AI-Enhanced Risk Registry for ReadmitPredict: A Health Sector Project Management Portfolio

Project Overview

This project builds upon **HTH 111: Project Management in the Health Sector**, a core course in the McMaster University Health Informatics Diploma program. In HTH 111, we focused on developing comprehensive project plans for health initiatives, including risk identification, mitigation, and stakeholder engagement.

Project Summary

Preventable 30-day hospital readmissions remain a critical challenge in U.S. healthcare. Nsama (2025) estimates that avoidable readmissions contribute to approximately 30% of wasteful spending, costing the system around $17 billion annually, while also driving up penalties under the Centers for Medicare & Medicaid Services (CMS) Hospital Readmissions Reduction Program (HRRP). In fiscal year 2026, more hospitals are expected to face increased penalties as pneumonia data is reinstated in performance calculations (Eastabrook, 2025). One Community Hospital, like many systems nationwide, continues to experience elevated readmission rates that inflate costs, strain resources, and compromise patient outcomes.

To address this proactively and mitigate rising financial exposure in 2026, we propose **ReadmitPredict**: a cloud-based predictive analytics tool that enhances the validated LACE index with machine learning to deliver real-time, percentage-based readmission risk scores through an intuitive, EMR-integrated clinician dashboard. Featuring color-coded alerts (green <20%, yellow 20–50%, red >50%), factor explanations, and actionable flags, the tool empowers clinicians to make informed discharge decisions.

Inspired by successful implementations such as Allina Health — which achieved a 10.3% overall reduction in potentially preventable readmissions (including 4.3% in high-risk and 21.3% in moderate/high-risk patients) through predictive analytics and process redesign (Health Catalyst, 2025) — ReadmitPredict targets a >30% reduction in 30-day readmissions within two years post-pilot. The focused rollout in Cardiology and Internal Medicine departments delivers the following key benefits:

- Significant reduction in readmissions (>30% target), lowering CMS penalties and improving patient safety
- Cost avoidance through reduced penalties, shorter stays, and optimized resource use
- Enhanced clinician workflow with transparent, actionable risk insights

- Strategic alignment with organizational goals for quality care, efficiency, and data-driven decision-making

Triple Constraint Overview

Constraint	Target	Key Assurance
Schedule	10–12 months	Phased pilot approach with clear milestones
Budget	~$550,000	Focused on cloud-based solution and existing EMR integration
Scope	Pilot in two departments	Excludes full rollout and new hardware

With a realistic 10–12 month timeline and ~$550,000 budget, the project ensures HIPAA compliance, bias mitigation, and clinician-friendly design. ReadmitPredict positions One Community Hospital as a leader in proactive, data-driven care, delivering measurable strategic and financial value.This portfolio project extends the HTH 111 assignment by incorporating data analysis techniques to enhance risk management, demonstrating strong skills in both project management and data analytics.

Overview of Risk Management Strategies

Risk management in health projects (EHR implementations, predictive analytics tools) typically follows frameworks such as PMBOK or PRINCE2, emphasizing identification, assessment, mitigation, and monitoring. Common strategies include:

- Qualitative assessment using likelihood-impact matrices
- Quantitative modeling (Monte Carlo, machine learning)
- Stakeholder crowdsourcing to uncover blind spots
- Contingency planning for high-impact risks

In this project we adopt a creative, AI-enhanced risk registry for ReadmitPredict. We collect risks via Google Forms, simulate historical data for ML training, predict escalation probabilities, and generate mitigation suggestions using generative AI. This data-driven approach integrates traditional PM strategies with modern analytics, enabling proactive decision-making to protect the triple constraints while addressing health-specific risks (HIPAA, ML bias, clinician adoption).

Data Collection Strategy

Data collection is foundational to effective risk management, ensuring diverse inputs for comprehensive analysis.

Google Forms for Stakeholder Input

We designed an anonymous Google Form to gather risks from stakeholders (clinicians, IT, finance, etc.). The form includes fields for Risk Category, Description, Likelihood (1–5), Impact (1–5), Potential Triggers/Examples, and Mitigation Ideas, with optional Role/Department.

Form link: https://forms.gle/ZSdRNrdyVaJMbfLF9

Anonymity encourages honest feedback and aligns with health sector ethical considerations.

Registry Samples Data Creation

To populate the registry with realistic samples, we used **Grok AI** (xAI's Grok-4 model) to generate 20 factitious but plausible risk entries. Grok created balanced examples across categories (Technical, Ethical, Operational, Regulatory, Financial), with varied likelihood/impact scores and tailored triggers/mitigations based on healthcare IT contexts. This simulation strategy ensured a diverse dataset for testing without real stakeholder delays.

Historical Data Creation Strategy

To train the ML model for risk escalation prediction, we generated synthetic historical data mimicking past health tech projects.

