11.12 : Monitoring Machine Learning Concepts

Why you need to monitor your model

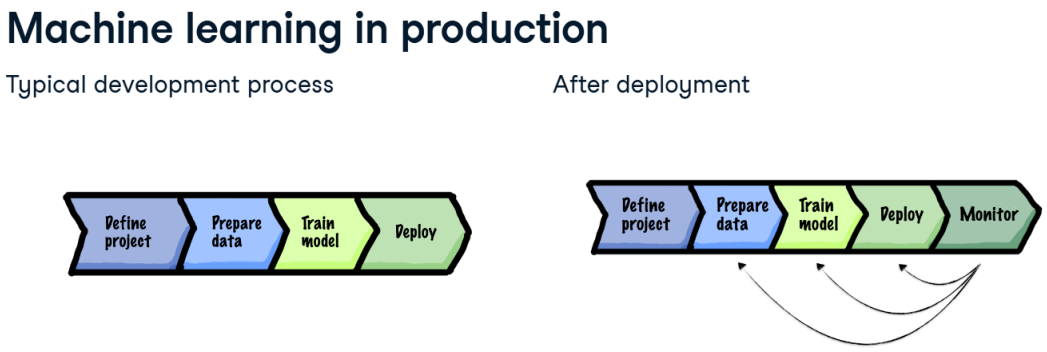
Why You Need to Monitor Your Model

00:00 - 00:25

Hakim, an experienced data engineer, data scientist, and CEO of NannyML, introduces the course on monitoring machine learning models in production.

Machine Learning in Production

00:25 - 01:07



Many companies are incorporating ML solutions, yet maintaining models in production requires ongoing monitoring, similar to babysitting. This vigilance ensures models stay safe and effective.

Reducing Risk of Failure

01:07 - 01:46

Model monitoring can prevent costly mistakes, like Zillow's failed model that caused significant financial loss. Monitoring helps catch issues like software bugs, data drift, and feature-target relationship changes.

Maximizing Business Impact

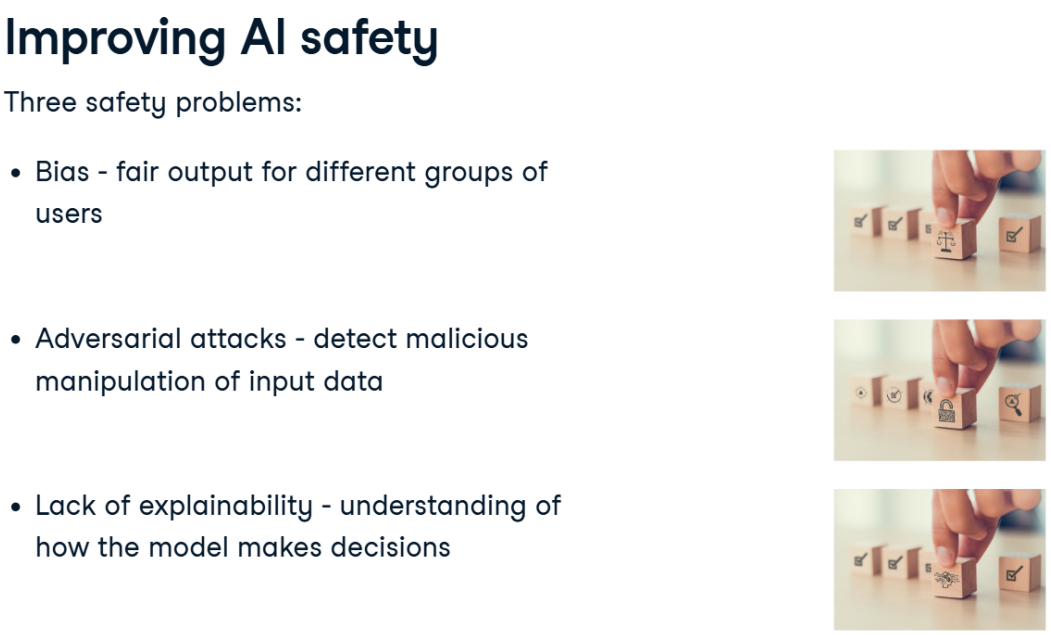
01:46 - 02:40

Monitoring also ensures models add value to the business. If models underperform on KPIs, they may need reconsideration. Monitoring can save on

cloud costs by enabling retraining only when necessary.

Improving AI Safety

02:40 - 03:25



Monitoring supports AI safety by preventing bias, detecting adversarial attacks, and improving model explainability, ensuring fair and safe model use in production.

Changing the World with Data

03:25 - 03:49

The ultimate goal is to enhance decision-making. Monitoring systems optimize automated processes, reducing errors, improving efficiency, and speeding up product deployment.

The ideal monitoring workflow

The Ideal Monitoring Workflow

00:00 - 00:06

Introducing the optimal workflow for monitoring models in production.

Monitoring Workflows

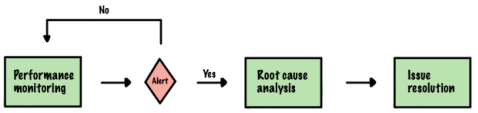
00:06 - 01:15

Traditional monitoring workflow

- Calculate technical performance
- Alert based on drifts in the input data
- Results in many false alerts

Ideal monitoring workflow

- Technical performance monitoring
  - Calculate and estimate performance
- Root cause analysis
  - Allows to link drifts with drops in performance



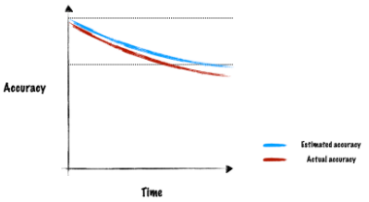
Traditional monitoring focuses on metrics like MSE or accuracy, but real-time ground truth is often unavailable. This leads to checking data distributions and alerting on any change, though not every distribution change harms performance, causing false alerts. The ideal approach emphasizes real-time performance estimation. If performance drops, a root cause analysis follows, examining distribution changes linked to the issue.

