Advance AI/ML Training – Telecom Data Analytics

MODULE 3 - Advanced Machine Learning in Telecom

Course Outline

- Mix. of theory & practical
  - **Module 1**: Foundation Recap and Telecom Analytics Basics
  - structured, semi-structured, and unstructured data in telecom with examples
  - Review SQL and Python basics for data collection and cleaning
  - Summarise ETL concepts and role of data pipelines in analytics
  - Visit Telecom predictive Models – Churn, telecom KPIs, and their role in decision making
  - Practical outcomes and use cases

- Mix. of theory & practical
  - **Module 2**: Advanced Data Engineering & Automation
  - Explain advanced ETL pipeline design and automation.
  - Role of data warehouses and data lakes in telecom
  - Introduce scalable data processing methods for large datasets.
  - Explain importance of data governance, privacy, and compliance
  - Practical outcomes and use cases

Course Outline

- Mix. of theory & practical
  - **Module 3**: Advanced Machine Learning in Telecom
  - Differentiate supervised, unsupervised, and semi-supervised learning in ML
  - Explain clustering techniques for customer segmentation and fraud detection.
  - Describe hyperparameter tuning (GridSearch, RandomSearch).
  - Explain model validation methods.
  - Discuss real-world ML challenges in telecom datasets
  - Practical outcomes and use cases

- Mix. of theory & practical
  - Module 4: Cloud Deployment, NLQ & Prescriptive Analytics
  - Explain deployment of ML models on cloud platforms
  - Describe Natural Language Query (NLQ) and its use in BI dashboards
  - Explain Natural Language Generation (NLG) for automated reporting.
  - Discuss prescriptive analytics for churn reduction, fraud detection, and network optimisation
  - Explain importance of real-time analytics in telecom operations
  - Practical outcomes and use cases

Your Learning

01

Types of Learning in ML

Understanding supervised, unsupervised, and semi-supervised approaches

02

Clustering Techniques

Segmenting customers and detecting anomalies

03

Regression & Classification

Predicting bills and customer churn

04

Hyperparameter Tuning

Optimizing model performance systematically

05

Model Validation

Evaluating and interpreting results

06

Real-World Challenges

Navigating telecom data complexities

Session 1: Types of Learning in ML

Understanding how machines learn from data

Three Ways Machines Learn

Supervised Learning

The model learns from **labeled data** where we already know the answer. It's like learning with a teacher who provides correct answers.

Telecom Example: Predict whether a customer will churn (Yes/No) based on their usage history and complaints.
Inputs are FEATURES and Outputs are LABELS

Use when:

- You have labeled training data
- You need to predict specific outcomes
- Clear input-output relationships exist

Best for: Churn prediction, bill forecasting ARPU , Network performance

Unsupervised Learning

The model finds patterns in **unlabeled data** without being told what to look for. It discovers hidden structures on its own.

Telecom Example: Group customers by data usage patterns without pre-defined categories—let the algorithm discover natural segments.

Use when:

- You're exploring unknown patterns
- Labels are unavailable or costly
- You want data-driven segmentation

Best for: Customer segmentation, anomaly detection

Semi-supervised Learning

Combines both approaches: uses a **small amount of labeled data** plus lots of unlabeled data. Perfect when labeling is expensive.

Telecom Example: Use 1,000 tagged churn records to help classify millions of unknown customers.

Use when:

- Limited labeled data exists
- Labeling is time-consuming
- You have abundant raw data

Best for: Scaling predictions, cost-effective modeling

Session 2: Clustering for Segmentation

Discovering patterns through unsupervised learning

Understanding Similarity and Distance

In clustering or segmentation, we don’t have labels (like “customer churned = yes/no”).

Instead, the model tries to find patterns or groups based on how close data points are to each other.

