15-Day Industry-Grade Data Science Project: Telecom Customer Churn Prediction

Project Overview

Project Title: Telecom Customer Churn Prediction for Revenue Optimization
Industry: Telecommunications
Duration: 15 Days
Dataset Size: 25+ MB
Project Type: Classification Problem - Customer Churn Prediction
Business Impact: High - Customer retention is 5-10x more cost-effective than acquisition

Problem Statement

The telecommunications industry faces an average annual churn rate of 15-25%, significantly impacting revenue and profitability. With customer acquisition costs being 5-10 times higher than retention costs, telecom companies need predictive models to identify at-risk customers before they churn. This project aims to develop a machine learning model that can predict customer churn with high accuracy using historical telecom data from India, enabling proactive customer retention strategies.

Dataset Specifications

Primary Dataset: TRAI Telecom Subscription Data

- **Source**: Telecom Regulatory Authority of India (Official Government Data)
- **Dataset**: Will be shared on the Portal
- **Format**: CSV
- **Size**: 25+ MB (70,728+ records)
- **Time Span**: 2009-2024 (15 years of comprehensive data)
- **Update Frequency**: Monthly
- **Granularity**: State, Operator, and Technology level data

Data Fields Description

- **year**: Numeric - Year of data collection
- **month**: Text - Month of measurement

- **circle**: Text - Telecom circle/state (38 unique values)
- **type_of_connection**: Text - Wireless/Wireline connection type
- **service_provider**: Text - Telecom operator (42 unique providers)
- **value**: Numeric - Number of subscribers
- **unit**: Text - Measurement unit
- **technology**: Text - Network technology (2G/3G/4G/5G)

Secondary Dataset: MySpeed TRAI Portal Data

- **URL**: https://data.gov.in/resource/month-wise-all-india-crowdsourced-mobile-data-speed-measurement
- **Purpose**: Network quality metrics for enhanced feature engineering
- **Fields**: Signal strength, data speeds, network performance indicators

15-Day Project Timeline

Phase 1: Project Setup & Data Acquisition (Days 1-3)

Day 1: Environment Setup & Data Collection Morning Session (4 hours) 1. **Environment Configuration** `bash # Required Python packages installation pip install pandas numpy scikit-learn matplotlib seaborn pip install xgboost lightgbm optuna shap lime pip install jupyter plotly dash streamlit`

2. **Data Download Process**
   - Download TRAI datasets from official government sources
   - Verify data integrity and completeness
   - Set up data directory structure

Afternoon Session (4 hours) 3. **Initial Data Exploration** "'python import pandas as pd import numpy as np

Load primary dataset df = pd.read_csv('trai_telecom_data.csv')

Basic data assessment print(f"Dataset shape: {df.shape}") print(f"Missing values: {df.isnull().sum().sum()}") print(f"Date range: {df['year'].min()} - {df['year'].max()}") "'

4. **Project Structure Setup**
   - Initialize Git repository
   - Create folder structure (data/, notebooks/, src/, models/, reports/)
   - Set up documentation framework

Day 2: Data Integration & Cleaning Morning Session (4 hours) 1. **Data Integration Pipeline** "'python def clean_telecom_data(df): # Handle missing values strategically df['value'] = df['value'].fillna(method='ffill')

```
# Standardize categorical values
df['service_provider'] = df['service_provider'].str.upper().str.strip()
df['circle'] = df['circle'].str.title().str.strip()

# Convert data types
df['year'] = pd.to_numeric(df['year'], errors='coerce')
df['value'] = pd.to_numeric(df['value'], errors='coerce')

return df
```

2. **Data Quality Assessment**
- Identify and handle missing values
- Detect and treat outliers using business logic
- Standardize data formats and types

Afternoon Session (4 hours)
3. **Master Dataset Creation**
- Merge multiple data sources
- Create consistent time series format
- Validate data consistency across sources

4. **Data Dictionary Documentation**
- Document all features and their business meaning
- Create metadata for reproducibility
- Establish data lineage tracking

