Machine Learning Capstone Project: Multi-Modal Employee Performance Prediction System

Course: Applied Machine Learning
Integrative Project: Supervised Learning, Feature Engineering, and Ensemble Methods
Duration: 7 days
Total Points: 100

Project Overview

In this capstone assignment, you will build a comprehensive employee performance prediction system that combines insights from three different data modalities: structured quantitative metrics, behavioral patterns, and audio-based communication features. This project integrates all supervised learning techniques you have mastered throughout the course.

You will work with real-world HR data containing **5,000 employee records** across **three separate CSV files**, each providing a different perspective on employee performance. Your challenge is to extract maximum predictive power by intelligently combining these data sources.

The assignment is structured in **four main phases**:

1. **Phase 1**: Individual Model Development (one model per data source)
2. **Phase 2**: Feature Engineering and Data Fusion
3. **Phase 3**: Ensemble Model Building and Optimization
4. **Phase 4**: Comprehensive Analysis and Business Insights Report

Learning Objectives:

- Master multimodal data integration techniques
- Apply regression and classification algorithms to real HR analytics
- Implement ensemble methods for improved prediction accuracy
- Conduct feature importance analysis across different data types
- Communicate technical findings to business stakeholders

Dataset Description

You will work with three interconnected CSV files:

File 1: structured_data.csv (Quantitative Performance Metrics)

9 features + target variable

Features:

- employee_id – Unique identifier (1001-6000)

- tasks_completed – Number of tasks completed
- average_task_quality – Quality rating (1-10 scale)
- projects_led – Number of projects led
- client_satisfaction_score – Client feedback (0-100 scale)
- hours_worked – Total hours worked
- deadline_met_score – Deadline adherence (1-10 scale)
- innovation_score – Creativity rating (1-10 scale)
- efficiency_score – Work efficiency (1-10 scale)

Target: performance_rating – Classification: High, Medium, or Low

File 2: behavior_logs.csv (Behavioral Performance Indicators)

9 features + target variable

Features:

- employee_id – Unique identifier
- meetings_attended – Number of meetings
- collaboration_score – Teamwork rating (1-10 scale)
- punctuality_score – Timeliness rating (1-10 scale)
- training_hours_completed – Training time invested
- work_engagement_score – Engagement level (1-10 scale)
- peer_interaction_score – Colleague interaction quality (1-10 scale)
- initiative_score – Proactiveness rating (1-10 scale)
- task_followup_score – Task follow-through (1-10 scale)

Target: performance_rating – Classification: High, Medium, or Low

File 3: audio_features.csv (Communication Pattern Analysis)

9 features + target variable

Features:

- employee_id – Unique identifier
- speech_sentiment_score – Speech sentiment (-1 to 1 scale)
- speech_energy_level – Voice energy (1-10 scale)
- speech_clarity_score – Communication clarity (1-10 scale)
- tone_consistency_score – Tone stability (1-10 scale)

- speaking_speed – Words per minute (80-159 range)
- pause_frequency – Speech pause frequency (0-9 scale)
- pitch_variation – Voice pitch variation (1-10 scale)
- volume_stability_score – Volume consistency (1-10 scale)

Target: performance_rating – Classification: High, Medium, or Low

Important Notes:

- All three files contain the same 5,000 employees (matched by employee_id)
- Performance ratings are perfectly synchronized across all three files
- Class distribution: Balanced across High, Medium, and Low categories
- No missing values in the dataset
- All features are numerical (some categorical encoding may be needed for employee_id)

Phase 1: Individual Model Development (25 points)

Objective

Build and evaluate separate predictive models for each data source to establish baseline performance and understand which data modality is most predictive.

Task Requirements

1.1 Data Loading and Initial Exploration (5 points)

For EACH of the three CSV files:

- Load the dataset using pandas
- Display dataset shape, column names, and data types
- Generate descriptive statistics (mean, std, min, max, quartiles)
- Check for missing values and duplicates
- Verify that all 5,000 employee_ids are present and match across files
- Create distribution plots for the target variable (performance_rating)
- Document class balance (count of High, Medium, Low)

Visualization Requirements:

- Distribution histograms for all numerical features (3 separate figures, one per file)
- Box plots to identify potential outliers in each dataset
- Correlation heatmap for each file's features
- Bar chart showing performance rating distribution for each file

Deliverable: Document any initial observations about data quality, feature distributions, and potential relationships.