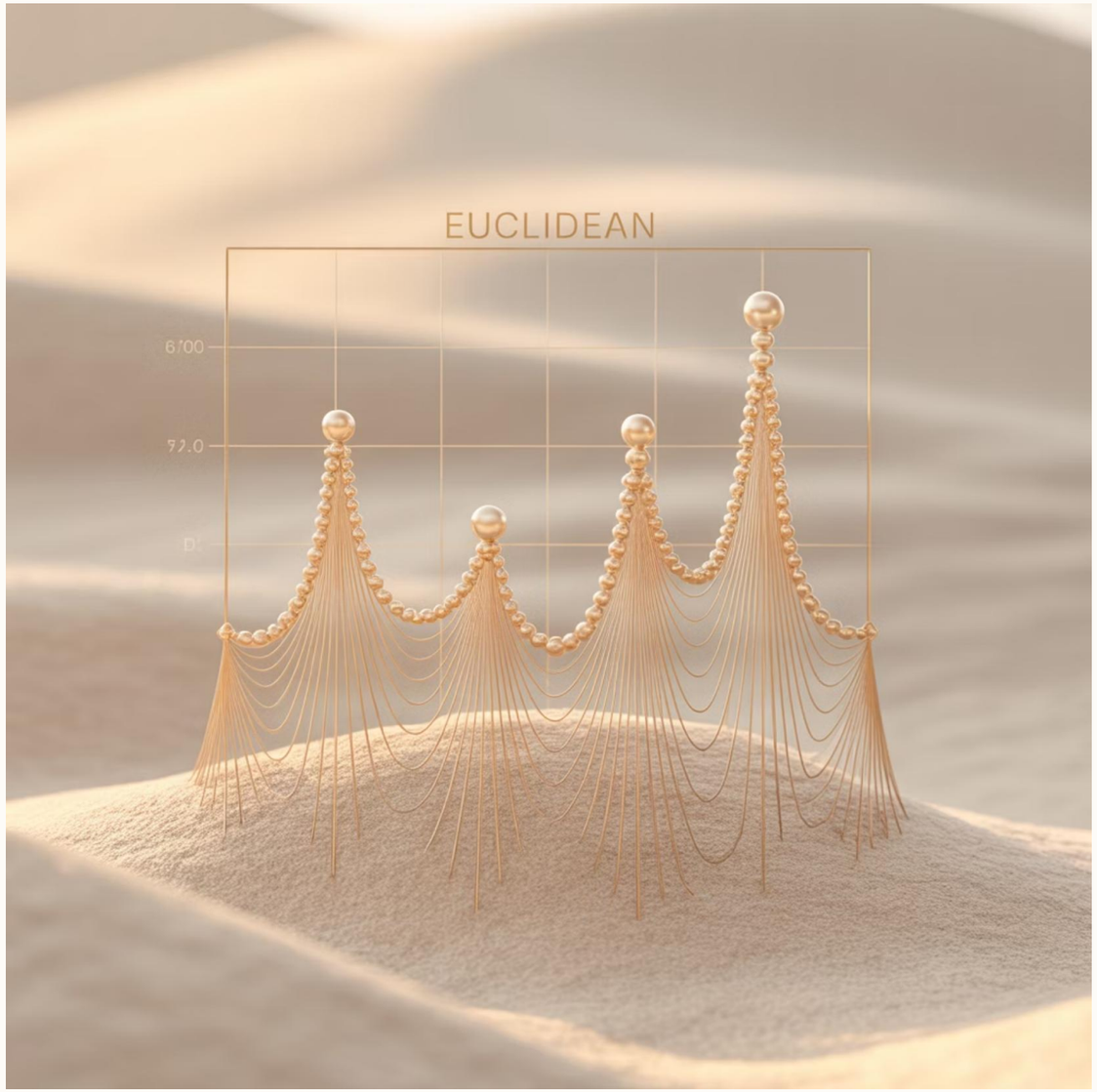
Imagine you have customers represented as points on a chart:

- X-axis = Data Usage (GB)
- Y-axis = Call Minutes

Customers who behave similarly (e.g., both use ~10 GB and make ~400 call minutes) should fall closer together on this graph. Those who behave differently (e.g., 1 GB and 50 minutes) will fall farther apart.So “**similarity**” means “closeness,” and the opposite is “distance.”