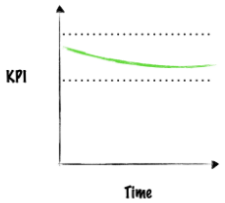
Monitoring Performance

01:15 - 02:21

Involves:

- Calculating performance - for technical metrics like accuracy
- Estimating performance - if ground truth is not available
- Measuring business impact - monitor key performance indicators





Performance monitoring is the first step, tracking metrics such as MSE or accuracy to evaluate production behavior. Even without ground truth, performance estimation is possible via error prediction models or confidence scores. Additionally, tracking business KPIs aligns model performance with business goals, where any KPI drop signals an issue requiring investigation.

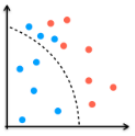
Root Cause Analysis

02:21 - 03:00

The goal is to investigate:

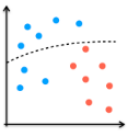
The original model

- Covariate shift - shifts in the input data distribution



Covariate shift

- Concept drift - changes in relationship between features and targets



Concept drift

The next step investigates why performance is dropping. Since the model is live, code issues can be ruled out, leaving data issues: covariate shift (input data changes) or concept drift (feature-target relationship changes). Detection methods help link these shifts to performance drops.

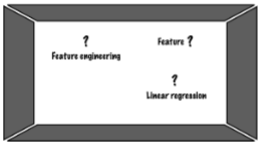
Issue Resolution

03:00 - 03:44

Possible solutions:

- Retraining - requires additional data and compute

- Refactoring the use case - take a step back and rethink used methods



- Changing the downstream processes - modify processes around the prediction

After identifying the problem, solutions may vary:

- **Retraining**: Often effective but needs labeled data and computational resources.
- **Refactoring the use case**: Revisiting features, engineering, or model types for robustness.
- **Changing downstream processes**: Adapting processes if the model lacks resilience.

Challenges of monitoring ML models

Challenges of Monitoring ML Models

00:00 - 00:06

Examining the challenges involved in monitoring machine learning models in production.

Machine Learning Project Components

00:06 - 00:36

ML models are part of complex systems requiring a mix of coding, data, and ML model knowledge, leading to unique monitoring challenges. Failures can occur across any of these areas, complicating monitoring at the production stage.

The Model Fails to Make Predictions

00:36 - 01:53

Sometimes, a model cannot generate output due to integration issues or system complexity:

- **Language barriers**: Integrating components built in different programming languages requires "glue" code, adding complexity and failure risk.
- **Code maintenance**: Library updates necessitate regular updates to stay compatible.
- **Scaling**: Increased users may strain infrastructure. Software monitoring systems can prevent or detect these issues promptly.

The Model Predictions Fail

01:53 - 03:05

A model can still produce outputs even if predictions become inaccurate. The causes often include:

- **Covariate Shift**: Changes in input feature distribution over time, detectable with statistical tests, though these often produce false alerts.
- **Concept Drift**: A change in the input-target relationship that affects business outcomes, challenging to detect.

Availability of Ground Truth

03:05 - 03:43

Target values are essential to monitor model accuracy, but real-world scenarios often lack immediate ground truth. For example, in demand forecasting, the actual demand may not be known for months, delaying evaluation.

Monitoring technical performance directly

Covariate Shift - Performance Relationship

00:05 - 02:32

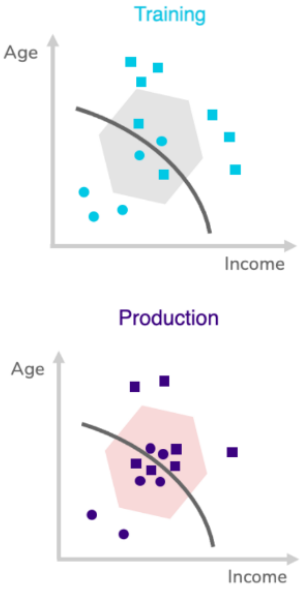
Covariates, or input features, can shift in production, impacting model performance in different ways:

1. **Shift to regions of certainty**: When production data moves to areas where the model has high certainty, performance may remain stable or improve, as in the example where more high-income individuals apply for loans.
2. **Shift to underrepresented regions**: If the data shifts to segments underrepresented in the training data, performance effects are unpredictable. For example, if more people in tech start applying for loans, accuracy may decrease.
3. **Shift to uncertain regions**: When data shifts to uncertain regions near the decision boundary, performance will likely degrade. For instance, if more applicants live in Manhattan, where predictions are harder, accuracy decreases.