1.2 Data Preprocessing (6 points)

For EACH dataset separately:

Categorical Encoding:

- Encode performance_rating as numerical labels:
  - Low = 0
  - Medium = 1
  - High = 2
- Store original labels for later interpretation
- Verify encoding is consistent across all three files

Feature Scaling:

- Note which features are on different scales (e.g., client_satisfaction_score is 0-100, while most scores are 1-10)
- Document your scaling strategy (StandardScaler vs. MinMaxScaler)
- **CRITICAL**: Do NOT apply scaling yet (this will be done after train-test split to avoid data leakage)

Feature Engineering (per dataset):

- Drop employee_id from feature set (it's just an identifier, not predictive)
- Create any ratio features that might be meaningful:
  - *Structured data*: quality_per_task = average_task_quality / tasks_completed
  - *Behavioral data*: engagement_per_training = work_engagement_score / training_hours_completed
  - *Audio data*: clarity_to_speed_ratio = speech_clarity_score / speaking_speed * 100
- Document why you created these features

Deliverable: Preprocessed datasets ready for modeling, with documentation of all transformations.

1.3 Train-Test Split (2 points)

CRITICAL: Synchronized Splitting

Since all three files represent the same employees, you must create synchronized train-test splits:

# Pseudocode approach:

# 1. Use the same random_state for all three datasets

# 2. Split based on employee_id to ensure same employees in train/test across all files

# 3. Use 80-20 split ratio

# 4. Verify that train/test sets have same employee_ids across all three files

Requirements:

- Train-test split: 80% training, 20% testing
- Random state: 42 (for reproducibility)
- Stratify by performance_rating to maintain class balance
- Verify class distribution in train/test sets for each file
- **Document** that you are splitting BEFORE scaling to avoid data leakage

Deliverable: Six datasets per file (X_train, X_test, y_train, y_test for each of 3 files), totaling 18 datasets.

1.4 Model Building for Each Data Source (9 points)

Build THREE separate models (3 points each):

Model 1: Structured Data Performance Predictor

- Apply StandardScaler to X_train, transform both X_train and X_test
- Build a **Random Forest Classifier** (n_estimators=100, random_state=42)
- Train on structured_data features
- Generate predictions on both training and test sets
- Calculate performance metrics (see 1.5 below)

Model 2: Behavioral Data Performance Predictor

- Apply StandardScaler to X_train, transform both X_train and X_test
- Build a **Logistic Regression classifier** (multi-class, max_iter=1000, random_state=42)
- Train on behavior_logs features
- Generate predictions on both training and test sets
- Calculate performance metrics

Model 3: Audio Features Performance Predictor

- Apply StandardScaler to X_train, transform both X_train and X_test
- Build a **Support Vector Classifier** (kernel='rbf', C=1.0, random_state=42)
- Train on audio_features
- Generate predictions on both training and test sets

- Calculate performance metrics

Why different algorithms?

- Random Forest: Handles non-linear relationships in task/quality metrics well
- Logistic Regression: Interpretable for behavioral patterns
- SVC: Effective for audio feature patterns with complex decision boundaries

1.5 Model Evaluation (3 points)

For EACH of the three models, calculate and report:

Classification Metrics:

- Accuracy (training and test)
- Precision, Recall, F1-Score (macro-averaged)
- Confusion Matrix (visualized as heatmap)
- Classification Report (showing per-class metrics)

Create visualizations:

- Confusion matrix heatmap for each model
- Bar chart comparing train vs. test accuracy
- Feature importance plot (for Random Forest model)
- Coefficient plot (for Logistic Regression model)

Analysis Questions (answer in markdown cells):

1. Which data source (structured, behavioral, or audio) is most predictive of performance?
2. Are any models showing signs of overfitting? How can you tell?
3. Which performance class (High, Medium, Low) is hardest to predict for each model?
4. Do you see any patterns in misclassifications?