Simple Code Explanation

The Python code uses NumPy and Pandas to create 500 rows with realistic but correlated distributions:

- Features: Project_Type, Team_Size, Complexity_Score, etc.
- Derived outcomes (Delay_Months, Budget_Overrun_Pct) incorporate correlations = Target (Risk_Escalation) defined logically with slight noise

In [1]:
```
import pandas as pd
import numpy as np
import random

np.random.seed(42)
n_projects = 300  # Simulate 300 past health tech projects (e.g., EHR/predictive to

data = {
    'Project_ID': range(1, n_projects + 1),
    'Project_Type': np.random.choice(['EHR_Implementation', 'Predictive_Analytics',
    'Team_Size': np.random.randint(5, 50, n_projects),
    'Budget_Planned_K': np.random.uniform(200, 2000, n_projects),  # in $K
    'Timeline_Months_Planned': np.random.randint(6, 24, n_projects),
    'Complexity_Score': np.random.uniform(1, 10, n_projects),  # 1=simple, 10=high
    'Stakeholder_Engagement_Level': np.random.choice(['High', 'Medium', 'Low'], n_p
    'Training_Quality': np.random.choice(['High', 'Medium', 'Low'], n_projects, p=[
    'Vendor_Reliability': np.random.choice(['High', 'Medium', 'Low'], n_projects),
    # Target variables (what we predict)
    'Budget_Overrun_Pct': np.random.normal(15, 10, n_projects).clip(0, 100),  # % o
```

```
    'Risk_Escalation': np.random.choice([0, 1], n_projects, p=[0.65, 0.35])  # 1 =
}


data['Delay_Months'] = (
    np.random.poisson(2, n_projects) +
    np.random.uniform(0, 5, n_projects) * (10 - data['Complexity_Score']) / 10
)

df = pd.DataFrame(data)

# Add some realistic correlations
df['Delay_Months'] = df['Delay_Months'] + (df['Training_Quality'].map({'Low': 4, 'M
df['Budget_Overrun_Pct'] += (df['Stakeholder_Engagement_Level'].map({'Low': 20, 'Me

df.to_csv('data/historical_healthtech_project_risks.csv', index=False)
print(df.head())
```

```
   Project_ID              Project_Type  Team_Size  Budget_Planned_K  \
0           1              Telemedicine         37       1426.904604
1           2  Clinical_Decision_Support          5        813.124225
2           3         EHR_Implementation         23        669.250151
3           4              Telemedicine          6       1092.867418
4           5              Telemedicine         48       1447.202646

   Timeline_Months_Planned  Complexity_Score Stakeholder_Engagement_Level  \
0                       11          4.647233                         High
1                        7          5.477184                       Medium
2                       16          7.481823                       Medium
3                        9          1.959209                         High
4                       20          2.260167                       Medium

  Training_Quality Vendor_Reliability  Budget_Overrun_Pct  Risk_Escalation  \
0           Medium             Medium           25.120679                0
1             High             Medium            8.000000                0
2              Low               High           25.393085                0
3           Medium             Medium           29.293243                0
4             High             Medium            9.621265                0

   Delay_Months
0      5.521430
1      3.473343
2      7.265392
3      6.237174
4      6.314558
```

Model Specifications and Threshold Usage

We used a **Random Forest Classifier** (scikit-learn) to predict Risk_Escalation (0/1).

Specifications:

- Preprocessing: OneHotEncoder for categoricals
- Model: `RandomForestClassifier(n_estimators=300, class_weight={0:3.0, 1:1.0}, max_depth=9, min_samples_leaf=5, random_state=42)`

- Evaluation: Accuracy ~0.74, macro F1 ~0.61 (focus on balanced performance given ~76% escalation rate)

Threshold tuning: Default 0.5 adjusted to **0.60** after testing 0.50–0.75 range — chosen for high recall on escalation events (minimize missed risks) while improving precision on non-escalation cases.

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report
import joblib


np.random.seed(42)
n_projects = 500  # Increase size for better learning

# Base features
data = {
    'Project_ID': range(1, n_projects + 1),
    'Project_Type': np.random.choice(['EHR_Implementation', 'Predictive_Analytics',
    'Team_Size': np.random.randint(5, 50, n_projects),
    'Budget_Planned_K': np.random.uniform(200, 2000, n_projects),
    'Timeline_Months_Planned': np.random.randint(6, 24, n_projects),
    'Complexity_Score': np.random.uniform(1, 10, n_projects),
    'Stakeholder_Engagement_Level': np.random.choice(['High', 'Medium', 'Low'], n_p
    'Training_Quality': np.random.choice(['High', 'Medium', 'Low'], n_projects, p=[
    'Vendor_Reliability': np.random.choice(['High', 'Medium', 'Low'], n_projects),
}

df = pd.DataFrame(data)