Guaranteed Negative Impact

02:32 - 03:08

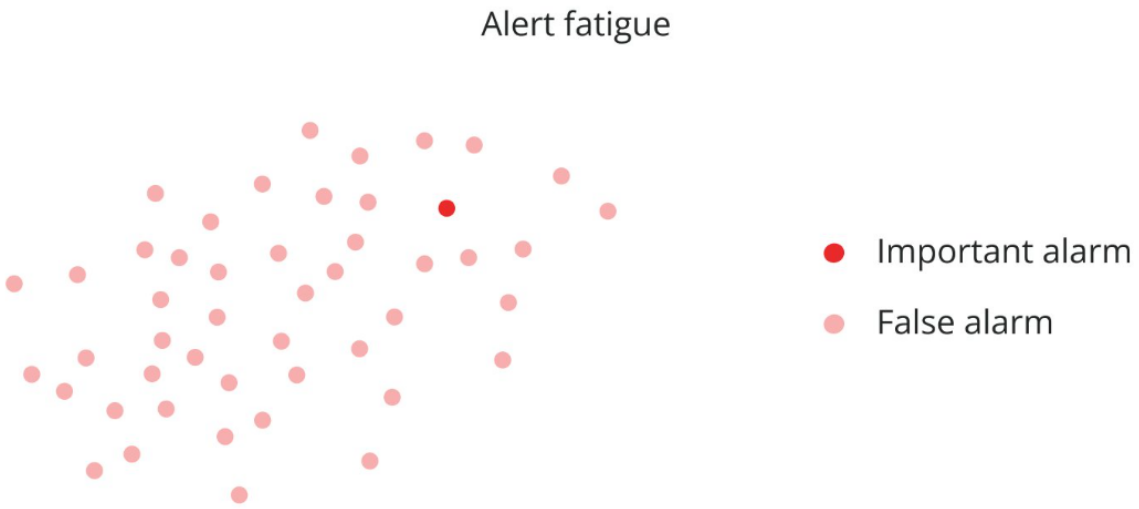
Covariate shift to uncertain regions always negatively impacts performance



Data shifts near the decision boundary, such as an increase in Manhattan-based applicants, consistently reduce performance due to higher prediction uncertainty in these regions.

False Alerts Problem

03:08 - 03:52

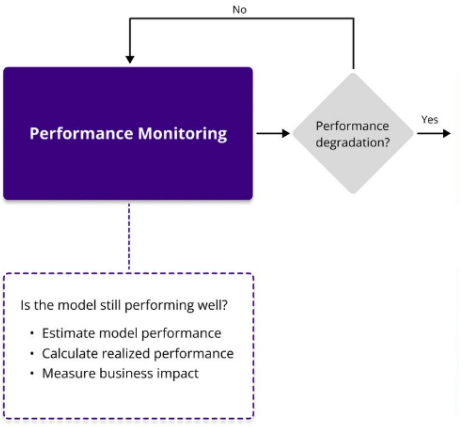


Standard drift detection methods lack specificity, often alerting for any shift, regardless of its effect on model performance. This results in false alarms, especially when irrelevant feature shifts occur, which can lead to alert fatigue and distract from real issues.

The Importance of Technical Performance

03:52 - 04:13

- Direct metric of how well the model performs the task at hand
- Reflects any silent model failure
- Removes the overload of false alerts



Technical performance directly reflects model efficacy in production. Monitoring performance avoids false alarms by focusing on actual predictive accuracy, making it the foundation of effective model monitoring workflows.

Availability of ground truth

Availability of Ground Truth in Production

00:00 - 00:05

Exploring how the availability of ground truth impacts the monitoring of machine learning models in production.

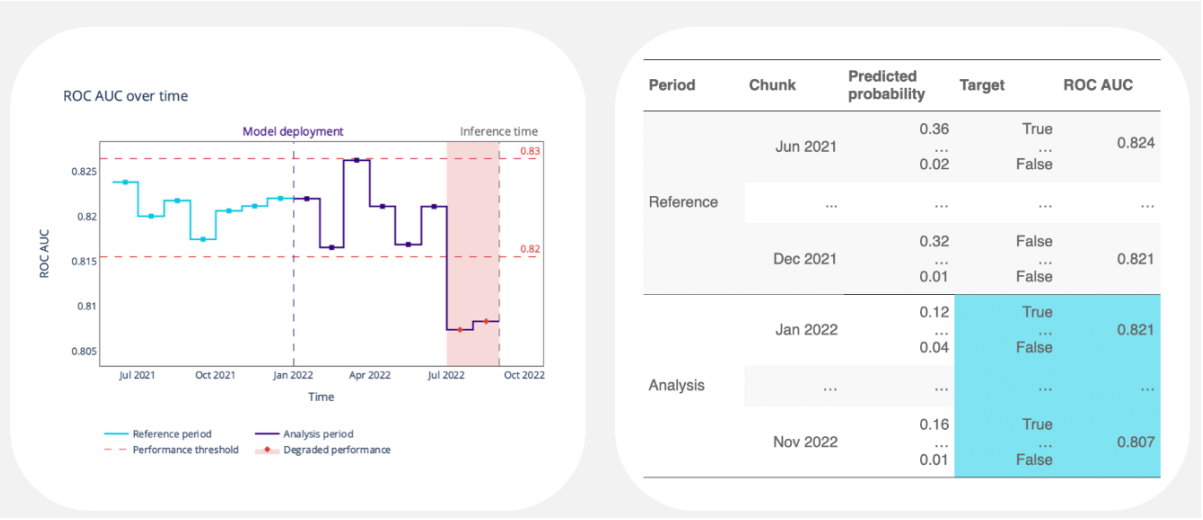
Instant Ground Truth

00:05 - 00:39

In some cases, ground truth is immediately available, allowing for real-time performance monitoring. For example, in taxi arrival estimation, the model's prediction can be evaluated as soon as the taxi arrives. This scenario allows for near real-time performance feedback, enabling rapid detection and resolution of issues, and provides the most accurate performance assessment.

Production Data - Instant

00:39 - 01:05



Period	Chunk	Predicted probability	Target	ROC AUC
Reference	Jun 2021	0.36 ... 0.02	True ... False	0.824
	...	...	...	...
	Dec 2021	0.32 ... 0.01	False ... False	0.821
Analysis	Jan 2022	0.12 ... 0.04	True ... False	0.821
	...	...	...	...
	Nov 2022	0.16 ... 0.01	True ... False	0.807

A performance graph for a classification model displays the "reference period" for testing data and an "analysis period" for production data. When instant ground truth is available, metrics such as ROC-AUC can be monitored in real time. A performance drop, for example, between June and October, signals the need for an investigation to understand and resolve the issue.