Day 3: Exploratory Data Analysis (EDA)
Morning Session (4 hours)
1. **Subscription Pattern Analysis**
```python
# Visualize subscription trends by operator
plt.figure(figsize=(15, 8))
for operator in df['service_provider'].unique()[:5]:
    operator_data = df[df['service_provider'] == operator]
    monthly_subs = operator_data.groupby('date')['value'].sum()
    plt.plot(monthly_subs.index, monthly_subs.values, label=operator)
plt.legend()
plt.title('Subscription Trends by Operator')
```

2. **Regional Analysis**
    - Analyze subscription patterns by telecom circles
    - Identify regional preferences and trends
    - Map competitive landscape by geography

Afternoon Session (4 hours) 3. **Temporal Trend Analysis** - Seasonal patterns identification - Technology adoption curves (2G→3G→4G→5G) - Market disruption events (Jio launch impact)

4. **Churn Pattern Identification**
    - Define churn indicators based on subscription drops
    - Identify early warning signals
    - Analyze churn triggers and patterns

Phase 2: Data Preprocessing & Feature Engineering (Days 4-6)

Day 4: Target Variable & Basic Preprocessing Morning Session (4 hours) 1. **Churn Target Definition** “‘python def create_churn_target(df): # Define churn as significant subscriber drop df_sorted = df.sort_values([‘service_provider’, ‘circle’, ‘date’])

```
# Calculate month-over-month change
df_sorted['subscriber_change'] = df_sorted.groupby(['service_provider', 'circle'])['value

# Define churn threshold (>20% subscriber loss)
churn_threshold = -0.20
df_sorted['churn_flag'] = (df_sorted['subscriber_change'] < churn_threshold).astype(int)

return df_sorted
```

```
2. **Categorical Variable Encoding**
- Label encoding for ordinal variables
- One-hot encoding for nominal variables
- Handle high cardinality categorical features

**Afternoon Session (4 hours)**
3. **Feature Scaling & Normalization**
- StandardScaler for numerical features
- Robust scaling for outlier-prone features
- Min-max scaling where appropriate

4. **Data Splitting Strategy**
- Time-based split (80% train, 10% validation, 10% test)
- Ensure no data leakage across time periods
- Stratified sampling for balanced representation

#### Day 5: Advanced Feature Engineering
**Morning Session (4 hours)**
1. **Time-Series Features**
```python
def create_temporal_features(df):
    # Lag features (previous months data)
    for lag in [1, 3, 6, 12]:
        df[f'subscribers_lag_{lag}'] = df.groupby(['service_provider', 'circle'])['value'].s

    # Rolling statistics
    for window in [3, 6, 12]:
        df[f'subscribers_ma_{window}'] = df.groupby(['service_provider', 'circle'])['value']

    # Growth rates
```
```

```
df['mom_growth'] = df.groupby(['service_provider', 'circle'])['value'].pct_change()
df['yoy_growth'] = df.groupby(['service_provider', 'circle'])['value'].pct_change(perioc

return df
```

2. **Market Dynamics Features**
   - Market share calculations
   - Competitive intensity metrics
   - Technology migration indicators

Afternoon Session (4 hours) 3. **Business Logic Features** "'python # Operator categorization major_operators = ['JIO', 'AIRTEL', 'VI', 'BSNL'] df['is_major_operator'] = df['service_provider'].isin(major_operators).astype(int)

Regional categorization metro_circles = ['Delhi', 'Mumbai', 'Chennai', 'Kolkata'] df['is_metro'] = df['circle'].isin(metro_circles).astype(int) "'

4. **Network Quality Integration**
   - Integrate MySpeed data for network quality metrics
   - Create service quality indicators
   - Customer experience proxy variables

Day 6: Feature Selection & Advanced Engineering Morning Session (4 hours) 1. **Statistical Feature Selection** - Correlation analysis and multicollinearity detection - Mutual information scores - Chi-square tests for categorical features

2. **Advanced Feature Creation**
   - Customer lifetime value indicators
   - Technology adoption lag features
   - Competitive pressure metrics

Afternoon Session (4 hours) 3. **Feature Importance Analysis** - Random Forest feature importance - Permutation importance testing - Recursive feature elimination

4. **Final Feature Set Preparation**
   - Feature engineering pipeline creation
   - Data validation and quality checks
   - Prepare final modeling dataset