Deliverables for Phase 1

- **Jupyter Notebook** with complete code, outputs, and visualizations for all three models
- **Three trained models** saved as pickle files
- **Comparison table** showing metrics for all three individual models
- **Written analysis** (300-400 words) comparing the three data sources
- **Feature importance insights** for Random Forest model
- **Coefficient interpretation** for Logistic Regression model

Phase 2: Feature Engineering and Data Fusion (25 points)

Objective

Combine insights from all three data sources into a unified dataset and engineer features that capture cross-modal patterns.

Task Requirements

2.1 Data Merging Strategy (5 points)

Merge all three datasets:

- Merge on employee_id to create a unified dataset
- Verify merge is successful (should have 5,000 rows)
- New dataset should have: 1 ID + 24 features (8+8+8 after dropping employee_id from each) + 1 target
- Verify no data loss during merge
- Check that performance_rating is consistent across all source files (it should be)

Create source indicators:

- Add prefix to column names to indicate source:
  - struct_ for structured_data features
  - behav_ for behavior_logs features
  - audio_ for audio_features features
- Example: tasks_completed becomes struct_tasks_completed

Deliverable: Single merged dataframe with 5,000 rows and 26 columns (including employee_id and target).

2.2 Cross-Modal Feature Engineering (10 points)

Create interaction features that combine insights from different modalities:

Productivity Indicators:

- productivity_index = (struct_tasks_completed * struct_efficiency_score) / struct_hours_worked
- quality_consistency = struct_average_task_quality * behav_task_followup_score / 10

Communication Effectiveness:

- communication_quality = (audio_speech_clarity_score * audio_tone_consistency_score) / 10
- engagement_communication = behav_work_engagement_score * audio_speech_energy_level / 10

Leadership Indicators:

- leadership_score = (struct_projects_led * behav_initiative_score) / 10
- collaborative_leadership = behav_collaboration_score * behav_peer_interaction_score / 10

Client-Facing Performance:

- client_interaction_quality = struct_client_satisfaction_score * audio_speech_sentiment_score
- reliable_delivery = struct_deadline_met_score * behav_punctuality_score / 10

Sentiment-Behavior Alignment:

- sentiment_engagement_alignment = (audio_speech_sentiment_score + 1) * behav_work_engagement_score / 2
  - Note: Adding 1 to sentiment score because it ranges from -1 to 1
- energy_initiative = audio_speech_energy_level * behav_initiative_score / 10

Statistical Aggregations:

- struct_feature_mean = Mean of all structured data features
- struct_feature_std = Standard deviation of all structured data features
- behav_feature_mean = Mean of all behavioral features
- audio_feature_mean = Mean of all audio features
- cross_modal_variance = Variance across means of the three modalities

Document your reasoning:

- Explain why each interaction feature makes business sense
- Which features do you hypothesize will be most predictive?
- Are there other domain-specific features you could create?

Deliverable: Extended dataframe with 24 original features + at least 15 engineered features.

2.3 Feature Selection and Multicollinearity Analysis (10 points)

Correlation Analysis:

- Calculate correlation matrix for all features (original + engineered)
- Visualize as heatmap
- Identify highly correlated feature pairs (|correlation| > 0.85)
- Document which features are redundant

Variance Inflation Factor (VIF):

- Calculate VIF for all numerical features
- Identify features with VIF > 10 (indicates multicollinearity)
- Make decisions about feature removal
- Document which features you keep/remove and why

Feature Importance (Preliminary):

- Train a Random Forest on the merged dataset
- Extract feature importances
- Visualize top 20 features
- Select top N features based on importance threshold

Create three feature sets:

1. **Full Feature Set**: All original + engineered features
2. **Reduced Feature Set**: After removing high VIF features
3. **Top Feature Set**: Top 15-20 features by importance

Deliverable:

- Three different feature sets ready for modeling
- VIF analysis report
- Feature importance visualization
- Written justification for feature selection decisions (200-300 words)

Phase 3: Ensemble Model Building and Optimization (30 points)

Objective

Build sophisticated ensemble models that leverage the combined dataset and compare against individual models from Phase 1.