And this is where distance metrics come in

Euclidean Distance is the most common metric—it's literally the straight-line distance between two points in space



Two Powerful Clustering Methods

K-Means Clustering

Partitions data into K distinct, non-overlapping clusters. Fast and efficient for large datasets.
You decide how many clusters (K) you want before starting — for example, K=3 for three groups of customers: light, medium, and heavy users.

How it works: Randomly places K "centroids," assigns each point to the nearest centroid, then moves centroids to cluster centers. Repeats until stable.

Best for: Customer segmentation with known target groups

It’s one of the fastest and most popular algorithms for large datasets.

Hierarchical Clustering

Builds a tree of clusters, showing relationships at different scales. No need to specify K upfront.

Imagine analyzing network usage across cities:

- Initially, every city is its own cluster.
- The algorithm merges nearby cities with similar usage patterns.
- You get regional clusters like “Metro users,” “Tier-2 city users,” and “Rural users.”

How it works: Starts with each point as its own cluster, then progressively merges the closest pairs until one cluster remains.
Best for: Exploring data structure and natural groupings, you don’t Know the groupings.

It’s Slower (good for small to medium datasets) for Exploratory analysis

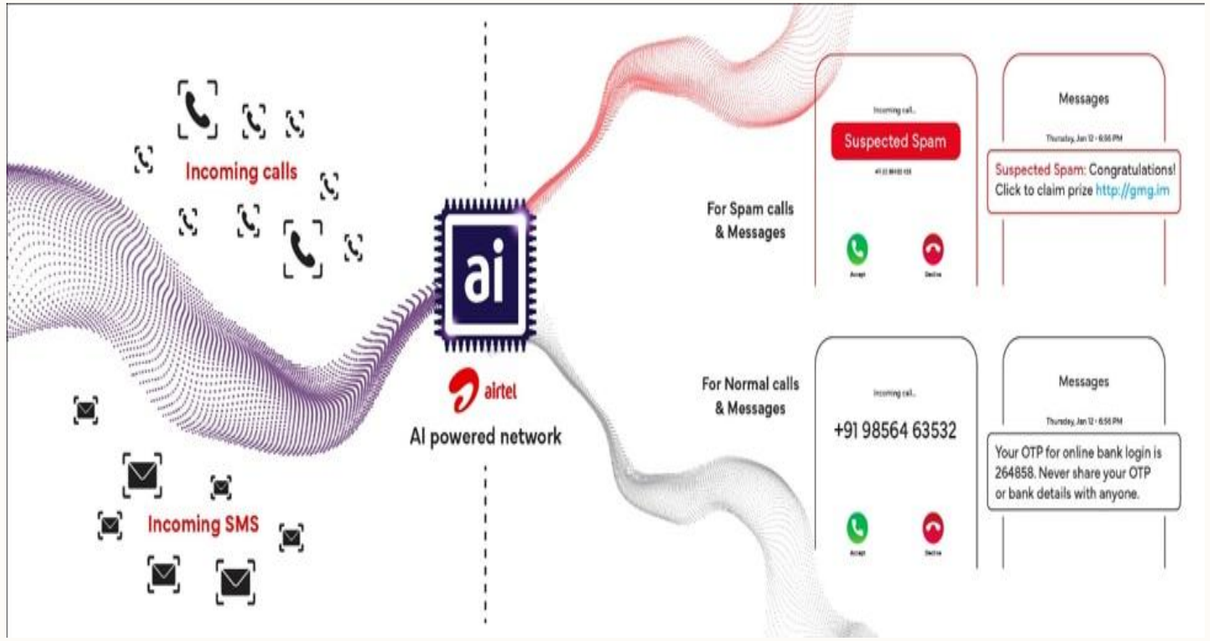
Hands-On: K-Means Customer Segmentation

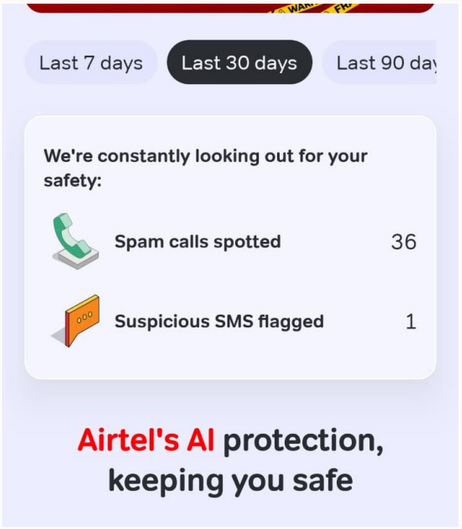
```
from sklearn.cluster import KMeansimport matplotlib.pyplot as pltimport seaborn as snsimport pandas as pddf =
pd.read_csv("telecom_usage.csv")# Cluster customers by data usage and call minuteskmeans = KMeans(n_clusters=3,
random_state=0).fit(df[['data_gb','call_mins']])df['cluster'] = kmeans.labels_# Visualize the three customer
