Uber Michelangelo: End-to-End ML Platform Case Study

1. Introduction & Project Context

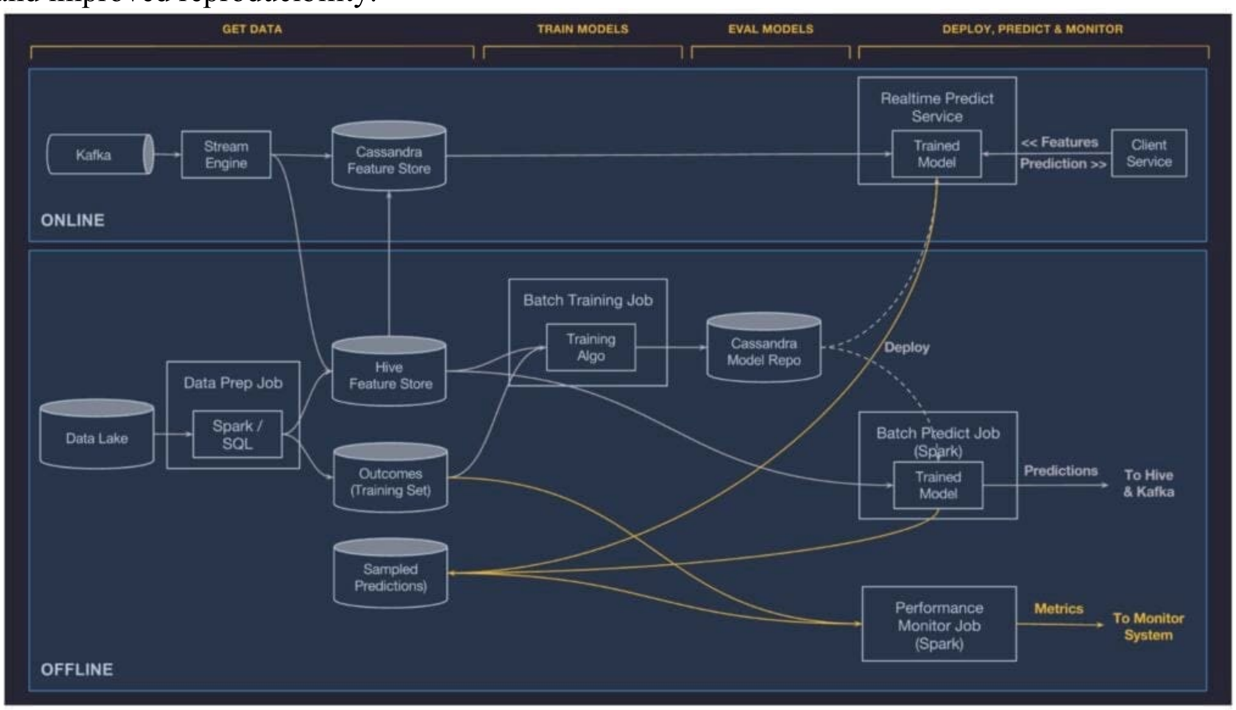
Uber operates in a **data-rich, real-time environment** where decisions—like ETA prediction, fraud detection, and surge pricing—must be made in milliseconds. Before Michelangelo, Uber's ML workflows were fragmented, with:

- Separate tools for different teams
- Manual deployment steps
- No unified monitoring framework

Michelangelo was built as an **internal ML-as-a-service** platform to unify this process. It offers:

- Centralized feature storage
- Automated model training and validation
- Seamless online/offline deployment
- Continuous monitoring

By centralizing infrastructure, Uber reduced engineering overhead, accelerated experimentation, and improved reproducibility.



Explanation: Shows the full architecture: data ingestion, feature store, model training, deployment, and monitoring layers.