Delayed Ground Truth

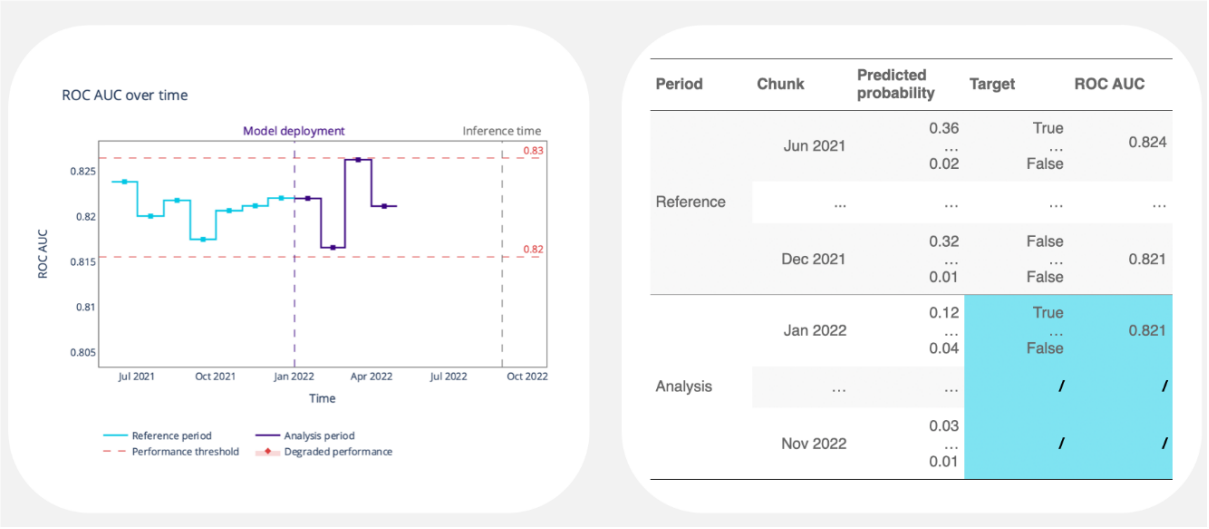
01:05 - 01:46

- Delay depends on the application
- A possible scenario is loan default prediction
- Unknown performance in the meantime
- Requires performance estimation

In many cases, ground truth is delayed. For instance, in loan default prediction, the actual outcome (whether a borrower defaults) might not be confirmed until months or years later. During this waiting period, the model's technical performance is unknown, making it difficult to assess if predictions remain reliable. A performance estimation approach is necessary until the true outcomes become available.

Production Data - Delayed

01:46 - 02:07

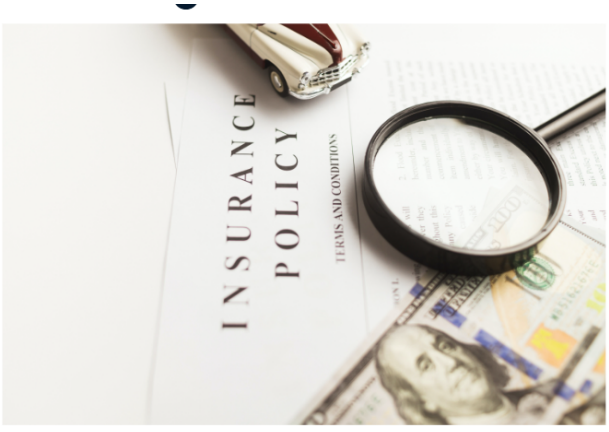


Period	Chunk	Predicted probability	Target	ROC AUC
Reference	Jun 2021	0.36 ... 0.02	True ... False	0.824
	...	...	...	...
	Dec 2021	0.32 ... 0.01	False ... False	0.821
Analysis	Jan 2022	0.12 ... 0.04	True ... False	0.821
	...	...	/	/
	Nov 2022	0.03 ... 0.01	/	/

With delayed ground truth, gaps appear in the performance graph (e.g., a missing ROC-AUC score from May to November). These gaps complicate ongoing model evaluation, as target values are unavailable during this period, making it difficult to validate the model's accuracy and reliability.

Absent Ground Truth

02:07 - 02:36

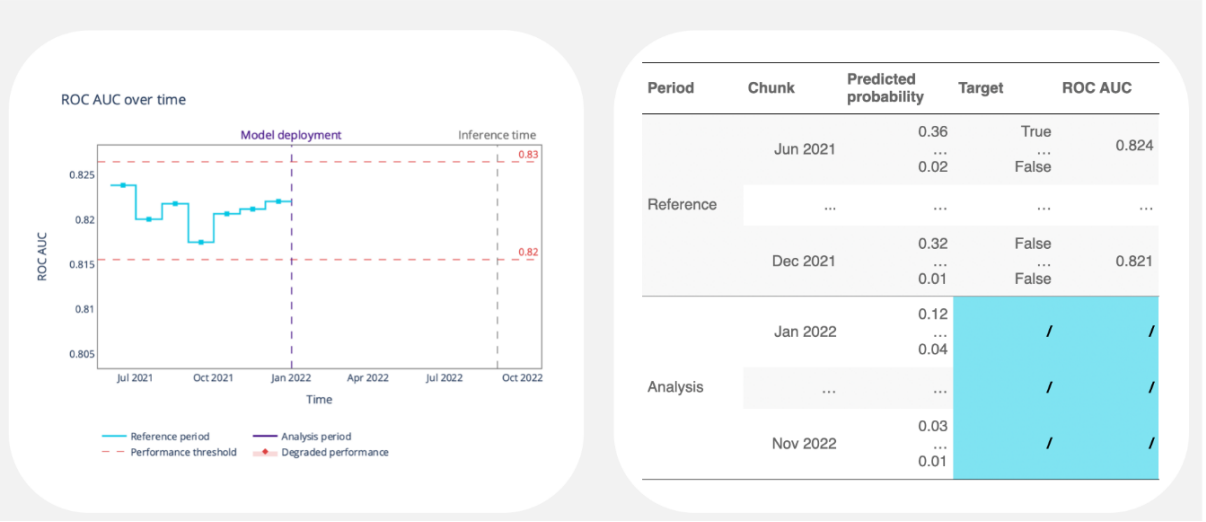


- Present in fully-automated processes
- A possible scenario is insurance pricing
- Actual performance is unknown
- Requires performance estimation

In some scenarios, ground truth may be entirely absent, such as with machine learning models used in insurance pricing. Determining the model's true performance could be prohibitively expensive or time-consuming, leaving the model's business value uncertain. Performance estimation becomes essential to gauge its effectiveness in these cases.