segmentssns.scatterplot(x='data_gb', y='call_mins', hue='cluster', data=df, palette='viridis')plt.xlabel('Data Usage
(GB)')plt.ylabel('Call Minutes')plt.title('Customer Segments by Usage Behavior')plt.show()
```

Refer - MOP 4 - Exercise 10 – ML – K-means

What to observe: You'll see three distinct groups emerge—perhaps light users, moderate users, and heavy users. Each cluster represents a different customer persona with unique needs and value to the business.

Telecom Use Cases for Clustering – Spam detection





an AI-powered service for all its customers that uses network intelligence to detect and warn about spam calls and SMS. It is automatically enabled for free on Airtel users with a VoLTE (4G) enabled smartphone.

- The AI-powered system analyzes various parameters on the network in real-time, such as usage patterns, call duration, and use of suspicious URLs.
- It flags calls and SMS messages as "Suspected Spam" based on these patterns and displays the alert on your phone
- **Real-time alerts:** You get live warnings for incoming calls and SMS suspected to be spam.
- **Privacy-focused:** The AI checks sender behavior and red flags, such as frequent SIM changes or message blasts, without reading the message content.
- **Network-level filtering:** Known spam numbers are filtered out automatically at the network level.
- **Multiple language support:** Spam alert notifications are available in over 10 Indian languages.

Session 3: Regression & Classification

The two fundamental prediction tasks in telecom analytics

Predictive Analytics in Telecommunications

From Descriptive to Predictive

Most analytics begins with describing what happened: how many customers called support last month, what was average revenue per user. Predictive analytics asks a more powerful question: what will happen next?

Predictive analytics transforms data from being a rear-view mirror into a forward-looking radar.