Task Requirements

3.1 Prepare Data for Ensemble Modeling (3 points)

Using your three feature sets from Phase 2.3:

- Create train-test splits (80-20, random_state=42, stratified)
- Scale features using StandardScaler (fit on train, transform train and test)
- Verify no data leakage (scaling applied AFTER split)
- Prepare all three feature sets for modeling

3.2 Build Individual Ensemble-Ready Models (12 points)

Train FOUR different algorithms on the **Reduced Feature Set**:

Model A: Random Forest Classifier

- n_estimators=200
- max_depth=15
- min_samples_split=10
- random_state=42
- Evaluate on train and test sets

Model B: Gradient Boosting Classifier

- n_estimators=100
- learning_rate=0.1
- max_depth=5
- random_state=42
- Evaluate on train and test sets

Model C: Logistic Regression (Multi-class)

- penalty='l2'
- C=1.0
- max_iter=1000
- multi_class='multinomial'
- random_state=42
- Evaluate on train and test sets

Model D: Support Vector Classifier

- kernel='rbf'
- C=10.0
- gamma='scale'
- random_state=42
- Evaluate on train and test sets

For each model, calculate:

- Training accuracy
- Test accuracy
- Precision, Recall, F1-Score (macro-averaged)

- Confusion matrix
- Training time (use time.time() to measure)

Deliverable:

- Four trained models with complete evaluation metrics
- Comparison table of all four models
- Visualization comparing their performance

3.3 Build Ensemble Models (10 points)

Ensemble 1: Voting Classifier (Hard Voting)

- Combine Random Forest, Gradient Boosting, and Logistic Regression
- Use hard voting (majority vote)
- Train on Reduced Feature Set
- Evaluate performance

Ensemble 2: Voting Classifier (Soft Voting)

- Same models as Ensemble 1
- Use soft voting (average probabilities)
- Train on Reduced Feature Set
- Evaluate performance

Ensemble 3: Stacking Classifier

- **Base models**: Random Forest, Gradient Boosting, SVC
- **Meta-model**: Logistic Regression
- Train on Reduced Feature Set
- Use cross-validation for base model predictions (cv=5)
- Evaluate performance

For each ensemble, calculate:

- Test accuracy
- Precision, Recall, F1-Score
- Confusion matrix
- Compare against individual models

Deliverable:

- Three ensemble models trained and evaluated

- Comparison showing ensemble vs. individual model performance
- Analysis of which ensemble strategy works best

3.4 Hyperparameter Tuning (5 points)

Select the best-performing ensemble from 3.3 and optimize it:

Grid Search Approach:

- Define parameter grid for the ensemble
- Use GridSearchCV with:
  - cv=5 (5-fold cross-validation)
  - scoring='f1_macro'
  - n_jobs=-1 (parallel processing)
- Search over at least 12 parameter combinations

Example parameter grid (if using Voting Classifier):

```
param_grid = {
  'rf__n_estimators': [100, 200],
  'rf__max_depth': [10, 15, 20],
  'gb__learning_rate': [0.05, 0.1],
  'gb__n_estimators': [50, 100]
}
```

Report:

- Best parameters found
- Best cross-validation score
- Performance on held-out test set
- Improvement over default parameters

Deliverable:

- Optimized ensemble model
- Grid search results visualization
- Documentation of performance improvement

Phase 4: Comprehensive Analysis and Business Insights Report (20 points)

Objective

Synthesize your findings into a business-focused report that compares all models and provides actionable insights for HR decision-making.

Report Structure

4.1 Executive Summary (3 points)

Write a **300-400 word executive summary** addressing:

- Project objective and approach
- Key findings from multimodal analysis
- Best performing model and its accuracy
- Top 5 features predicting employee performance
- Business recommendations based on model insights

Target audience: HR Director who is not technical but data-literate.

4.2 Comprehensive Model Comparison (7 points)

Create a master comparison table including ALL models:

- Phase 1: Three individual models (structured, behavioral, audio)
- Phase 3: Four individual algorithms on merged data
- Phase 3: Three ensemble models
- Phase 3: Optimized final model

Metrics to include:

- Model name and configuration
- Feature set used (which of the three sets)
- Number of features
- Training accuracy
- Test accuracy
- F1-Score (macro)
- Training time
- Overfitting indicator (train_acc - test_acc)

Analysis Questions (400-500 words total):

1. **Cross-Modal Performance**: How did models on individual data sources compare to models on merged data? Was the merged approach better? Why or why not?
2. **Feature Set Impact**: How did performance differ across Full, Reduced, and Top feature sets? Which feature selection strategy worked best?