---

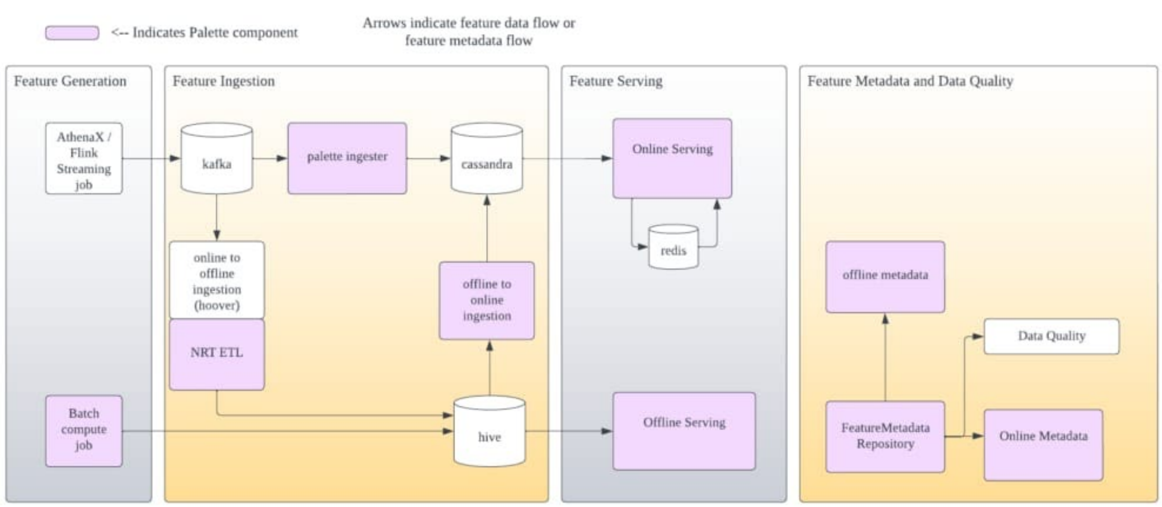
2. Data Preprocessing

Uber collects data from diverse sources:

- **GPS logs** from driver and rider devices
- **Transaction records** from payment systems
- **User interaction data** from apps
- **Third-party data** such as traffic feeds and maps

Key Steps:

1. **Data Cleaning** – Remove invalid GPS coordinates, fix inconsistent time zones, impute missing values using median or domain rules.
2. **Outlier Detection** – Use statistical thresholds and Isolation Forest models to detect anomalies like extreme detours.
3. **Feature Scaling** – Apply Min-Max scaling to normalize features for gradient-based models.
4. **Versioning** – Store every dataset and feature definition in **Palette** with version control for reproducibility.
5. **Real-Time Feature Computation** – Calculate features such as "current traffic speed" from streaming data in under 50 ms.



Explanation: Depicts how raw data is ingested, transformed into reusable features, stored in both offline and online stores, and served to models in training and production.