# Realistic derived outcomes (these influence escalation)
df['Delay_Months'] = (
    np.random.poisson(1.5, n_projects) +                    # base poisson noise
    (10 - df['Complexity_Score']) * 0.4 +                   # lower complexity → fewe
    df['Training_Quality'].map({'Low': 4.0, 'Medium': 1.5, 'High': 0.2}) +
    df['Stakeholder_Engagement_Level'].map({'Low': 3.5, 'Medium': 1.0, 'High': 0.3}
    df['Vendor_Reliability'].map({'Low': 3.0, 'Medium': 1.0, 'High': 0.4})
).clip(0, 15)

df['Budget_Overrun_Pct'] = (
    np.random.normal(12, 8, n_projects) +
    (df['Complexity_Score'] - 5) * 2.5 +                    # higher complexity → mor
    df['Training_Quality'].map({'Low': 18, 'Medium': 6, 'High': -2}) +
    df['Stakeholder_Engagement_Level'].map({'Low': 15, 'Medium': 5, 'High': -3})
).clip(0, 80)

# Now define target logically (escalation if delay >3 months OR overrun >20%)
df['Risk_Escalation'] = ((df['Delay_Months'] > 4.5) | (df['Budget_Overrun_Pct'] > 2
```

```
#add some pure randomness/noise so not everything is deterministic
df['Risk_Escalation'] = df['Risk_Escalation'] ^ np.random.choice([0,1], size=n_proj

# Features and target
X = df.drop(['Project_ID', 'Delay_Months', 'Budget_Overrun_Pct', 'Risk_Escalation']
y = df['Risk_Escalation']

cat_cols = ['Project_Type', 'Stakeholder_Engagement_Level', 'Training_Quality', 'Ve

preprocessor = ColumnTransformer(
    transformers=[('cat', OneHotEncoder(handle_unknown='ignore'), cat_cols)],
    remainder='passthrough'
)

model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=300, max_depth=9, class_weig
])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
model.fit(X_train, y_train)

preds = model.predict(X_test)
probs = model.predict_proba(X_test)[:, 1]

# Save model
joblib.dump(model, 'models/risk_predictor_model.pkl')

print("Accuracy:", accuracy_score(y_test, preds))
print(classification_report(y_test, preds))
print("\nClass distribution:", y.value_counts(normalize=True))
```

```
Accuracy: 0.744
              precision    recall  f1-score   support

           0       0.37      0.40      0.38        25
           1       0.85      0.83      0.84       100

    accuracy                           0.74       125
   macro avg       0.61      0.61      0.61       125
weighted avg       0.75      0.74      0.75       125


Class distribution: Risk_Escalation
1    0.762
0    0.238
Name: proportion, dtype: float64
```

In [3]:

```
# Try different thresholds
for thresh in [0.5, 0.6, 0.65, 0.7, 0.75]:
    preds_custom = (probs >= thresh).astype(int)
    print(f"Threshold {thresh:.2f}:")
    print(classification_report(y_test, preds_custom, zero_division=0))
    print("-"*50)
```

```
Threshold 0.50:
              precision    recall  f1-score   support

           0       0.37      0.40      0.38        25
           1       0.85      0.83      0.84       100

    accuracy                           0.74       125
   macro avg       0.61      0.61      0.61       125
weighted avg       0.75      0.74      0.75       125

--------------------------------------------------
Threshold 0.60:
              precision    recall  f1-score   support

           0       0.39      0.68      0.49        25
           1       0.90      0.73      0.81       100

    accuracy                           0.72       125
   macro avg       0.64      0.71      0.65       125
weighted avg       0.80      0.72      0.74       125

--------------------------------------------------
Threshold 0.65:
              precision    recall  f1-score   support

           0       0.33      0.72      0.46        25
           1       0.90      0.64      0.75       100

    accuracy                           0.66       125
   macro avg       0.62      0.68      0.60       125
weighted avg       0.79      0.66      0.69       125

--------------------------------------------------
Threshold 0.70:
              precision    recall  f1-score   support

           0       0.29      0.84      0.43        25
           1       0.92      0.48      0.63       100

    accuracy                           0.55       125
   macro avg       0.61      0.66      0.53       125
weighted avg       0.80      0.55      0.59       125

--------------------------------------------------
Threshold 0.75:
              precision    recall  f1-score   support

           0       0.24      0.96      0.38        25
           1       0.96      0.24      0.38       100

    accuracy                           0.38       125
   macro avg       0.60      0.60      0.38       125
weighted avg       0.82      0.38      0.38       125

--------------------------------------------------
```

Mapping Data Model to Response Data

To apply the model to form responses, we map qualitative fields to training features:

- Fixed project values: Project_Type = 'Predictive_Analytics', Team_Size = 15, Budget = 550, Timeline = 11
- Mapped: Stakeholder_Engagement_Level from Role/Department (e.g., IT/Compliance = 'High')
- Derived: Complexity_Score = ((Likelihood + Impact)/2 × category_boost).clip(1,10)

In [4]:
```
import pandas as pd