3. **Ensemble Benefit**: Did ensemble methods outperform individual algorithms? Which ensemble strategy (voting vs. stacking) was superior? By how much?
4. **Overfitting Analysis**: Which models showed the largest train-test gap? What does this tell you about model complexity vs. generalization?
5. **Final Model Selection**: Which model would you deploy in production? Consider accuracy, interpretability, training time, and maintenance complexity.

4.3 Feature Importance Insights (5 points)

Deep-dive into what drives employee performance:

Analysis 1: Cross-Modal Importance

- Compare feature importance across the three data sources
- Create visualization showing:
  - Top 10 features from structured data
  - Top 10 features from behavioral data
  - Top 10 features from audio data
- Which modality contributes most to prediction?

Analysis 2: Engineered Feature Impact

- Which engineered features (from Phase 2.2) appear in top 20 important features?
- Were your hypotheses about useful interactions correct?
- Do interaction features outperform raw features?

Analysis 3: Business Interpretation

- Translate technical feature importance to business language
- Example: "speech_sentiment_score is the 2nd most important feature, suggesting that positive communication style is a strong indicator of high performance"
- Provide at least 5 business insights from feature importance

Deliverable:

- Feature importance visualization
- Written analysis (400-500 words)
- Ranked list of top 20 features with business interpretation

4.4 Prediction Error Analysis (3 points)

Analyze where your best model fails:

Confusion Matrix Deep-Dive:

- Visualize confusion matrix for optimized model

- Calculate per-class precision and recall
- Which performance category is hardest to predict?
- What types of misclassifications are most common?
    - High predicted as Medium?
    - Low predicted as Medium?

Error Pattern Investigation:

- Extract 10-15 examples of misclassified employees
- Examine their feature values
- Look for patterns in misclassifications
- Example: "Employees with high structured metrics but low behavioral scores are often misclassified"

Deliverable:

- Confusion matrix heatmap with annotations
- Error analysis report (300-400 words)
- Example misclassifications with explanation

4.5 Business Recommendations (2 points)

Actionable insights for HR:

Based on your model findings, provide recommendations (200-300 words):

1. **Hiring Focus**: Which attributes should HR prioritize when evaluating candidates?
2. **Performance Monitoring**: Which metrics should managers track most closely?
3. **Intervention Opportunities**: Can the model identify employees at risk of low performance early?
4. **Training Programs**: What skills/behaviors have the biggest impact on performance?
5. **Data Collection**: Should the company invest more in capturing structured, behavioral, or audio data?

Report Deliverables

- **Written report** (2000-2500 words total) in PDF or Markdown format
- **Master comparison table** with all models
- **All visualizations** professionally formatted with titles and labels
- **Feature importance analysis** with business interpretations
- **Error analysis** with specific examples

- **Business recommendations** section

General Requirements (10 points)

Code Quality and Organization (5 points)

Jupyter Notebook Standards:

- Organized into clearly labeled sections matching project phases
- Markdown headers for each major section
- Code cells have descriptive comments explaining logic
- No unnecessary code or commented-out experiments left in final version
- All visualizations have titles, axis labels, and legends
- Consistent variable naming conventions

Reproducibility:

- All random states set to 42 for reproducibility
- Clear data file paths (use relative paths)
- Required libraries listed at top of notebook
- Notebook runs from top to bottom without errors
- No hardcoded values that should be variables

Function Usage:

- Repeated code is abstracted into functions
- Functions have docstrings explaining purpose and parameters
- Example: Create a function for model evaluation that you reuse for all models

Documentation and Presentation (5 points)

In-notebook Documentation:

- Each phase has an introductory markdown cell explaining objectives
- Results are interpreted, not just displayed
- All visualizations are referenced and explained in text
- Key findings highlighted in markdown cells

Professional Formatting:

- Consistent formatting throughout
- Tables are clean and easy to read

- Visualizations use consistent color schemes
- No spelling or grammatical errors in written sections
- Proper citation of any external resources used

Reflection Section:

- Include a final reflection (200-300 words) discussing:
    - What surprised you about the results?
    - What was most challenging?
    - What would you do differently with more time?
    - What did you learn about multimodal data analysis?