Phase 3: Model Development & Training (Days 7-10)

Day 7: Baseline Model Development Morning Session (4 hours) 1. **Model Pipeline Setup** "'python from sklearn.pipeline import Pipeline from sklearn.preprocessing import StandardScaler from sklearn.linear_model import LogisticRegression

Create preprocessing pipeline preprocessor = Pipeline([('scaler', StandardScaler())])

Baseline model baseline_model = Pipeline([('preprocessor', preprocessor), ('classifier', LogisticRegression(random_state=42))]) "'

2. **Cross-Validation Framework**
    - Time series cross-validation setup
    - Performance metrics definition
    - Evaluation pipeline creation

Afternoon Session (4 hours) 3. **Baseline Model Training** - Train logistic regression baseline - Initial performance evaluation - Identify areas for improvement

4. **Performance Metrics Setup**
    - Business-relevant metrics definition
    - Cost-sensitive evaluation framework
    - ROI calculation methodology

Day 8: Tree-Based Models Morning Session (4 hours) 1. **Random Forest Implementation** "'python from sklearn.ensemble import RandomForestClassifier from sklearn.model_selection import GridSearchCV

Random Forest with hyperparameter tuning rf_params = { 'n_estimators': [100, 200, 300], 'max_depth': [10, 20, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4] }

rf_model = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=5, scoring='roc_auc') "'

2. **Hyperparameter Optimization**
    - Grid search implementation
    - Random search for efficiency
    - Bayesian optimization with Optuna

Afternoon Session (4 hours) 3. **Feature Importance Analysis** - Tree-based feature importance - SHAP values for interpretability - Business insight extraction

4. **Model Performance Comparison**
    - Compare against baseline
    - Analyze improvement sources
    - Document model characteristics

Day 9: Gradient Boosting & Ensemble Methods Morning Session (4 hours) 1. **XGBoost Implementation** "'python import xgboost as xgb from sklearn.model_selection import RandomizedSearchCV

XGBoost with advanced hyperparameter tuning xgb_params = { 'n_estimators': [100, 200, 500], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 5, 7], 'subsample': [0.8, 0.9, 1.0], 'colsample_bytree': [0.8, 0.9, 1.0] }

xgb_model = RandomizedSearchCV(xgb.XGBClassifier(random_state=42), xgb_params, n_iter=20, cv=5) “‘

2. **LightGBM Implementation**
   - Fast gradient boosting alternative
   - Handle categorical features natively
   - Optimize for speed and accuracy

Afternoon Session (4 hours) 3. **Class Imbalance Handling** - SMOTE for oversampling - Undersampling techniques - Cost-sensitive learning approaches

4. **Ensemble Methods**
   - Voting classifier implementation
   - Stacking with meta-learner
   - Blending multiple models

Day 10: Model Selection & Validation **Morning Session (4 hours)** 1. **Advanced Model Techniques** - Neural network implementation (if beneficial) - Time series specific models - Hybrid approaches

2. **Model Interpretability**

```python
import shap

# SHAP analysis for model interpretability
explainer = shap.TreeExplainer(best_model)
shap_values = explainer.shap_values(X_test)

# Generate interpretation plots
shap.summary_plot(shap_values, X_test)
```

Afternoon Session (4 hours) 3. **Final Model Selection** - Business metric optimization - Production readiness assessment - Model complexity vs performance trade-off

4. **Holdout Validation**
   - Final model validation on unseen data
   - Performance stability testing
   - Confidence interval calculation

Phase 4: Evaluation & Business Analysis (Days 11-12)

Day 11: Comprehensive Model Evaluation **Morning Session (4 hours)** 1. **Performance Metrics Analysis** “‘python from sklearn.metrics import classification_report, roc_auc_score, precision_recall_curve