# Load the Excel file
df_responses = pd.read_excel('data/Responses.xlsx')

print("Shape:", df_responses.shape)
print("Columns:", df_responses.columns.tolist())
print("\nFirst few rows:\n")
print(df_responses.head(5))
```

```
Shape: (20, 8)
Columns: ['Timestamp', 'Risk Category', 'Risk Description', 'Likelihood', 'Impact',
'Your Role/Department', 'Potential Triggers / Examples: ', 'Mitigation Ideas: ']

First few rows:

                Timestamp Risk Category  \
0 2026-01-28 15:04:32.359     Technical
1 2026-01-28 15:07:48.820      Ethical
2 2026-01-28 15:09:21.265  Operational
3 2026-01-28 15:11:10.590    Regulatory
4 2026-01-28 15:12:15.766     Financial

                                 Risk Description  Likelihood  Impact  \
0  Delays or failures in integrating ReadmitPredi...           4       5
1  Model may produce inaccurate risk scores for r...           3       4
2  Clinicians resist using the new dashboard due ...           4       4
3  Changes to CMS HRRP rules (e.g., pneumonia rei...           3       5
4  Budget overrun due to unexpected cloud computi...           3       4

  Your Role/Department                   Potential Triggers / Examples:   \
0                   IT  Vendor API changes mid-project, mismatched dat...
1    Internal Medicine  Training data skewed toward urban hospital adm...
2           Cardiology  Dashboard alerts interrupt rounding, or color-...
3           Compliance  2026 policy update expands metrics, requiring ...
4              Finance  Higher-than-planned API calls during testing o...

                                Mitigation Ideas:
0  Conduct early API compatibility testing with v...
1  Perform regular bias audits using fairness too...
2  Run clinician-led demo sessions and feedback l...
3  Build modular model architecture for fast retr...
4  Set monthly burn-rate tracking dashboard, prio...
```

In [5]:
```
# Create df_model with EXACTLY the columns the model expects
required_columns = [
    'Project_Type',
    'Team_Size',
    'Budget_Planned_K',
    'Timeline_Months_Planned',
    'Complexity_Score',
    'Stakeholder_Engagement_Level',
    'Training_Quality',
    'Vendor_Reliability'
]

df_model = pd.DataFrame(index=df_responses.index)  # same number of rows

# Required categorical columns (must match training exactly)
df_model['Project_Type'] = 'Predictive_Analytics'  # most similar to ReadmitPredict
df_model['Vendor_Reliability'] = 'Medium'          # default

df_model['Training_Quality'] = df_responses['Risk Category'].map({
    'Operational': 'Low',      # e.g. adoption/training risks imply weaker training
    'Technical': 'Medium',
    'Ethical': 'High',         # bias mitigation might require strong training
    'Financial': 'Medium',
    'Regulatory': 'High'
}).fillna('Medium')

engagement_map = {
    'IT': 'High',
    'Compliance': 'High',
    'Internal Medicine': 'High',
    'Cardiology': 'High',
    'Finance': 'Medium',
    'Prefer not to say': 'Medium',
    # Add fallback
}

df_model['Stakeholder_Engagement_Level'] = df_responses['Your Role/Department'].map

# Optional: print the distribution to check it looks reasonable
print(df_model['Stakeholder_Engagement_Level'].value_counts())
```

```
Stakeholder_Engagement_Level
High      15
Medium     5
Name: count, dtype: int64
```

In [6]:
```
# Numeric columns - derive where possible
#if 'Likelihood' in df_responses.columns and 'Impact' in df_responses.columns:
    #df_model['Complexity_Score'] = (df_responses['Likelihood'] + df_responses['Imp
#else:
    #df_model['Complexity_Score'] = 5.0  # neutral default

# Instead of /2 *2, use a stronger multiplier or add category boost
base_complex = (df_responses['Likelihood'] + df_responses['Impact']) / 2
category_boost = df_responses['Risk Category'].map({
    'Technical': 1.4,      # integration is complex
```

```
    'Regulatory': 1.5,
    'Ethical': 1.3,
    'Operational': 1.2,
    'Financial': 1.0
}).fillna(1.0)

df_model['Complexity_Score'] = (base_complex * category_boost).clip(1, 10)

#Add small random/project-specific variation
np.random.seed(42)
df_model['Complexity_Score'] += np.random.uniform(-1.5, 1.5, len(df_model))
df_model['Complexity_Score'] = df_model['Complexity_Score'].clip(1, 10)

# Add other numerics if you have proxies (or use defaults)
df_model['Team_Size'] = 15                    # average for pilot project
df_model['Budget_Planned_K'] = 550            # from your exec summary
df_model['Timeline_Months_Planned'] = 11      # mid of 10-12 months

