Fraud Detection and Prevention

Use Case: Stripe Radar

Question: How would you design a fraud detection system?

Detailed Explanation:

Purpose: To detect and prevent fraudulent transactions in real-time with minimal false positives.

Data Sources:

- **Transaction Data:** Amount, merchant, customer details, payment method.
- **Historical Fraud Data:** Previously flagged fraudulent transactions.
- **Contextual Data:** Geolocation, device information, IP address.
- **Behavioral Data:** User login times, frequency of transactions.

Feature Engineering:

- **Transaction Characteristics:** Transaction amount, frequency, merchant type.
- **User Behavior Patterns:** Login times, changes in IP address or device, spending habits.
- **Device and Geolocation Information:** Device fingerprinting, geolocation consistency.
- **Historical Context:** Comparison with previous transactions, historical fraud patterns.

Model Choice:

- **Initial Model:** Logistic Regression for its interpretability and simplicity in identifying basic patterns.
- **Advanced Models:**
  - **Deep Neural Networks (DNNs):** For capturing complex relationships and patterns in high-dimensional data.
  - **Gradient Boosting Machines (GBMs):** Such as XGBoost for robust handling of various types of data and feature importance.
  - **Anomaly Detection:** Using techniques like Isolation Forests or One-Class SVM for detecting outliers.
  - **Ensemble Methods:** Combining the strengths of multiple models (e.g., stacking, bagging).

System Design:

Data Pipeline:

- **Data Ingestion:** Real-time ingestion using Apache Kafka or similar streaming platforms.

- **Preprocessing:** Cleaning and normalizing data, handling missing values, and generating features.
- **Feature Store:** Centralized storage for features using tools like Feast to ensure consistency across models.

Model Training:

- **Training Infrastructure:** Use distributed computing frameworks like Apache Spark or TensorFlow on clusters.
- **Cross-Validation:** Stratified K-fold cross-validation to ensure robust model performance.
- **Hyperparameter Tuning:** Using grid search or Bayesian optimization (e.g., Hyperopt).
- **Class Imbalance Handling:** Techniques like SMOTE (Synthetic Minority Over-sampling Technique) or weighted loss functions.

Real-Time Inference:

- **Deployment:** Models deployed as RESTful microservices using frameworks like Flask or FastAPI.
- **Inference Engine:** Real-time inference with latency under 100 milliseconds, possibly using an inference-optimized environment like TensorFlow Serving or NVIDIA Triton Inference Server.

Continuous Learning:

- **Feedback Loop:** Automated retraining pipelines using CI/CD tools (e.g., Jenkins) to incorporate new data and retrain models.
- **Drift Detection:** Monitoring data drift and model performance degradation using tools like Evidently AI.

Scalability:

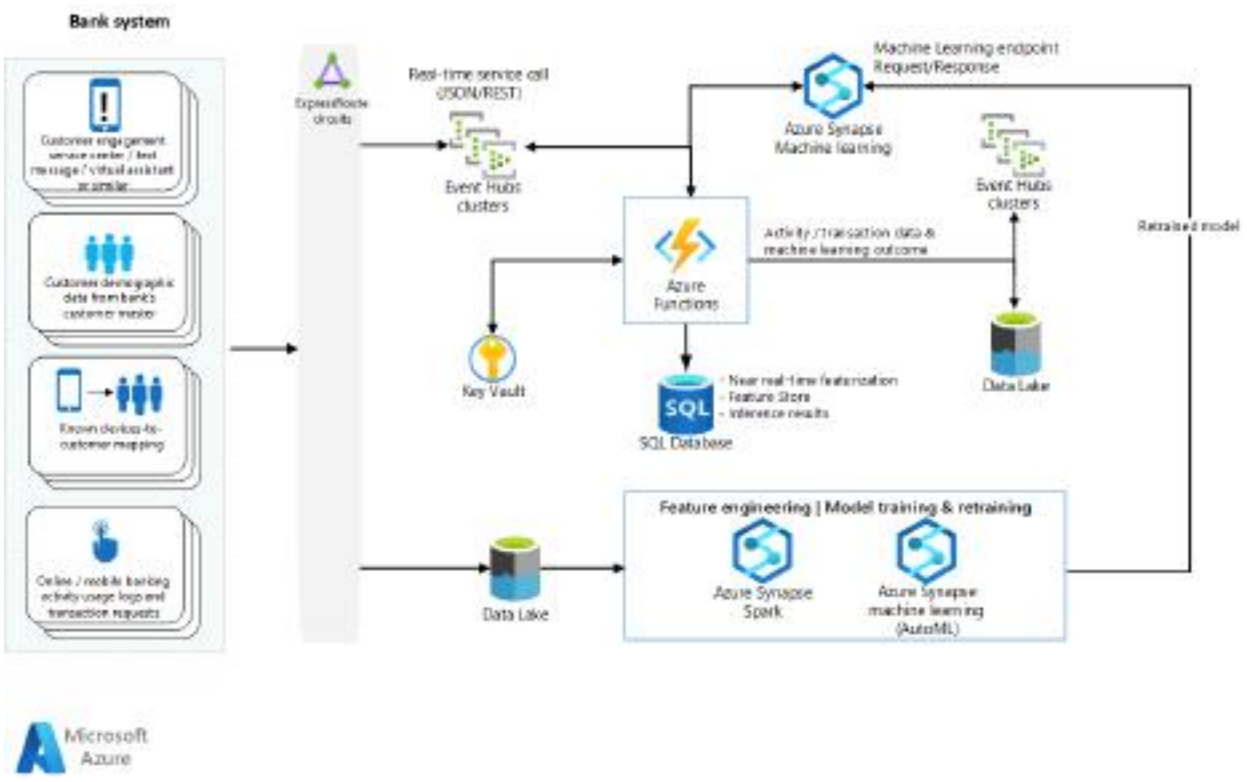
- **Infrastructure:** Use cloud services (e.g., AWS EC2, Lambda, S3) for scalability and reliability.
- **Distributed Processing:** Implement distributed training and serving using Kubernetes for orchestration.

Monitoring and Alerting:

- **Monitoring:** Track model performance metrics (e.g., precision, recall, F1 score) using monitoring tools like Prometheus and Grafana.
- **Alerting:** Set up alerts for anomalies in transaction patterns or model performance drops using PagerDuty or similar tools.

Evaluation Metrics:

- **Precision, Recall, F1 Score:** To balance false positives and false negatives.
- **ROC-AUC:** To measure the trade-off between true positive rate and false positive rate.
- **Latency:** Ensuring that real-time processing capabilities meet the sub-100 milliseconds requirement.
- **Model Interpretability:** Using SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) for compliance and trust.


Bank system
Customer engagement service center / text message / virtual assistant or similar
Customer demographic data from bank's customer master
Known devices-to-customer mapping
Online / mobile banking activity usage logs and transaction requests
ExpressRoute circuits
Real-time service call (JSON/REST)
Event Hubs clusters
Azure Synapse Machine learning
Machine Learning endpoint Request/Response
Azure Functions
Activity / transaction data & machine learning outcome
Event Hubs clusters
Retrained model
Key Vault
Near real-time featurization
Feature Store
Inference results
SQL Database
Data Lake
Feature engineering | Model training & retraining
Azure Synapse Spark
Azure Synapse machine learning (AutoML)
Data Lake
Microsoft Azure