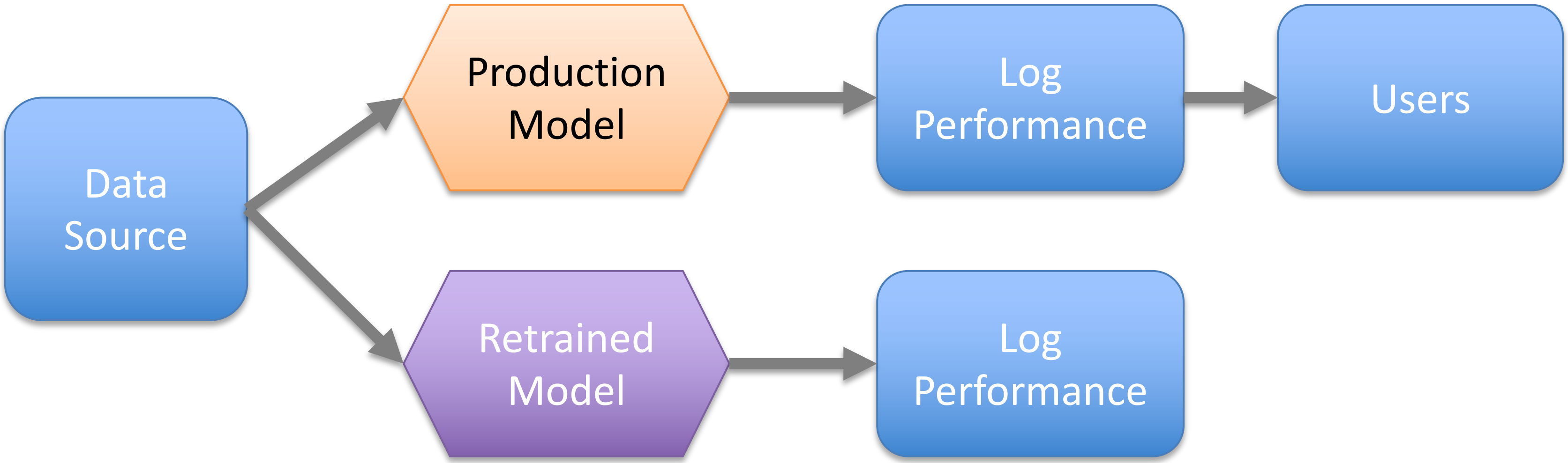
- Data, pipeline, model, and application code
- Dependencies of each



Why version models?

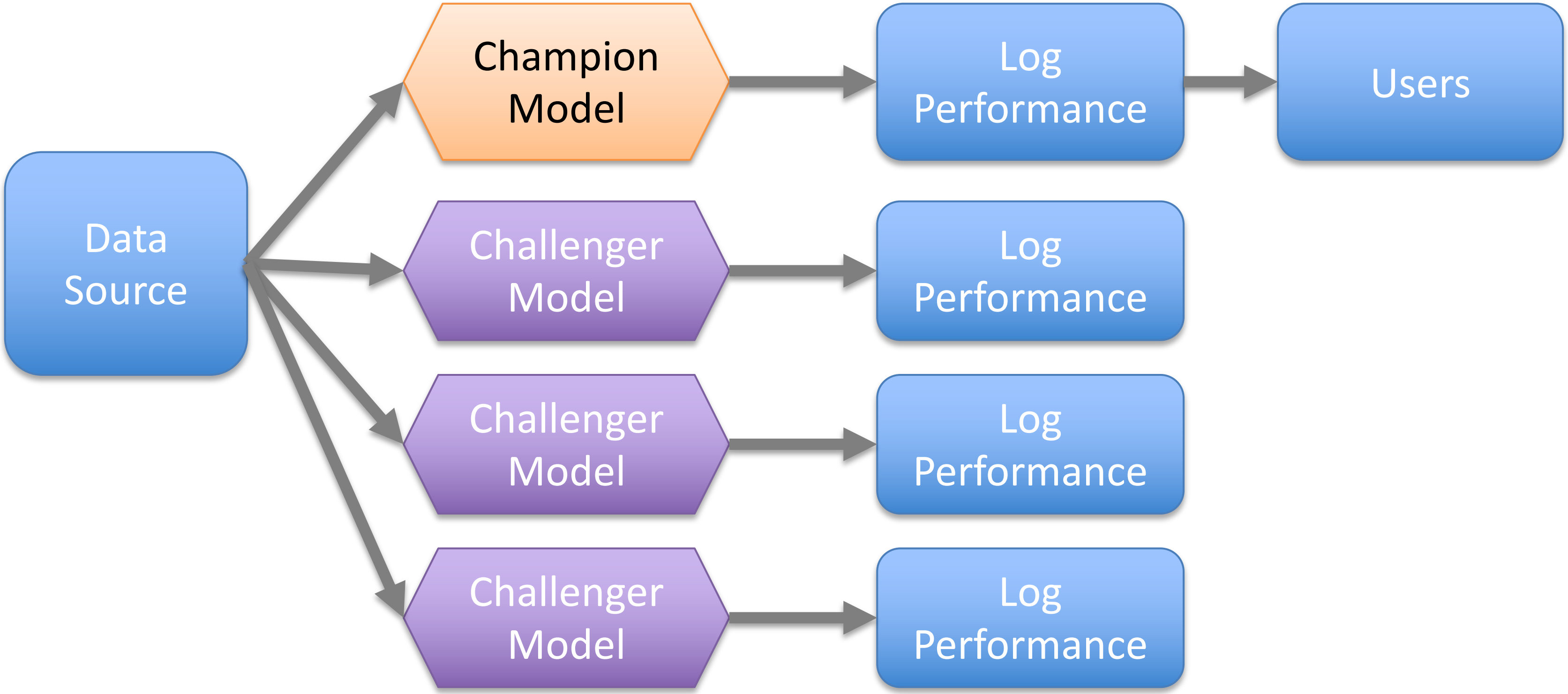
- Evaluate performance across iterations
- Track dependencies - ensure reproducibility
- Facilitates collaboration
- Build in rollback capability
- Enable testing of multiple models

Shadow releasing



If retrained model performance exceeds production model, move retrained model into production.

Champion-challenger testing



Model versioning

- Begin model versioning at the start of the project
- Will be used for both:
  - **Model development:** tracking iterative experiments
  - **Production:** shifting models over time

outraggeously AMBITIOUS

Organizational Considerations

Duke
PRATT SCHOOL *of* ENGINEERING

Ongoing model support

- Ongoing support of ML products in production includes:
  - Monitoring systems & processes
  - Model maintenance: retraining & updating
  - Model version management
- Requires ongoing commitment of team resources

Supporting users

- Team must also support users of the model
- Customer Service is often first point of contact for users
  - Equipped with understanding of what model does
  - Ability to explain model predictions
  - Recourse for model issues

Wrap-up

Wrap Up

- ML models in production face many more risks than normal software systems
- Models are dynamic - a robust monitoring and model maintenance plan ensures performance as the environment changes
- A model versioning system organizes the iterative model development work and facilitates optimization of the production model