Comprehensive evaluation def evaluate_model_performance(y_true, y_pred_proba): # ROC-AUC analysis roc_auc = roc_auc_score(y_true, y_pred_proba)

````
    # Precision-Recall analysis
    precision, recall, _ = precision_recall_curve(y_true, y_pred_proba)

    # Business metrics
    top_10_percent = int(len(y_pred_proba) * 0.1)
    top_10_indices = np.argsort(y_pred_proba)[-top_10_percent:]
    precision_at_10 = y_true.iloc[top_10_indices].mean()

    return {
        'roc_auc': roc_auc,
        'precision_at_10': precision_at_10
    }


2. **Business Impact Assessment**
- Cost-benefit analysis framework
- ROI calculation for retention campaigns
- Revenue protection estimates

**Afternoon Session (4 hours)**
3. **Model Bias & Fairness Evaluation**
- Demographic parity assessment
- Equal opportunity analysis
- Bias mitigation strategies

4. **Sensitivity Analysis**
- Model stability under different conditions
- Threshold optimization for business objectives
- Scenario planning and stress testing

#### Day 12: Customer Segmentation & Business Insights
**Morning Session (4 hours)**
1. **Customer Risk Segmentation**
```python
# Risk-based customer segmentation
def create_risk_segments(churn_probabilities):
    segments = pd.cut(churn_probabilities,
                     bins=[0, 0.3, 0.7, 1.0],
                     labels=['Low Risk', 'Medium Risk', 'High Risk'])
    return segments
````

2. **Churn Driver Analysis**
    - Primary churn factors identification
    - Regional and demographic patterns
    - Technology and service quality impacts

Afternoon Session (4 hours) 3. **Actionable Insights Generation** - Cus-
````

tomer retention strategy recommendations - Targeted intervention guidelines - Resource allocation optimization

4. **Business Recommendation Framework**
    - Proactive vs reactive strategies
    - Campaign targeting methodology
    - Success measurement framework

Phase 5: Deployment & Documentation (Days 13-15)

Day 13: Model Deployment Preparation Morning Session (4 hours)
1. **Model Serialization** "'python import joblib

Save trained model joblib.dump(best_model, 'models/churn_prediction_model.pkl')

Save preprocessing pipeline joblib.dump(preprocessor, 'models/preprocessing_pipeline.pkl')
"'

2. **Prediction Pipeline Creation**
    - Real-time prediction capability
    - Batch processing framework
    - Data validation and error handling

Afternoon Session (4 hours) 3. **API Development** "'python from flask import Flask, request, jsonify

app = Flask(**name**)

@app.route('/predict_churn', methods=['POST']) def predict_churn(): data = request.get_json() # Process data and return prediction prediction = model.predict_proba(processed_data) return jsonify({'churn_probability': prediction[0][1]}) "'

4. **Performance Monitoring Setup**
    - Model drift detection
    - Performance degradation alerts
    - Automated retraining triggers

Day 14: Dashboard & Reporting Systems Morning Session (4 hours)
1. **Business Dashboard Development** "'python import streamlit as st import plotly.express as px

Streamlit dashboard for business users st.title('Customer Churn Risk Dashboard')

Risk distribution visualization fig = px.histogram(df, x='churn_probability', title='Customer Churn Risk Distribution') st.plotly_chart(fig) "'

2. **Automated Reporting System**
    - Weekly churn risk reports
    - Executive summary generation

- KPI tracking and alerting

Afternoon Session (4 hours) 3. **Model Retraining Pipeline** - Automated data ingestion - Model performance monitoring - Scheduled retraining workflow

4. **Documentation & Code Review**
   - Technical documentation completion
   - Code quality assessment
   - Deployment checklist verification

Day 15: Final Presentation & Handover **Morning Session (4 hours)** 1. **Executive Presentation Preparation** - Business case summary - Model performance highlights - Implementation roadmap

2. **ROI Calculation & Business Case**

```python
# Business impact calculation
def calculate_business_impact(true_positives, false_positives,
                              retention_cost, customer_value):
    saved_revenue = true_positives * customer_value
    campaign_cost = (true_positives + false_positives) * retention_cost
    net_benefit = saved_revenue - campaign_cost
    roi = (net_benefit / campaign_cost) * 100
    return roi
```

Afternoon Session (4 hours) 3. **Future Improvements & Recommendations** - Model enhancement opportunities - Additional data sources integration - Advanced analytics possibilities

4. **Project Handover Documentation**
   - Complete technical documentation
   - Maintenance and support guidelines
   - Knowledge transfer to operations team

Key Performance Indicators (KPIs)