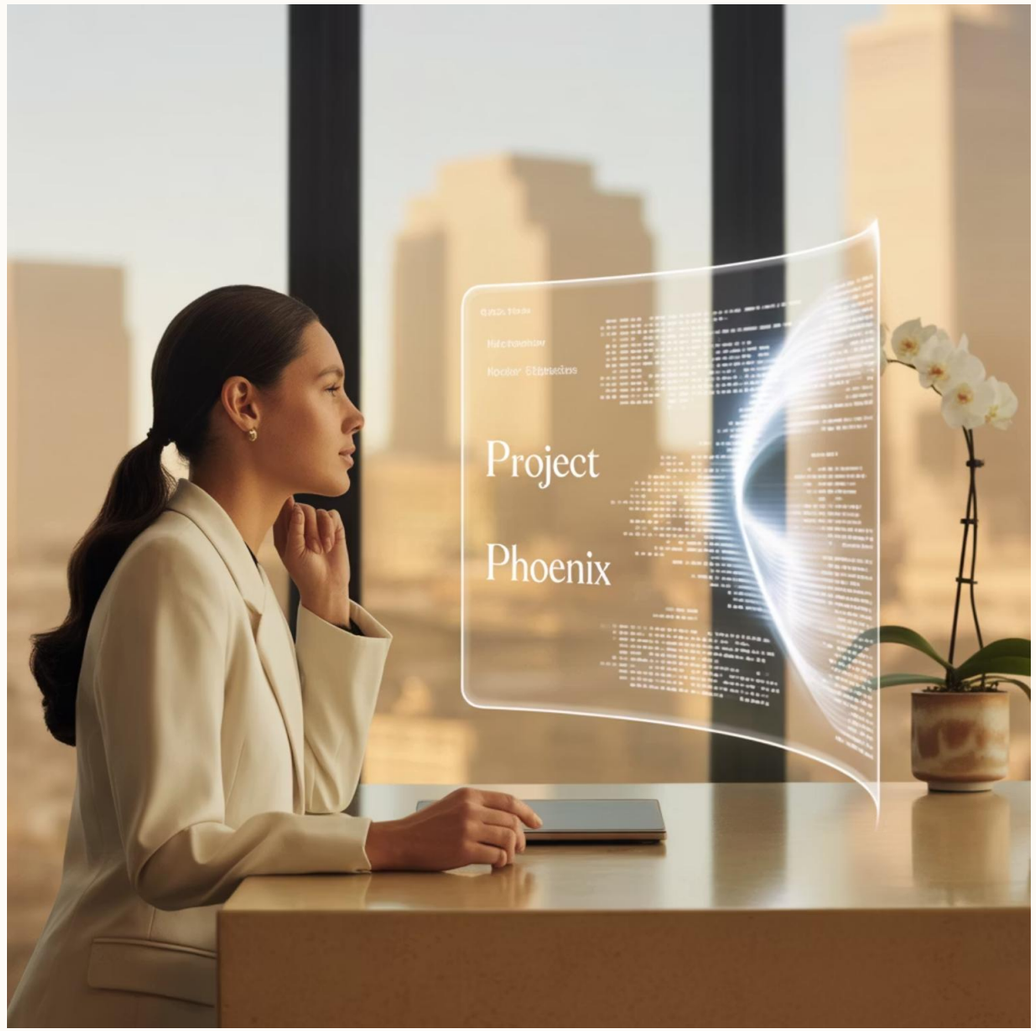
Telecommunications companies use predictive models to anticipate customer behavior, forecast network demand, and optimize operations. The two most common model types are regression (predicting numbers like revenue) and classification (predicting categories like "will churn" or "won't churn").

Regression Models

Predict continuous numeric outcomes like ARPU (Average Revenue Per User), monthly data consumption, or expected customer lifetime value

Classification Models

Predict categorical outcomes like whether a customer will churn, which service plan a prospect will choose, or if a network node will fail



Two Types of Predictions - fundamental in telecom analytics

Regression

Predicts a **continuous numerical value**. The output can be any number within a range.

Telecom Examples:

- Predicting ARPU (INR 405.67, INR 178.21, etc.)
- Estimating monthly data usage (15.3 GB)
- Network bandwidth required (125 Mbps)

Model examples:
Linear Regression
Decision Trees (for continuous outputs)

Think of it as: "How much?" or "How many?"

Classification

Predicts a **category or class label**. The output is discrete—one of several possible categories.

Telecom Examples:

- Predicting whether a customer will churn (Yes/No)
- Detecting network node failures (Fail/Not Fail)
- Fraud risk level (Low/Medium/High)

Model examples:
Logistic Regression
Decision Trees
Random Forest
Support Vector Machines