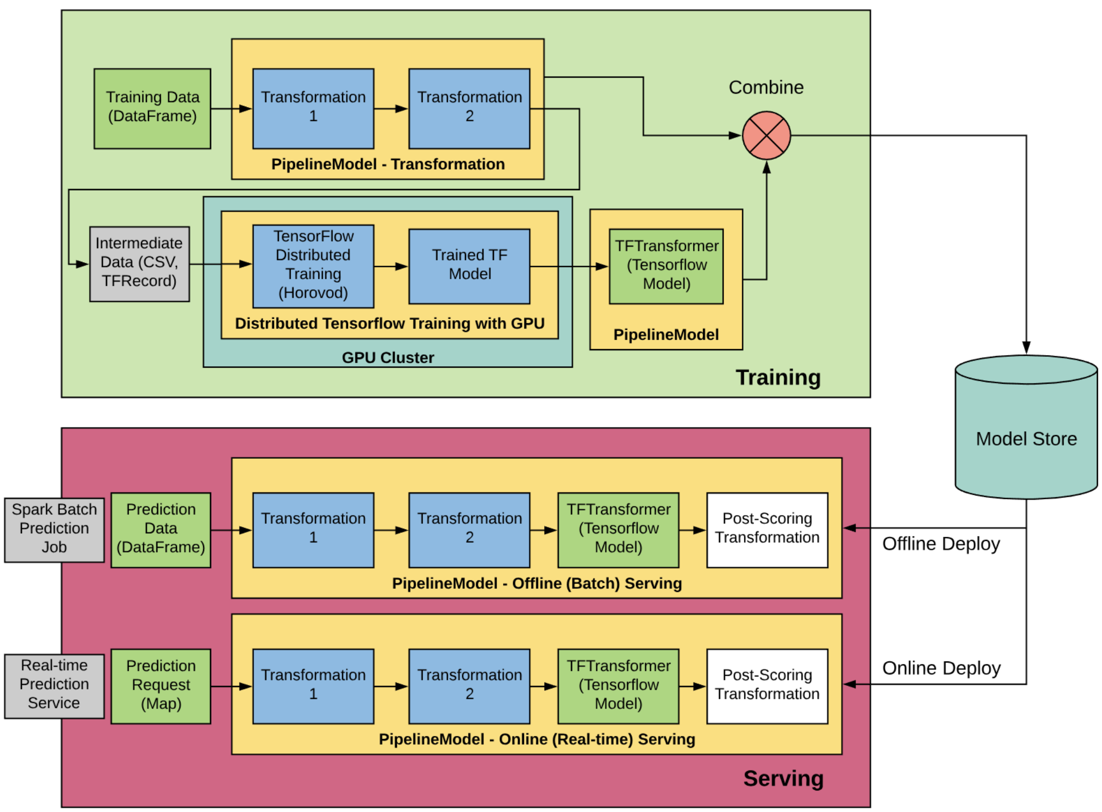
3. Model Selection & Validation

Michelangelo supports **multiple ML algorithms**:

- **XGBoost / LightGBM** – Structured tabular data
- **TensorFlow / PyTorch** – Deep learning for unstructured data
- **Time Series Models** – Demand and supply forecasting

Validation Process:

- **Offline Validation:** K-fold cross-validation, holdout test sets
- **Hyperparameter Optimization:** Bayesian optimization for tuning model parameters
- **Online Validation:** Live A/B testing with traffic splitting
- **Shadow Deployment:** New models run silently alongside production models to compare results without affecting live opera