Primary Model Metrics

- **Accuracy**: Target >85%
- **Precision**: Target >80% (minimize false positives)
- **Recall**: Target >75% (capture actual churners)
- **F1-Score**: Target >78% (balanced performance)
- **ROC-AUC**: Target >0.85 (discrimination ability)

Business Metrics

- **Churn Detection Rate**: % of actual churners identified
- **False Positive Rate**: <20% (avoid unnecessary retention costs)
- **Precision at Top 10%**: Accuracy in highest risk segment

- **Lift at Top Decile**: Improvement over random selection
- **Cost-Benefit Ratio**: ROI of retention campaigns

Expected Business Impact

Revenue Protection

- Potential to save 15-25% of at-risk revenue
- Improve customer lifetime value by 8-12%
- Reduce customer acquisition costs through better retention

Operational Efficiency

- Reduce blanket marketing costs by 40-60%
- Enable targeted, data-driven retention campaigns
- Automate risk scoring and customer segmentation

Competitive Advantage

- Proactive customer management capability
- Data-driven decision making framework
- Improved market share retention

Technical Requirements

Software & Libraries

```
# Core data science stack
pip install pandas>=1.3.0
pip install numpy>=1.21.0
pip install scikit-learn>=1.0.0
pip install matplotlib>=3.4.0
pip install seaborn>=0.11.0

# Advanced ML libraries
pip install xgboost>=1.5.0
pip install lightgbm>=3.3.0
pip install optuna>=2.10.0

# Model interpretability
pip install shap>=0.40.0
pip install lime>=0.2.0

# Deployment & APIs
pip install flask>=2.0.0
pip install streamlit>=1.2.0
pip install fastapi>=0.70.0
```

```
# Development tools
pip install jupyter>=1.0.0
pip install git+https://github.com/python-git/python-git.git
```

Hardware Requirements

- **Memory**: Minimum 8GB RAM (16GB recommended)
- **Storage**: 5GB free space for data and models
- **Processing**: Multi-core CPU (GPU optional for neural networks)

Project Deliverables

Technical Deliverables

1. Cleaned and processed dataset (CSV format)
2. Feature engineering pipeline (Python scripts)
3. Trained machine learning models (pickle files)
4. Model evaluation report with performance metrics
5. Prediction API (Flask/FastAPI implementation)
6. Jupyter notebooks with complete analysis
7. Automated retraining pipeline
8. Technical documentation and code comments

Business Deliverables

1. Executive summary with business insights
2. Customer churn risk dashboard
3. Actionable retention strategy recommendations
4. ROI analysis and business case
5. Implementation roadmap for production
6. Training materials for business users
7. Performance monitoring framework
8. Quarterly review and update process

Success Criteria

Technical Success

- Model achieves target performance metrics (>85% accuracy, >0.85 ROC-AUC)
- Reproducible and maintainable codebase
- Production-ready deployment pipeline
- Comprehensive technical documentation

Business Success

- Clear identification of top churn risk factors

- Actionable insights for retention strategies
- Positive ROI projection for implementation
- Stakeholder approval for production deployment

Learning Outcomes

By completing this project, participants will gain:

1. **End-to-End Data Science Workflow**: Experience with complete project lifecycle
2. **Real-World Data Handling**: Skills with messy, large-scale datasets
3. **Advanced Feature Engineering**: Time-series and business logic features
4. **Model Selection & Tuning**: Systematic approach to algorithm selection
5. **Business Impact Analysis**: Translation of technical results to business value
6. **Production Deployment**: Model serving and monitoring capabilities

Additional Resources

Dataset Access

- **Primary Source**: TRAI Subscription Data
- **Secondary Source**: MySpeed TRAI Portal
- **Backup Sources**: TRAI official reports and press releases

Reference Materials

- TRAI Annual Reports for industry context
- Telecom industry research papers
- Customer churn prediction literature
- Indian telecommunications market analysis

Support & Community

- GitHub repository for code sharing
- Documentation wiki for knowledge base
- Regular check-ins with domain experts
- Peer review and feedback sessions

This comprehensive 15-day project provides industry-grade experience in data science while solving a real business problem with significant impact potential. The structured approach ensures both technical skill development and practical business application.