Senior Data Scientist Code Challenge

E-commerce Returns Prediction

Duration: 90 minutes
Tools: Python, any ML libraries, full internet/documentation access
Submission: Jupyter Notebook + Executive Summary

Business Context

You work for **ShopFlow**, an e-commerce platform processing 100,000+ orders monthly.

The Problem:

- 22% of products are returned
- Each return costs **$18** (shipping, restocking, processing)
- Monthly cost: ~$400,000
- No systematic prediction or intervention

The Opportunity:

- Predict which orders will likely be returned
- Apply targeted interventions: improved product info, proactive support
- Intervention cost: **$3 per order**
- Intervention effectiveness: Reduces return probability by **35%**

Your Challenge: Build a model that maximizes ROI by identifying the right customers for intervention.

What You'll Receive

Datasets:

- `ecommerce_returns_train.csv` (8,000 orders)
- `ecommerce_returns_test.csv` (2,000 orders)
- `baseline_model.py` (simple logistic regression)

Folder with all the material

Features:

Feature	Description	Type
order_id	Unique identifier	ID
customer_age	Age of customer	Numeric
customer_tenure_days	Days since first purchase	Numeric
product_category	Fashion, Electronics, Home_Decor	Categorical
product_price	Price in USD	Numeric
days_since_last_purchase	Days since last order	Numeric
previous_returns	Returns in last 6 months	Numeric
product_rating	Average rating (1-5)	Numeric
size_purchased	XS, S, M, L, XL, XXL	Categorical
discount_applied	0 or 1	Binary
is_return	**TARGET** (0=kept, 1=returned)	**Binary**

Your Tasks

Part 1: Baseline Evaluation (10 minutes)

What to do:

1. Load and run the provided baseline logistic regression
2. Evaluate performance on test set
3. Create comprehensive evaluation with:
    - Multiple metrics with justification
    - Confusion matrix with interpretation
    - Performance by product category
    - Model weakness identification

Deliverable:

- Code cells with evaluation
- Markdown explaining findings
- At least 3 metrics with reasoning for each

Key Questions:

- What are strengths and weaknesses?
- Where does it fail most?
- Is accuracy the right metric? Why/why not?

Part 2: Business-Aligned Metrics (20 minutes)

Financial Reality:

- Returns cost: **$18**
- Interventions cost: **$3** (reduce return probability by 35%)

Cost-Benefit Matrix:

Prediction	Reality	Action	Financial Impact
Will return	Returns	Intervention applied	Save $15 ($18 - $3)
Will return	Doesn't return	Intervention applied	Lose $3 (wasted)
Won't return	Returns	No action	Lose $18 (missed)
Won't return	Doesn't return	No action	$0 (correct)

What to do:

1. Define "success" in business terms
2. Recommend 2-3 metrics aligned with business goals
3. Analyze false positive vs. false negative trade-offs
4. Calculate financial impact of predictions
5. Determine optimal threshold

Deliverable:

- Metric recommendations with justification
- Cost-benefit analysis
- Threshold selection rationale
- "Good enough to deploy" criteria

Critical Question: What's the optimal balance between catching returns (recall) and avoiding wasted interventions (precision)?

Part 3: Model Improvement (20 minutes)

Goal: Improve upon the baseline using one or more approaches:

- Feature engineering
- Hyperparameter tuning
- Different algorithms

Requirements:

1. Document hypothesis for each improvement
2. Implement the improvement
3. Validate no data leakage
4. Quantify performance gain vs. baseline
5. Demonstrate it's not overfitting

Deliverable:

- Code implementing improvements
- Markdown documenting:
    - What you tried and WHY
    - What worked and what didn't
    - Quantified improvements with comparisons
    - Validation that improvements generalize

We're evaluating:

- Clear reasoning for decisions
- Proper validation methodology
- Quantitative comparisons
- Systematic experimentation approach

Part 4: Deployment Planning (10 minutes)

Task 1: Production Monitoring Plan

Document what you'd monitor:

- Which metrics to track
- How to detect model degradation
- Specific alerts to configure
- When to retrain
- Rollback criteria

Task 2: Stakeholder Summary

Create one-page summary covering:

- Model performance in business terms
- Expected ROI
- Deployment risks and mitigations
- Success metrics to track post-launch

Deliverable:

- Deployment monitoring plan (markdown)
- Stakeholder summary (markdown)

Questions to address:

- What could degrade this model over time?
- How would you detect seasonal patterns?
- What A/B testing strategy?

- When to retrain?

Submission

Upload to a public Github repository you own the following content:

1. **Jupyter Notebook** (`lastname_firstname_challenge.ipynb`)
    - Clear sections with headers
    - Code cells with comments
    - Markdown cells explaining reasoning
    - All outputs visible (run all cells before submitting)
2. **Executive Summary** (`summary.md`)
    - 300-500 words
    - Approach overview
    - Key findings
    - Business impact estimate
    - Deployment recommendation
3. **Model File** (`model.pkl` or similar)
    - Your final trained model
4. **Optional:** Additional files if needed

Evaluation Rubric (100 points)

Technical Execution (35 points)

- **Model Evaluation** (15 pts): Multiple appropriate metrics, thorough baseline analysis, failure mode identification
- **Improvement Methodology** (12 pts): Valid approach, proper validation, no leakage
- **Code Quality** (8 pts): Clean, readable, well-organized

Reasoning & Business Thinking (40 points)