# Copy over useful original columns for later reference
df_model['Original_Risk_Description'] = df_responses.get('Risk Description', '')
df_model['Original_Category'] = df_responses.get('Risk Category', '')
df_model['Likelihood'] = df_responses.get('Likelihood', 3)
df_model['Impact'] = df_responses.get('Impact', 3)
df_model['Triggers'] = df_responses.get('Potential Triggers / Examples', '')
df_model['Mitigation'] = df_responses.get('Mitigation Ideas', '')

print("Prepared input shape:", df_model.shape)
print(df_model.head(3))
```

```
Prepared input shape: (20, 14)
           Project_Type Vendor_Reliability Training_Quality  \
0  Predictive_Analytics             Medium           Medium
1  Predictive_Analytics             Medium           Medium
2  Predictive_Analytics             Medium           Medium

  Stakeholder_Engagement_Level  Complexity_Score  Team_Size  Budget_Planned_K  \
0                         High          5.923620         15               550
1                         High          4.852143         15               550
2                         High          4.695982         15               550

   Timeline_Months_Planned                          Original_Risk_Description  \
0                       11  Delays or failures in integrating ReadmitPredi...
1                       11  Model may produce inaccurate risk scores for r...
2                       11  Clinicians resist using the new dashboard due ...

  Original_Category  Likelihood  Impact Triggers Mitigation
0         Technical           4       5
1          Ethical            3       4
2      Operational            4       4
```

Model Fitting

In [7]:

```
import joblib

# Load the trained pipeline
```

```
model = joblib.load('models/risk_predictor_model.pkl')   # adjust path/filename

# Predict probabilities and custom threshold
probs = model.predict_proba(df_model)[:, 1]   # probability of escalation (class 1)

df_model['Escalation_Probability_%'] = probs * 100
df_model['Predicted_Escalation'] = (probs >= 0.60).astype(int)  # your chosen thres

# Optional: Classic manual risk score for comparison
df_model['Manual_Risk_Score'] = df_model['Likelihood'] * df_model['Impact']
df_model['Risk_Level'] = pd.cut(
    df_model['Escalation_Probability_%'],
    bins=[0, 40, 65, 100],
    labels=['Low', 'Medium', 'High'],
    include_lowest=True
)

# Save enriched results
df_model['Score_Discrepancy'] = df_model['Escalation_Probability_%'] - (df_model['L

df_model.to_excel('data/Final_ReadmitPredict_Risk_Predictions.xlsx', index=False)

print("\nPredictions complete! Sample output:\n")
print(df_model[['Original_Category', 'Original_Risk_Description', 'Manual_Risk_Scor
               'Escalation_Probability_%', 'Predicted_Escalation', 'Risk_Level']].
```

```
Predictions complete! Sample output:

  Original_Category                          Original_Risk_Description  \
0         Technical  Delays or failures in integrating ReadmitPredi...
1          Ethical   Model may produce inaccurate risk scores for r...
2      Operational   Clinicians resist using the new dashboard due ...
3        Regulatory  Changes to CMS HRRP rules (e.g., pneumonia rei...
4         Financial  Budget overrun due to unexpected cloud computi...
5         Technical  Data security breach during cloud-based real-t...
6      Operational   Staff turnover delays training and knowledge t...
7          Ethical   Over-reliance on tool leads to reduced clinica...

   Manual_Risk_Score  Escalation_Probability_%  Predicted_Escalation  \
0                 20                 63.438470                     1
1                 12                 60.675323                     1
2                 16                 61.010617                     1
3                 15                 37.603948                     0
4                 12                 63.608403                     1
5                 10                 59.395771                     0
6                 12                 63.467175                     1
7                 12                 60.483472                     1

  Risk_Level
0     Medium
1     Medium
2     Medium
3        Low
4     Medium
5     Medium
6     Medium
7     Medium
```

AI Mitigation Strategy and Code

We integrated **generative AI** (OpenAI GPT-4o-mini via ChatGPT subscription) to auto-generate 3–4 mitigation suggestions per risk.

Strategy

- Context-aware prompts include risk description, triggers, budget (~$550K), and timeline (10–12 months)
- Outputs are supplementary — subject to human review
- Results added as new column and visualized in dashboard

In [8]:
```python
import os
import pandas as pd
from openai import OpenAI
from dotenv import load_dotenv


# Load your data (Responses.xlsx or the enriched one)
df = pd.read_excel('data/Responses.xlsx')  # or 'Responses_with_Predictions.xlsx'
```

```
load_dotenv("process.env")

# Initialize OpenAI client (automatically uses your env variable)
client = OpenAI()  # If you hardcoded: OpenAI(api_key="sk-your-key-here")

# Choose a strong model (your subscription gives access to these)
MODEL = "gpt-4o-mini"   # fast & cheap — great for bulk
# Alternatives: "gpt-4o" (more powerful), "o1-mini" (strong reasoning)

def generate_mitigation_suggestions(row):
    risk_desc = row['Risk Description'] if 'Risk Description' in row else row.get('
    triggers  = row['Potential Triggers / Examples'] if 'Potential Triggers / Examp

    prompt = f"""
    You are an expert in healthcare project risk management.
    For the following risk in a hospital readmission prediction tool project (Readm