Production Data - Absent

02:36 - 02:50



Period	Chunk	Predicted probability	Target	ROC AUC
Reference	Jun 2021	0.36 ... 0.02	True ... False	0.824
	...	...	...	...
	Dec 2021	0.32 ... 0.01	False ... False	0.821
Analysis	Jan 2022	0.12 ... 0.04	/	/
	...	...	/	/
	Nov 2022	0.03 ... 0.01	/	/

When ground truth is absent, there are no target values, leaving the performance graph blank post-deployment. Without this data, assessing the model's predictions and trustworthiness becomes challenging, necessitating performance estimation for ongoing evaluation.

Performance estimation

Overview of Performance Estimation

00:06 - 00:33

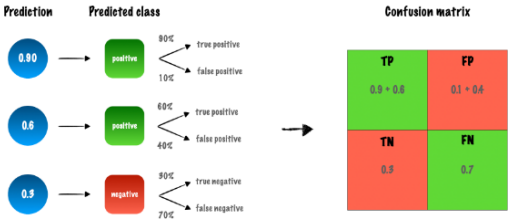
When ground truth is delayed or absent, estimating performance becomes essential. Two algorithms help with performance estimation in these situations:

- **CBPE (Confidence-Based Performance Estimation)** for classification tasks.
- **DLE (Direct Loss Estimation)** for regression tasks.Let's examine how each algorithm functions.

CBPE - Confidence-Based Performance Estimation

00:33 - 01:20

- Estimates a confusion matrix
- Allows to calculate classification metrics like accuracy, precision, recall
- Captures the impact of covariate shift on the model



CBPE estimates a model's performance for classification tasks by using confidence scores to approximate the confusion matrix. For example, a model predicting a score of 0.9 for a case implies it is 90% likely correct. This process is repeated for all cases, and the aggregated scores create an estimated confusion matrix, enabling calculations of metrics like accuracy, precision, recall, and F1-score. If the model is impacted by covariate shift, the performance estimation will reflect this.

CBPE - Important Considerations

01:20 - 02:28

- No covariate shift in the unseen regions
- No concept drift is present in the incoming data
- Requires a probability calibration

CBPE is effective but assumes specific conditions in production:

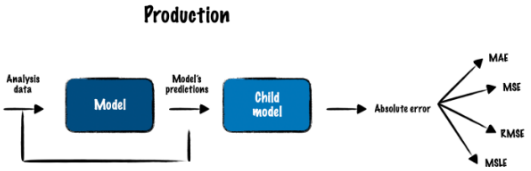
1. **No covariate shift in unseen regions:** For instance, a model trained with data from 40-70-year-olds may yield unreliable results if deployed in a context with customers below 40.
2. **Absence of concept drift:** Concept drift changes the relationship between inputs and targets, which can make the model's predictions inaccurate.

3. **Calibration required:** The model's probability scores should be calibrated to reflect actual outcomes (e.g., a 0.9 confidence score means the model is correct 90% of the time). Calibration can be applied before production deployment.

DLE - Direct Loss Estimation

02:28 - 03:07

- Predicts the absolute error of the model
- Uses an external child model
- Allows to calculate various regression metrics like MAE, MSE, MSLE
- Captures the presence of covariate shift in the input data



DLE estimates model performance in regression tasks by predicting absolute error. It uses a secondary model (e.g., LightGBM) trained on reference data and main model predictions to calculate regression metrics like MAE and MSLE. DLE can indicate if a covariate shift impacts the model's output.

DLE - Important Considerations

03:07 - 03:33

- No covariate shift in the unseen regions
- No concept drift is present in the incoming data
- Extra complexity

Similar to CBPE, DLE assumes:

1. **No covariate shift in unseen regions**

2. **Absence of concept drift**
3. **Increased complexity:** Since DLE uses an additional model for estimation, it can require more computational resources, impacting the overall system's efficiency.

What is covariate shift?

1. What is Covariate Shift?

00:00 - 00:13

Introduction to covariate shift, a critical form of data drift in real-world applications.

2. Definitions

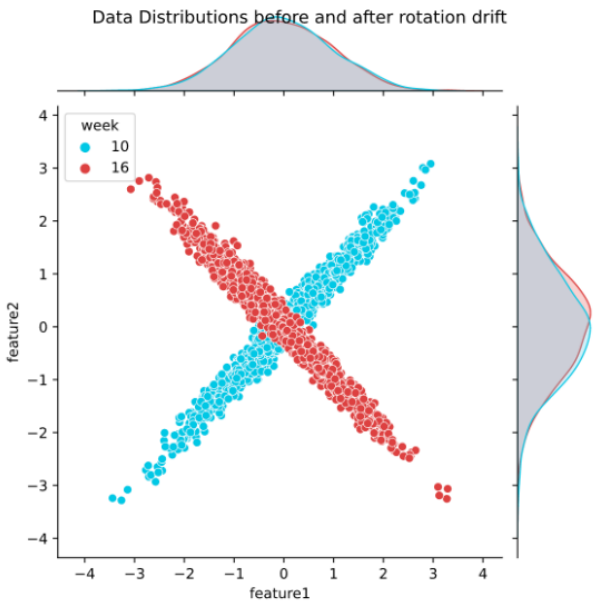
00:13 - 00:54

- covariate variables $=$ input features
- $P(X)$ changes
- joint probability $P(Y|X)$ remains the same
- changes in the joint distribution of the covariates

Covariate shift is defined as changes in the input features' distribution, denoted as **P(X)**, while the output conditional probability, **P(Y|X)**, remains unchanged. The term "covariate" refers to the input features, and the key aspect of covariate shift lies in the "joint" distribution of the covariates, meaning that the combined distribution of features changes rather than each feature individually.