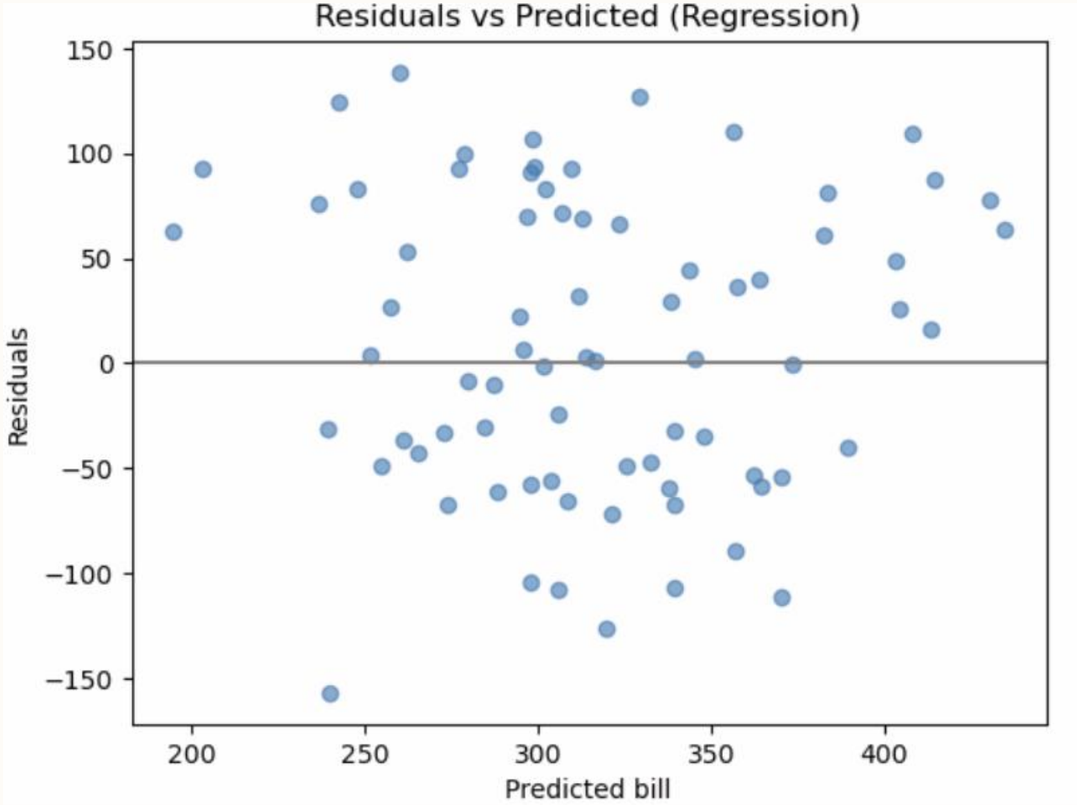
Think of it as: "Which category?" or "Is it?"

Building Both Models Side by Side

```
from sklearn.model_selection import train_test_splitfrom sklearn.linear_model import LinearRegression,
LogisticRegressionfrom sklearn.metrics import accuracy_score, mean_squared_errorimport numpy as np, pandas as pddf =
pd.read_csv("billing_churn.csv")X = df[['usage_gb','call_mins','complaints']]y_reg = df['monthly_bill']y_clf = df['churn']#
REGRESSION: Predict monthly billX_train, X_test, y_train, y_test = train_test_split(X, y_reg, test_size=0.3)reg =
LinearRegression().fit(X_train, y_train)print("RMSE:", np.sqrt(mean_squared_error(y_test, reg.predict(X_test))))#
CLASSIFICATION: Predict churnX_train, X_test, y_train, y_test = train_test_split(X, y_clf, test_size=0.3)clf =
LogisticRegression().fit(X_train, y_train)print("Accuracy:", accuracy_score(y_test, clf.predict(X_test)))
```

Refer - MOP 4 - Exercise 11 – ML – models

- Features: ['usage_gb', 'call_mins', 'complaints'] Coefficients: {'usage_gb': np.float64(6.804), 'call_mins': np.float64(0.226), 'complaints': np.float64(34.865)}
- Intercept: 150.35
- RMSE: 71.07



Metric / Indicator	Interpretation	Verdict
Coefficients	Direction & magnitude make domain sense	Good
Intercept	Logical baseline value (~₹135)	Good
RMSE (71.07)	~25–30% relative error	Acceptable, could be improved
Residual pattern	Random & balanced	Good model behavior

Session 5: Model Validation & Evaluation

Measuring what matters and interpreting your results

Train/Test Split and Cross-Validation

Train/Test Split

Divide your data into two sets: one for training (typically 70-80%) and one for testing (20-30%). Train on the first set, evaluate on the second to estimate real-world performance.