- **Metric Selection** (15 pts): Clear business connection, justified choices
- **Cost-Benefit Analysis** (12 pts): Thoughtful threshold selection, trade-off discussion
- **Communication** (13 pts): Clear articulation of rationale, well-structured summary

Deployment Readiness (25 points)

- **Monitoring Strategy** (12 pts): Specific metrics, degradation detection, systematic approach
- **Risk Assessment** (8 pts): Identified risks, mitigation strategies
- **Stakeholder Communication** (5 pts): Business-friendly summary, actionable insights

Passing Score: 70+

Scoring Interpretation

90-100 (Excellent):

- Deep understanding of evaluation metrics
- Strong business thinking with clear ROI connection
- Systematic improvement with rigorous validation
- Production-ready monitoring strategy
- Exceptional communication

80-89 (Strong):

- Solid evaluation with appropriate metrics
- Good business thinking with ROI connection
- Valid improvements with proper validation
- Reasonable deployment considerations
- Clear communication

70-79 (Acceptable):

- Basic evaluation with standard metrics
- Some business thinking
- Attempted improvements with basic validation
- Generic deployment considerations
- Adequate communication

Below 70 (Needs Improvement):

- Superficial evaluation (accuracy only)
- No business connection
- Improvements without validation
- Missing deployment strategy
- Unclear reasoning

Red Flags (Require Discussion)

If your submission has these, we'll discuss during follow-up:

🚩 Only accuracy as evaluation metric
🚩 No discussion of class imbalance
🚩 No cost-benefit analysis
🚩 Improvements without baseline comparison
🚩 No validation strategy
🚩 Can't explain feature choices
🚩 No overfitting prevention discussion
🚩 Vague deployment plan
🚩 No data drift consideration
🚩 Executive summary lacks business impact

Baseline Model Code

Python

```
"""
Baseline Model - Simple Logistic Regression
Use this as your starting point
"""

import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score,
classification_report
import joblib

# Load data
train = pd.read_csv('ecommerce_returns_train.csv')
test = pd.read_csv('ecommerce_returns_test.csv')

def preprocess(df):
    """Simple preprocessing pipeline"""
    df_processed = df.copy()

    # Encode categorical: product_category
    le_category = LabelEncoder()
    df_processed['product_category_encoded'] =
le_category.fit_transform(
        df_processed['product_category']
    )

    # Handle missing sizes (Fashion items only have sizes)
    if df_processed['size_purchased'].notna().any():
        most_common_size =
df_processed['size_purchased'].mode()[0]

df_processed['size_purchased'].fillna(most_common_size,
inplace=True)

        le_size = LabelEncoder()
        df_processed['size_encoded'] = le_size.fit_transform(
            df_processed['size_purchased']
        )

    # Feature selection
    feature_cols = [
```

```
        'customer_age', 'customer_tenure_days',
'product_category_encoded',
        'product_price', 'days_since_last_purchase',
'previous_returns',
        'product_rating', 'size_encoded', 'discount_applied'
    ]

    X = df_processed[feature_cols]
    y = df_processed['is_return']

    return X, y

# Prepare data
X_train, y_train = preprocess(train)
X_test, y_test = preprocess(test)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train baseline model
baseline_model = LogisticRegression(random_state=42,
max_iter=1000)
baseline_model.fit(X_train_scaled, y_train)

# Predictions
y_pred = baseline_model.predict(X_test_scaled)

# Basic evaluation
print("Baseline Model Performance")
print("=" * 50)
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Save artifacts
joblib.dump(baseline_model, 'baseline_model.pkl')
joblib.dump(scaler, 'scaler.pkl')

print("\n" + "=" * 50)
print("YOUR TASK: Evaluate thoroughly and improve this
baseline")
print("=" * 50)
```

FAQs

Q: Can I use deep learning?
A: Yes, but justify why and show it outperforms simpler approaches. With 8,000 samples, you need strong justification.

Q: How much feature engineering?
A: Focus on thoughtful, well-reasoned features over quantity. 2-3 justified features > 20 random ones.

Q: Can I use AutoML?
A: Yes, but explain what it's doing and validate results independently.

Q: What if I don't finish?
A: Submit what you have. We value depth over completion. Thorough Parts 1-2 > rushed all 4.

Q: Can I ask questions?
A: Clarifying business context: yes. Technical implementation help: no.

Example: Strong Part 2 Submission

Business-Aligned Metrics

1. Precision at threshold=0.42 (Primary)

Rationale: False positives waste $3, false negatives lose $18. Need high precision to avoid wasted interventions.

- Baseline (threshold=0.5): 0.68 precision, 0.52 recall
- Optimal (threshold=0.42): 0.72 precision, 0.58 recall

2. Expected Value per Customer (Business)

Formula: `EV = (TP_rate × $15) - (FP_rate × $3) - (FN_rate × $18)`

- Baseline EV: $2.40/customer
- Goal: Maximize ROI

3. Recall for High-Risk Segment (Secondary)

Fashion has a 30% return rate vs. 15% overall. Want higher recall for Fashion specifically.

Threshold Analysis Results:

Threshold	Precision	Recall	Expected Value	Monthly Savings
0.30	0.65	0.68	$1.80	$180K
0.42	0.72	0.58	$3.20	$320K

0.50	0.76	0.52	$2.40	$240K

Optimal: 0.42 - maximizes expected value

Deployment Criteria:

- ✅ EV > $2.00/customer
- ✅ Precision > 0.65
- ✅ Recall > 0.50
- ✅ Passes A/B test (95% confidence)

[Code showing analysis...]

Good luck! We're excited to see your approach.