    Risk: {risk_desc}
    Potential triggers: {triggers}

    Generate 3-4 concise, practical mitigation strategies or actions.
    Focus on feasibility within a 10-12 month pilot budget of ~$550K.
    Number them 1-4 and keep each suggestion 1-2 sentences.
    """

    try:
        response = client.chat.completions.create(
            model=MODEL,
            messages=[{"role": "user", "content": prompt}],
            temperature=0.7,   # 0.0 = deterministic, 0.7 = creative but safe
            max_tokens=300
        )
        return response.choices[0].message.content.strip()

    except Exception as e:
        return f"Error: {str(e)}"

# Apply to every row (may take a few seconds per row depending on volume)
print("Generating AI mitigation suggestions...")
df['AI_Generated_Mitigations'] = df.apply(generate_mitigation_suggestions, axis=1)

# Save the updated file
output_file = 'data/Responses_with_AI_Mitigations.xlsx'
df.to_excel(output_file, index=False)

print(f"Done! Results saved to: {output_file}")
print("\nSample:")
print(df[['Risk Description', 'AI_Generated_Mitigations']].head(3))
```

```
Generating AI mitigation suggestions...
Done! Results saved to: data/Responses_with_AI_Mitigations.xlsx

Sample:
                                    Risk Description  \
0  Delays or failures in integrating ReadmitPredi...
1  Model may produce inaccurate risk scores for r...
2  Clinicians resist using the new dashboard due ...

                            AI_Generated_Mitigations
0  1. **Stakeholder Engagement**: Establish a cro...
1  1. **Diverse Data Collection**: Ensure that th...
2  Here are four concise mitigation strategies fo...
```

Dashboard Overview

An interactive dashboard was built using **Plotly Dash** to visualize the risk registry.

Key components:

- Category filter
- KPI cards (total risks, high risks, avg AI probability, % flagged)
- Risk heat map (Likelihood × Impact scatter)
- Word cloud from AI mitigations
- Box plot: AI probability by category
- Pie chart: Flagged vs not flagged
- Top 10 words bar chart from AI mitigations
- Detailed table with conditional formatting

The dashboard demonstrates how data analysis supports project management tasks: real-time prioritization, theme identification, and stakeholder communication.

Dashboard link on Plotly Cloud: https://c97adaa2-b827-4a32-ac7d-a7faec385ac2.plotly.app/

In [9]:
```python
import dash
from dash import dcc, html, dash_table
from dash.dependencies import Input, Output
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import io
import base64
from collections import Counter
import re

# Load your enriched data
df_ai = pd.read_excel('data/Responses_with_AI_Mitigations.xlsx')
df = pd.read_excel('data/Final_ReadmitPredict_Risk_Predictions.xlsx')
```

```
df['AI_Generated_Mitigations'] = df_ai['AI_Generated_Mitigations']

# Helper: generate word cloud as base64
def generate_wordcloud(text_column):
    if text_column.empty or text_column.isna().all():
        return ""
    text = ' '.join(text_column.dropna().astype(str))
    wordcloud = WordCloud(
        width=800, height=400,
        background_color='white',
        min_font_size=10, max_font_size=150,
        colormap='viridis'
    ).generate(text)

    img = io.BytesIO()
    wordcloud.to_image().save(img, format='PNG')
    img.seek(0)
    return "" + base64.b64encode(img.read()).decode()

# Helper: get top 10 words from AI mitigations
def get_top_words(text_column, n=10):
    text = ' '.join(text_column.dropna().astype(str)).lower()
    # Clean text: remove punctuation, numbers, short words
    words = re.findall(r'\b[a-z]{3,}\b', text)
    # Remove common stop words (simple list)
    stop_words = {'the', 'and', 'for', 'with', 'the', 'to', 'in', 'of', 'a', 'on',
    words = [w for w in words if w not in stop_words]

    counter = Counter(words)
    top = counter.most_common(n)
    if not top:
        return pd.DataFrame({'word': [], 'count': []})
    return pd.DataFrame(top, columns=['word', 'count'])

# Dash app
app = dash.Dash(__name__)

app.layout = html.Div([
    html.H1("ReadmitPredict Risk Registry Dashboard", style={'textAlign': 'center',
    html.Hr(),

    # Filter
    html.Div([
        html.Label("Filter by Category:", style={'fontWeight': 'bold'}),
        dcc.Dropdown(
            id='category-filter',
            options=[{'label': cat, 'value': cat} for cat in sorted(df['Original_Ca
            value=None,
            placeholder="All categories",
            multi=True,
            style={'width': '100%'}
        ),
    ], style={'width': '60%', 'margin': '20px auto'}),