Why It Matters

Without proper validation, you can't trust your model's performance. Cross-validation helps detect overfitting and gives you confidence that your model will work on new, unseen customers.

1

2

3

K-Fold Cross-Validation

Split data into K equal parts. Train K times, each time using a different fold as the test set and the rest as training. Average the results for a more robust performance estimate.

Understanding Classification Metrics

Accuracy

Formula: (TP + TN) / (All predictions)

The percentage of predictions that were correct. Simple and intuitive, but misleading with imbalanced data. If 95% of customers don't churn, predicting "no churn" for everyone gives 95% accuracy!

Precision

Formula: TP / (TP + FP)

Of all the customers you predicted would churn, what percentage actually did? High precision means fewer false alarms—you won't waste money offering discounts to loyal customers.

Recall

Formula: TP / (TP + FN)

Of all the customers who actually churned, what percentage did you catch? High recall means you're identifying most at-risk customers—critical when losing customers is expensive.

F1 Score

Formula: 2 × (Precision × Recall) / (Precision + Recall)

The harmonic mean of precision and recall. Use this when you need to balance both concerns—neither false positives nor false negatives are acceptable.

Hands-On: Confusion Matrix Analysis

```
from sklearn.metrics import confusion_matrix, classification_report, ConfusionMatrixDisplayy_pred = clf.predict(X_test)# Print detailed metricsprint(classification_report(y_test, y_pred, target_names=['No Churn', 'Churn']))# Visualize the confusion matrixConfusionMatrixDisplay(confusion_matrix(y_test, y_pred), display_labels=['No Churn', 'Churn']).plot()plt.title('Churn Prediction Confusion Matrix')plt.show()
```

Refer - MOP 2 - Exercise 4 – Churn Prediction

Interpreting your confusion matrix:

- **True Positives (bottom-right):** Churners you correctly identified—great!
- **True Negatives (top-left):** Loyal customers you correctly identified—also great!
- **False Positives (top-right):** Loyal customers you thought would churn—wasted discount budget
- **False Negatives (bottom-left):** Churners you missed—lost revenue

Session 6: Real-World Challenges

Navigating the messy reality of telecom data

The 3 Vs of Telecom Data

Volume

Telecom companies generate **terabytes of data daily** from call detail records (CDRs), network logs, and customer interactions. A single carrier might track billions of events per day.

Challenge: Standard tools can't handle this scale. You'll need distributed systems like Spark or cloud platforms.

Velocity

Data arrives in **real-time streams**—calls connecting, texts sending, data packets flowing. Fraud detection must happen in seconds, not hours.

Challenge: Batch processing isn't enough. You need streaming analytics and models that update continuously.

Variety

Data comes in **many formats**: structured CDRs, unstructured call center transcripts, semi-structured network logs, location data, app usage, and more.

Challenge: Integration requires sophisticated ETL pipelines and feature engineering across diverse sources.

Additional Real-World Considerations

Data Quality Issues

Missing records: Dropped calls, logging failures, system outages

Duplicates: Same event logged multiple times across systems

Noise: Test calls, internal traffic, bot activity

Solution: Robust data cleaning pipelines, anomaly detection, and validation checks before modeling

Ethics & Privacy

Anonymization: Protecting customer identities in datasets

Bias: Ensuring models don't discriminate by demographics

Consent: Using data only for disclosed purposes

Solution: Privacy-preserving techniques, fairness audits, and regulatory compliance (GDPR, CCPA)

Model Lifecycle Management

Retraining: Customer behavior evolves—models decay over time

Monitoring drift: Detecting when performance degrades in production

Version control: Tracking model iterations and reproducibility

Solution: MLOps practices, automated retraining pipelines, and performance dashboards

Discussion prompt: "If your churn model misclassifies 5% of users, what's the real cost to the company?" Consider both false positives (wasted retention offers) and false negatives (lost customers). With 1 million subscribers, 5% is 50,000 people!

Thank YOU

By ISHAN MARWAH - +91 97178 56777 - Ishan Marwah | LinkedIn