3. Why Joint Probability Distribution?

00:54 - 01:39



In some cases, if each feature is analyzed independently, there may be no visible change in its distribution, but the joint distribution can still shift significantly. For example, if two features are mistakenly swapped during data processing, the model could misinterpret feature relationships, leading to prediction errors. This highlights the necessity of focusing on joint probability distribution when defining covariate shift.

4. Why Does Covariate Shift Occur?

01:39 - 02:21

Covariate shift can happen due to:

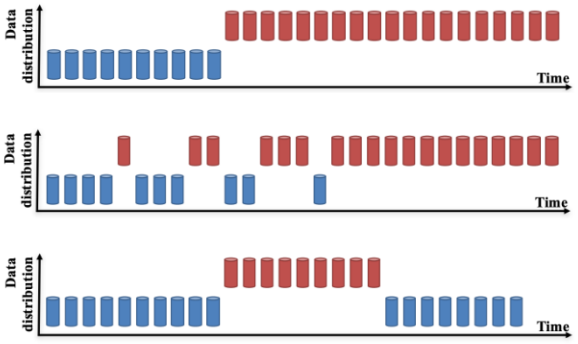
1. **Real-world changes** - Economic, social, or environmental shifts alter feature distributions.
2. **Variations in data sources** - For instance, using data from native English speakers and then applying the model to a broader range of speakers.
3. **System and environment updates** - For example, using training data from a previous software version and deploying it on an updated one.

5. How Does Covariate Shift Occur?

02:21 - 03:04

Dynamics of the changes in the distribution:

- Sudden
- Gradual
- Seasonal



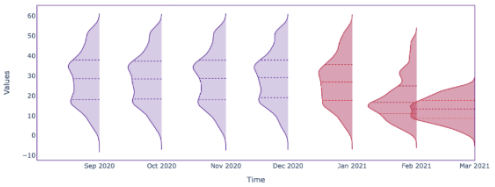
Covariate shift can manifest in three dynamics:

- **Sudden shifts** - Abrupt changes, like those seen during the COVID-19 pandemic.
- **Gradual shifts** - Evolving trends over time, such as changing social media preferences.
- **Seasonal shifts** - Regular, predictable changes, such as increased demand for taxi services in colder weather.

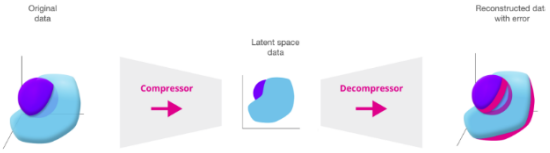
6. How to Detect Covariate Shift?

03:04 - 03:40

Univariate method



Multivariate method



Covariate shift detection methods include:

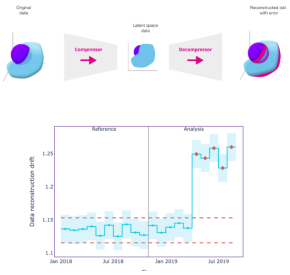
- **Univariate methods** - Monitoring each feature's distribution individually.
- **Multivariate methods** - Using dimensionality reduction (e.g., PCA) to detect changes in joint distributions, as shifts in reconstruction error can indicate drift.

How to detect covariate shift

2. Multivariate Drift Detection

00:07 - 00:47

- Looks for changes in joint distribution
- Uses the PCA algorithm for data compression
- Uses reconstruction error as a measure of drift



To diagnose a decline in model performance, the first step is to detect any covariate shift in the joint data distribution. Multivariate drift detection leverages PCA to compress data and capture its internal structure, filtering out noise. After reducing dimensions, inverse PCA is used to reconstruct the data. Comparing the reconstruction error to a baseline allows us to assess any shifts in the input data distribution.

3. Univariate Drift Detection

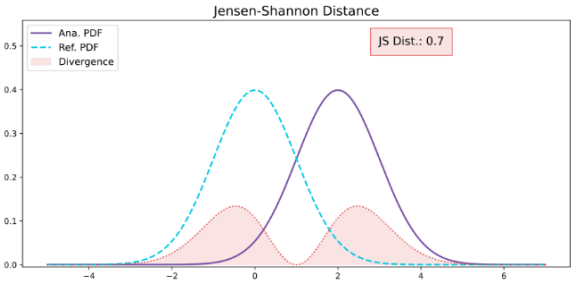
00:47 - 01:18

After confirming drift using multivariate detection, the next step is to identify individual features responsible for the drift. Feature-specific drift detection methods vary based on data type: categorical (distinct groups) and continuous (real values in a range).

4. Continuous Methods - Jensen-Shannon

01:18 - 01:36

- Measures the similarity of two distributions
- Range [0, 1]
- Catches meaningful low-magnitude drifts

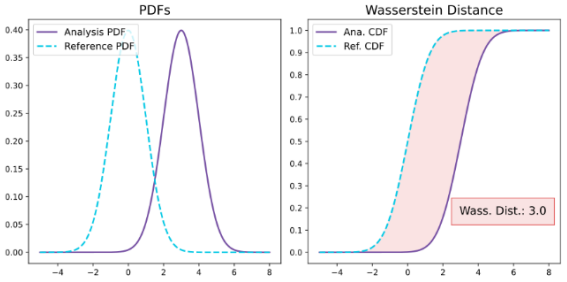


The Jensen-Shannon distance measures distribution similarity within a 0 to 1 range using Kullback-Leibler divergence, making it ideal for detecting subtle changes in data.