    # KPI cards
    html.Div([
```

```
        html.Div(id='kpi-total', className='kpi-card'),
        html.Div(id='kpi-high', className='kpi-card'),
        html.Div(id='kpi-avg-prob', className='kpi-card'),
        html.Div(id='kpi-flagged', className='kpi-card'),
    ], style={'display': 'flex', 'justifyContent': 'center', 'gap': '20px', 'margin

    # Row 1: Risk Matrix + Word Cloud
    html.Div([
        dcc.Graph(id='risk-matrix', style={'width': '50%', 'display': 'inline-block
        html.Div([
            html.H3("Word Cloud – AI Mitigation Themes", style={'textAlign': 'cente
            html.Img(id='wordcloud', style={'width': '90%', 'margin': 'auto', 'disp
        ], style={'width': '50%', 'display': 'inline-block', 'verticalAlign': 'top'
    ], style={'display': 'flex', 'margin': '20px'}),

    # Row 2: Box plot + Pie + Top 10 Words Bar
    html.Div([
        dcc.Graph(id='prob-by-category', style={'width': '33%', 'display': 'inline-
        dcc.Graph(id='escalation-pie', style={'width': '33%', 'display': 'inline-bl
        dcc.Graph(id='top-words-bar', style={'width': '33%', 'display': 'inline-blo
    ], style={'display': 'flex', 'margin': '20px'}),

    # Detailed table
    html.H3("Risk Details", style={'textAlign': 'center'}),
    dash_table.DataTable(
        id='risk-table',
        columns=[
            {"name": "Category", "id": "Original_Category"},
            {"name": "Description", "id": "Original_Risk_Description"},
            {"name": "Manual Score", "id": "Manual_Risk_Score"},
            {"name": "AI Probability %", "id": "Escalation_Probability_%"},
            {"name": "Flagged", "id": "Predicted_Escalation"},
            {"name": "AI Mitigations", "id": "AI_Generated_Mitigations"}
        ],
        style_table={'overflowX': 'auto'},
        style_cell={'textAlign': 'left', 'padding': '10px', 'whiteSpace': 'normal',
        style_data_conditional=[
            {'if': {'filter_query': '{Predicted_Escalation} eq 1'}, 'backgroundColo
            {'if': {'filter_query': '{Risk_Level} eq "High"'}, 'color': '#c0392b',
        ],
        page_size=12
    ),
], style={'padding': '20px', 'fontFamily': 'Arial, sans-serif'})

# Callback
@app.callback(
    [
        Output('risk-matrix', 'figure'),
        Output('wordcloud', 'src'),
        Output('prob-by-category', 'figure'),
        Output('escalation-pie', 'figure'),
        Output('top-words-bar', 'figure'),
        Output('risk-table', 'data'),
        Output('kpi-total', 'children'),
        Output('kpi-high', 'children'),
        Output('kpi-avg-prob', 'children'),
```

```
        Output('kpi-flagged', 'children'),
    ],
    [Input('category-filter', 'value')]
)
def update_dashboard(selected_categories):
    dff = df.copy()
    if selected_categories:
        dff = dff[dff['Original_Category'].isin(selected_categories)]

    # Risk matrix
    fig_matrix = px.scatter(
        dff,
        x='Impact',
        y='Likelihood',
        size='Manual_Risk_Score',
        color='Risk_Level',
        hover_data=['Original_Risk_Description', 'Escalation_Probability_%', 'Origi
        color_discrete_map={'Low': '#27ae60', 'Medium': '#f39c12', 'High': '#c0392b
        title='Risk Heat Map (bubble size = Manual Score)'
    )
    fig_matrix.update_layout(xaxis_range=[0.5, 5.5], yaxis_range=[0.5, 5.5], height

    # Word cloud
    wc_src = generate_wordcloud(dff['AI_Generated_Mitigations'])

    # Probability by category (box)
    fig_prob = px.box(
        dff, x='Original_Category', y='Escalation_Probability_%',
        color='Original_Category', title='AI Probability by Category',
        height=450, width=650
    )
    fig_prob.update_layout(showlegend=False)

    # Pie: Flagged vs Not
    flagged_counts = dff['Predicted_Escalation'].value_counts().reset_index()
    flagged_counts.columns = ['Flagged', 'Count']
    flagged_counts['Flagged'] = flagged_counts['Flagged'].map({1: 'Flagged', 0: 'No
    fig_pie = px.pie(
        flagged_counts, values='Count', names='Flagged',
        title='AI-Flagged Risks (≥65%)',
        color_discrete_sequence=['#e74c3c', '#3498db'],
        height=450, width=650
    )

    # Top 10 words bar chart
    top_words_df = get_top_words(dff['AI_Generated_Mitigations'], 10)
    fig_top_words = px.bar(
        top_words_df,
        x='count',
        y='word',
        orientation='h',
        title='Top 10 Words in AI Mitigation Suggestions',
        text='count',
        color='count',
        color_continuous_scale='Blues'
    )
```

```
    fig_top_words.update_layout(
        height=450, width=650,
        xaxis_title="Frequency",
        yaxis_title="Word",
        showlegend=False
    )
    fig_top_words.update_traces(textposition='auto')