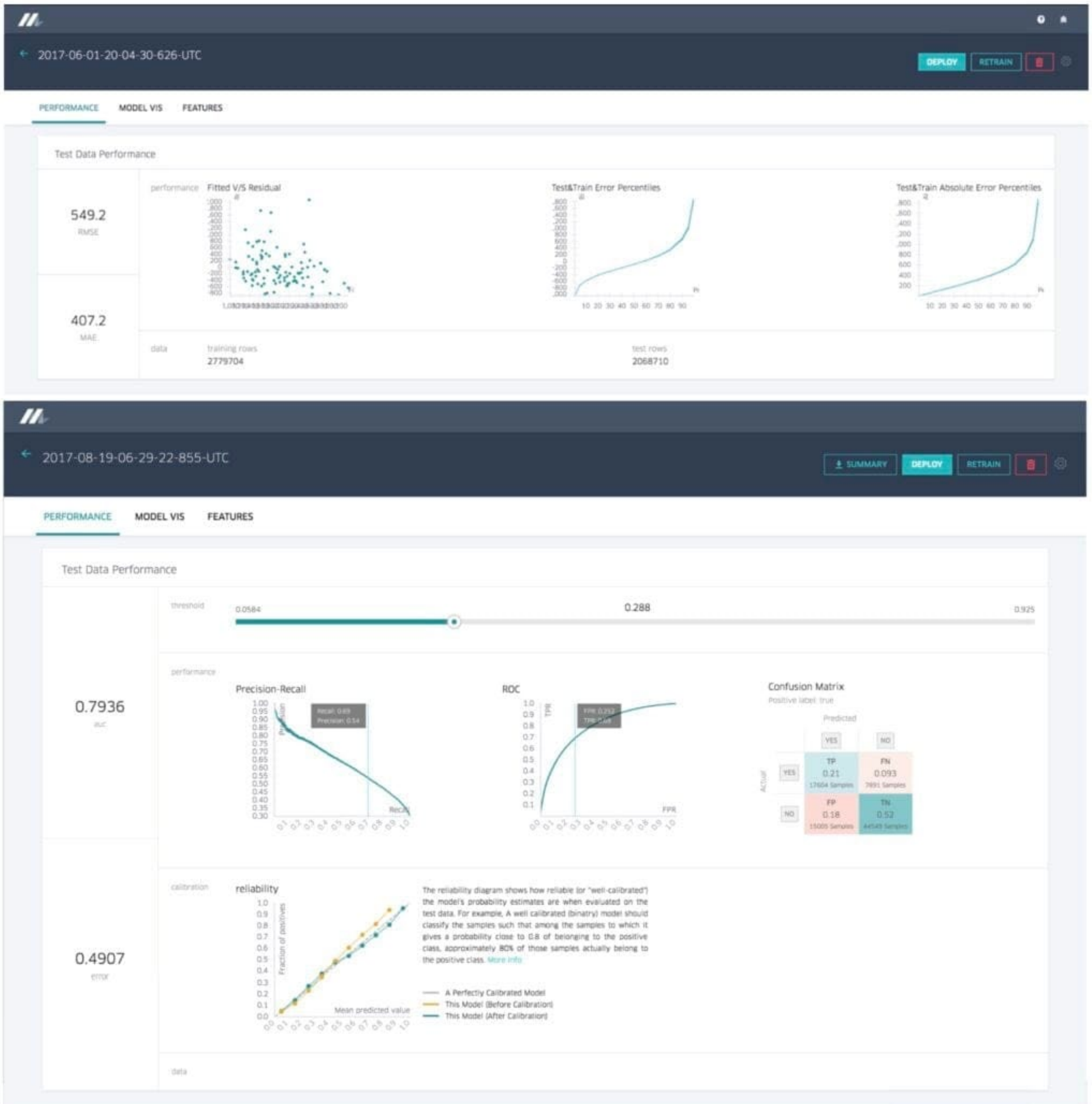


Explanation: Demonstrates the iterative lifecycle from data prep to model deployment and feedback loops.

4. Performance Metrics

Uber evaluates models on:

- **Classification:** Precision, Recall, F1-score, ROC-AUC
- **Regression:** RMSE, MAE, R^2
- **Operational:** Latency < 100 ms, uptime $> 99.9\%$
- **Business Impact:** Fraud detection rate, ETA accuracy, reduced cancellations



Explanation: Used to understand trade-offs between precision and recall, crucial for fraud detection models.

---

5. Ethical Considerations

Michelangelo integrates governance into ML workflows:

- **Bias Detection:** Check for demographic parity in pricing models.
- **Fairness Constraints:** Exclude features correlated with protected attributes.

- **Privacy Compliance:** Adhere to GDPR, CCPA; no PII in feature store without explicit approval.
- **Transparency:** Model cards document intended use, limitations, and training data.

---

6. Post-Deployment Monitoring

Uber maintains performance with:

1. **Data Drift Detection:** KS tests and Population Stability Index for distribution changes.
2. **Alerting:** KPIs outside thresholds trigger Slack/PagerDuty alerts.
3. **Automated Retraining:** Nightly/weekly retraining when drift or performance decay is detected.

Explanation: Visualizes how Michelangelo detects and responds to shifts in input data distributions.

---

7. Tools & Technologies

- **Data Ingestion:** Apache Kafka, Hadoop, Hive
- **Processing & Training:** Apache Spark, Spark MLlib, XGBoost, TensorFlow, PyTorch
- **Feature Storage:** Palette Feature Store (offline via Hive, online via Cassandra)
- **Deployment:** Docker containers, Kubernetes, uDeploy
- **Monitoring:** Grafana dashboards, drift detection services
- **Experimentation:** A/B testing platform, shadow deployment framework

---

Conclusion

This case study illustrates how large-scale ML deployments benefit from unified infrastructure, reproducible workflows, and proactive monitoring. These principles are directly aligned with the course's best practices for lifecycle management, ethical safeguards, and continuous improvement in AI systems. Uber's Michelangelo demonstrates that success in ML is not solely dependent on accuracy metrics, but also on building systems that are scalable, maintainable, and ethically responsible. The integration of data quality checks, fairness assessments, and privacy controls ensures trust and compliance, while the platform's modular design enables rapid innovation and adaptation in dynamic environments.

AI Tool Usage Statement
Some sections of this report were developed with assistance from AI tools (ChatGPT) to aid in drafting, structuring, and refining the content. All material was critically reviewed, fact-checked, and edited by the author to ensure accuracy, relevance, and compliance with academic standards.

References

Uber Engineering. (2017, September 5). *Michelangelo: Machine learning platform*. Uber Engineering Blog. Retrieved August 9, 2025, from https://www.uber.com/en-US/blog/michelangelo-machine-learning-platform/

Uber Engineering. (2018, March 28). *Scaling machine learning at Uber with Michelangelo*. Uber Engineering Blog. Retrieved August 9, 2025, from https://eng.uber.com/scaling-michelangelo/

Uber Engineering. (2019, July 15). *Improving real-time predictions with Michelangelo*. Uber Engineering Blog. Retrieved August 9, 2025, from https://eng.uber.com/michelangelo-real-time/

Uber Engineering. (2020, January 22). *Monitoring and managing ML models in production*. Uber Engineering Blog. Retrieved August 9, 2025, from https://eng.uber.com/ml-model-monitoring/