5. Continuous Methods - Wasserstein

01:36 - 01:54

- The minimum effort needed to transform one distribution into another
- Range [0, +inf]

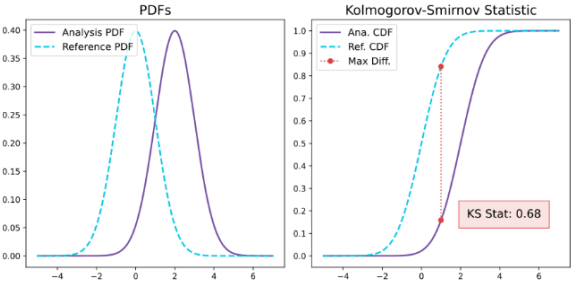


The Wasserstein distance calculates the minimum effort to convert one distribution into another. With a range from 0 to infinity, it's sensitive to outliers, which can impact accuracy.

6. Continuous Methods - Kolmogorov-Smirnov

01:54 - 02:15

- Maximum distance of the cumulative distribution functions
- Range [0, 1]
- Prone to false positives

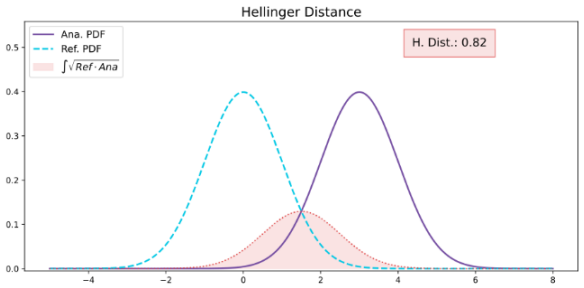


The Kolmogorov-Smirnov test evaluates the maximum difference in cumulative distribution functions between samples, ranging from 0 to 1. It may raise false positives in large datasets, potentially misidentifying meaningful drifts.

7. Continuous Methods - Hellinger

02:15 - 02:45

- Overlap between distributions
- Range [0, 1]
- Doesn't differentiate between strong shifts



Continuous methods - Recommendation

- Jensen-Shannon and Wasserstein generally perform well

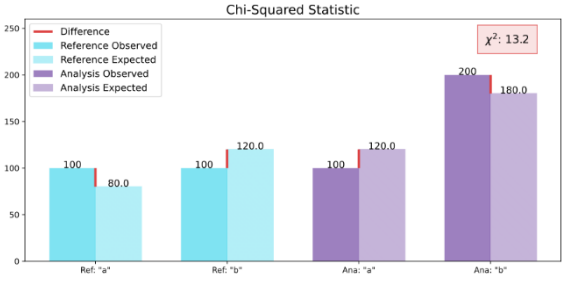
The Hellinger distance assesses distribution overlap and can be applied to both categorical and continuous data. However, it only captures shifts with no overlap, making it less effective when distributions are close but not identical. For continuous data, Jensen-Shannon or Wasserstein distances are generally more effective.

8. Categorical Methods - Chi-squared

02:45 - 03:01

Categorical methods - Chi-squared

- Sensitive in changes for low-frequency categories

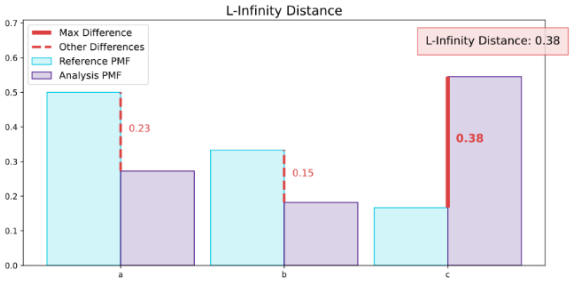


The Chi-squared test monitors categorical shifts, particularly those with low-frequency changes, where even slight adjustments can impact results significantly.

9. Categorical Methods - L-Infinity

03:01 - 03:16

- Identifies the most significant shift across all categories

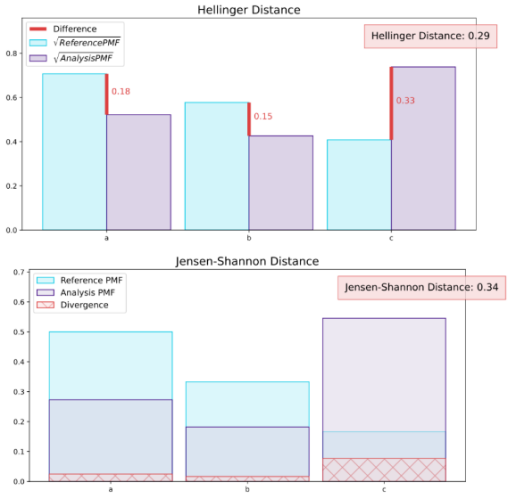


The L-infinity method identifies the largest category difference, handling large category sets well and effectively detecting significant shifts in individual categories.

10. Categorical Methods - Jensen-Shannon and Hellinger

03:16 - 03:36

- Jensen-Shannon or L-Infinity when dealing with many categories
- L-Infinity distance to detect changes in individual categories



The Jensen-Shannon and Hellinger methods are versatile and suitable for many variable types. For detecting shifts in specific categories, L-Infinity or Jensen-Shannon distances are recommended.

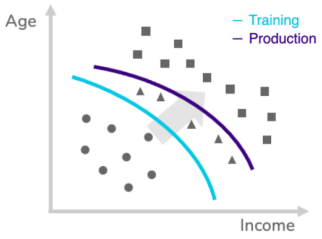
This approach helps isolate drifts in both continuous and categorical features, improving root cause analysis for model performance issues.