    # KPIs
    total = len(dff)
    high = len(dff[dff['Risk_Level'] == 'High'])
    avg_prob = round(dff['Escalation_Probability_%'].mean(), 1) if total > 0 else 0
    pct_flagged = round((len(dff[dff['Predicted_Escalation'] == 1]) / total * 100),

    kpi_total   = html.Div([html.H4("Total Risks"),   html.H2(total)],   className=
    kpi_high    = html.Div([html.H4("High Risks"),    html.H2(high)],    className=
    kpi_avg     = html.Div([html.H4("Avg AI Prob"),   html.H2(f"{avg_prob}%")], cla
    kpi_flagged = html.Div([html.H4("% Flagged"),     html.H2(f"{pct_flagged}%")],

    return (
        fig_matrix,
        wc_src,
        fig_prob,
        fig_pie,
        fig_top_words,
        dff.to_dict('records'),
        kpi_total, kpi_high, kpi_avg, kpi_flagged
    )

# CSS for KPI cards (same as before)
app.index_string = '''
<!DOCTYPE html>
<html>
    <head>
        {%metas%}
        <title>ReadmitPredict Dashboard</title>
        {%favicon%}
        {%css%}
        <style>
            .kpi-card {
                background: #f8f9fa;
                border: 1px solid #ddd;
                border-radius: 8px;
                padding: 15px 25px;
                min-width: 160px;
                text-align: center;
                box-shadow: 0 2px 5px rgba(0,0,0,0.1);
            }
            .kpi-card h4 { margin: 0; font-size: 14px; color: #555; }
            .kpi-card h2 { margin: 8px 0 0; font-size: 28px; }
        </style>
    </head>
    <body>
        {%app_entry%}
        <footer>{%config%}{%scripts%}{%renderer%}</footer>
    </body>
```

```
</html>
'''

#if __name__ == '__main__':
    #app.run(debug=True, mode ='inline')

if __name__ == '__main__':
    app.run(debug=True, mode ='inline')
```

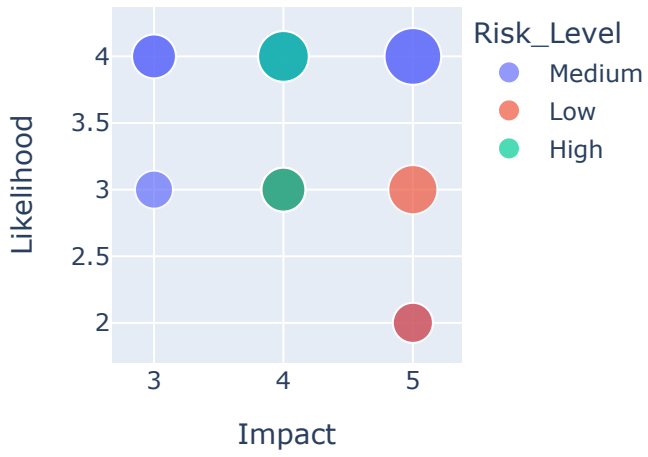
ReadmitPredict Risk Registry Dashboard

Filter by Category

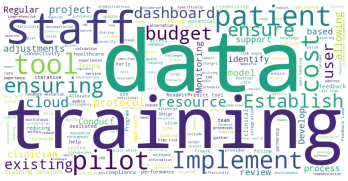
All Categories

Total Risks	High Risks	Avg AI Prob	% Flagged
20	2	58.2%	60.0%

Risk Heat Map



Word Cloud – AI Mitigation Themes



Code Best Practices, Model Tuning, and Acknowledgments

Code Best Practices

- Modular structure: separate files for data generation, model training, prediction, dashboard

- Reproducibility: fixed random seeds `(np.random.seed(42))`
- Error handling: try-except blocks for API calls, NaN checks
- Documentation: comments, README.md, clear variable names
- Version control: GitHub repository structure recommended

Model Tuning

- Addressed class imbalance with `class_weight={0:3.0, 1:1.0}`
- Tuned threshold to 0.60 after evaluating 0.50–0.75 range
- Focused on macro F1 and recall on escalation class (healthcare priority: avoid missing risks)

Acknowledgments

This project was significantly enhanced by Grok AI (xAI's Grok-4 model), which provided guidance on data generation strategies, code structure, model tuning suggestions, dashboard design, and generative AI integration.

References

Eastabrook, D. (2025). More hospitals to face high readmission penalties in fiscal 2026. AIMHI. Retrieved January 23, 2026 from https://aimhi.mobi/more-hospitals-to-face-high-readmission-penalties-in-fiscal-2026/

Nsama, F. (2025). Assessing the Cost-Containment Effectiveness of AI-Based Predictive Models in Reducing Avoidable Readmissions and Overtreatment in US Medicare Hospitals. Applied Sciences, Computing, and Energy, 2(2), 452-466