Data Leakage Prevention Checklist

Verify you have NOT committed these errors:

- **Scaling applied AFTER train-test split**: StandardScaler fit only on X_train
- **No test set information in feature engineering**: Engineered features use only training data statistics
- **Synchronized splits**: Same employees in train/test across all three data files
- **No target leakage**: No features derived from or correlated with the target variable
- **Cross-validation in grid search**: GridSearchCV uses only training data
- **Feature selection on training data**: VIF and feature importance calculated on training set only
- **Stratified splitting**: Class distribution maintained in train/test splits

Submission Checklist

Before submitting, ensure you have:

Phase 1:

- Three separate models trained and evaluated (one per data source)
- Comparison of individual model performance
- Feature importance analysis for Random Forest model
- Visualizations for all three models

Phase 2:

- Merged dataset with all three data sources

- At least 15 engineered interaction features
- VIF analysis and feature selection
- Three feature sets created (Full, Reduced, Top)

Phase 3:

- Four individual algorithms trained on merged data
- Three ensemble models (hard voting, soft voting, stacking)
- Hyperparameter tuning with GridSearchCV
- Optimized final model selected

Phase 4:

- Executive summary (300-400 words)
- Master comparison table with ALL models
- Feature importance analysis with business interpretation
- Error analysis with confusion matrix
- Business recommendations section
- Reflection on learning and challenges

General:

- Code is well-commented and organized
- All visualizations are properly labeled
- No data leakage issues present
- Notebook runs from top to bottom without errors
- Report is professionally formatted
- All deliverables are included

Grading Rubric Summary

Component	**Points**	**Key Evaluation Criteria**
Phase 1: Individual Models	25	Three models built correctly, proper evaluation, meaningful comparison
Phase 2: Feature Engineering	25	Data merged correctly, creative features, proper VIF/correlation analysis
Phase 3: Ensemble Models	30	Multiple ensembles tested, hyperparameter tuning, performance improvement

Phase 4: Analysis Report	20	Comprehensive comparison, business insights, professional writing
Code Quality & Documentation	10	Clean code, reproducibility, professional presentation
TOTAL	**110**	Extra 10 points possible for exceptional work

Bonus Points Opportunities (up to +10):

- Creative and impactful engineered features (+3)
- Exceptionally thorough error analysis (+2)
- Advanced ensemble techniques (e.g., custom stacking) (+3)
- Outstanding business insights and visualizations (+2)

Important Notes

1. **Multimodal Integration is Key**: The core challenge is intelligently combining three different data perspectives. Simple concatenation is not enough - think about meaningful interactions.
2. **Business Context Matters**: You are building a tool for HR decision-making. Your model needs to be accurate AND interpretable. Balance performance with explainability.
3. **Compare Apples to Apples**: When comparing models, use the same train-test split, same random states, and same evaluation metrics. This ensures fair comparison.
4. **Document Your Journey**: Include both successes and failures. If a feature engineering idea didn't work, explain why. This shows critical thinking.
5. **Avoid Overfitting**: With 24+ features and many possible interactions, overfitting is a real risk. Use the train-test gap as your guide.
6. **Think Like a Data Scientist**: Don't just build models - understand why they work (or don't work). The "why" is more important than the "what."

Resources

Required Libraries:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier, StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
import time
import pickle
```

Recommended Reading:

- Scikit-learn Ensemble Methods: https://scikit-learn.org/stable/modules/ensemble.html
- Feature Engineering Guide: https://www.kaggle.com/learn/feature-engineering
- VIF Calculation: https://www.statsmodels.org/stable/generated/statsmodels.stats.outliers_influence.variance_inflation_factor.html
- Model Stacking Tutorial: https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/

Tips for Success:

- Start early - this project requires significant time for experimentation
- Build incrementally - test each phase before moving to the next
- Visualize everything - plots reveal patterns that numbers hide
- Save your models - you'll need to compare them later
- Ask questions - if something doesn't make sense, investigate it

Good luck! This project will challenge you to integrate everything you've learned and think like a professional ML engineer. Your goal is not just to build models, but to extract actionable business value from complex, multimodal data.