What is concept drift?

2. Definition

00:05 - 00:25

- Change in relationship between the model inputs and the target
- $P(Y|X)$ changes, $P(X)$ stays the same



Concept drift occurs when the relationship between model inputs and the target changes, meaning the training relationship P(Y ∣ X)P(Y|X)P(Y ∣ X) differs from the production P(Y ∣ X)P(Y|X)P(Y ∣ X) while P(X)P(X)P(X) (input distribution) remains the same.

3. Why Drift Happens?

00:25 - 01:29

Concept drift can be caused by:

- **External events** like policy changes or unexpected events that alter data distributions suddenly.
- **Seasonality** through regular patterns like daily or yearly cycles (e.g., increased demand during holidays).
- **Data-generating process changes** due to interface updates or app modifications, leading to new user interaction patterns.
- **Evolving user behavior** where shifts in preferences and habits change the incoming data patterns.

4. The Dynamics of Concept Drift

01:29 - 02:21

Concept drift manifests in three main types:

- **Sudden drift** occurs quickly due to unforeseen events, like COVID-19.
- **Gradual drift** happens slowly as new concepts replace old ones, e.g., inflation affecting pricing over time.
- **Reoccurring drift** appears cyclically, such as shifts in consumer behavior during holiday seasons.

5. Effects of Covariate Shift on Concept Drift

02:21 - 03:43

Covariate shift and concept drift can appear separately or together and interact in two ways:

- **Negative Interaction**: Concept drift effects decrease when covariate shift reduces the influence of affected groups. For instance, if the average income of loan applicants shifts to a higher range, the impact of concept drift on low-income default predictions decreases.
- **Positive Interaction**: Concept drift effects intensify when covariate shift amplifies the influence of affected groups. If the applicant income shifts lower, concept drift increases prediction errors as more low-income applicants are prone to default.

This dynamic interaction highlights the importance of addressing both types of drift in monitoring workflows to capture their combined effects on model performance.

How to handle concept drift?

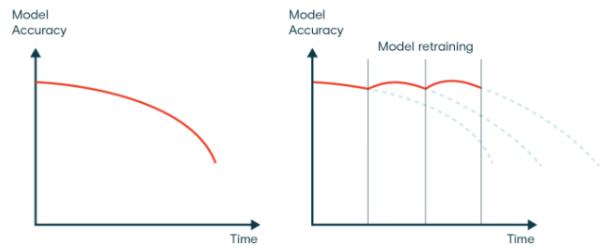
2. Concept Drift Detection

00:06 - 01:03

Detecting concept drift is challenging and lacks a unified solution, as it is still an active area of research without established industry standards. Many methods focus on **error-based detection**, monitoring error evolution for sudden or gradual changes, which can indicate concept drift. However, this approach relies on a constant stream of ground truth data, which isn't always available. Another technique involves training a new model on a mix of training and production data; a significant difference in performance between the original and new model suggests concept drift. This approach, however, is costly, especially with larger models and extensive datasets.

3. Retraining

01:03 - 01:45



Pros :

- keep the model up-to-date with recent patterns

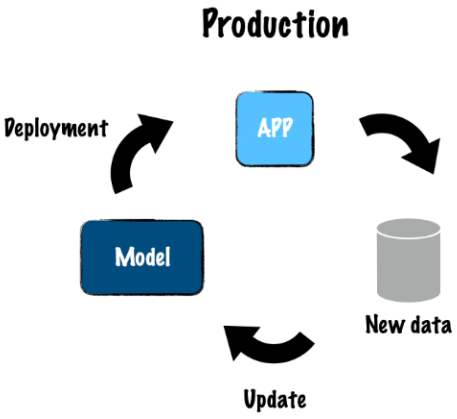
Cons :

- increased costs and risk of failure
- doesn't provide the root cause of the problem

Periodic or event-triggered retraining helps models stay updated with new data patterns. While effective, frequent retraining increases the risk of update failures and demands more computational resources, driving up costs. Furthermore, retraining is not a catch-all solution; underlying issues, such as **downstream process changes, data leakage, or training-serving skew**, may also degrade model performance and require investigation rather than retraining.

4. Online Learning

01:45 - 02:42



Pros :

- real-time adaptation to changing conditions

Cons :

- requires constant access to ground truth
- sensitive to noise
- needs careful parameter tuning

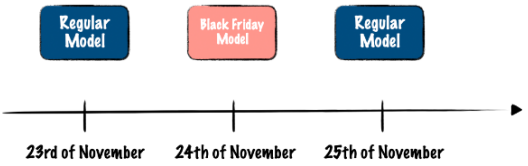
Online learning (incremental or streaming learning) allows continuous model updates with incoming data, enabling real-time adaptation to concept drift. Benefits of online learning include handling evolving data streams efficiently and providing timely responses to data changes. It processes data instance-by-instance, making it computationally feasible for large-scale, high-velocity

scenarios. However, it has limitations: online learning needs regular ground truth data for updates, is sensitive to noisy data, and requires careful tuning to maintain performance over time.

5. Other Resolutions

02:42 - 03:21

- A event-specific model for reoccurring events



- Weighting the importance of new data
  - with most focus on newer data, model can adapt easier

In cases of **reoccurring concept drift** (e.g., during events like Black Friday), where behavior patterns deviate temporarily, models trained on regular data may become less effective. Possible solutions include:

- **Deploying event-specific models** that are trained on historical event data for these periods.
- **Weighted data importance**: Assigning higher weights to recent data helps models prioritize recent trends, adapting more quickly to shifts. This approach, however, risks model degradation if new data is overweighted.

These solutions emphasize flexible strategies for maintaining model relevance under dynamic conditions.