**KB Article User Interest Analysis**

**Project Overview**

This project aims to analyze user behavior data and recommend KB articles to users:

1. Which KB articles users are interested in.
2. How job functions and content types influence KB article views.
3. The performance of machine learning models in predicting user behavior.

**Dataset**

The dataset contains:

* User-related information (job function, content type, etc.).
* KB article metadata (article number, description, etc.).
* Interaction details (whether a user viewed a specific KB article).

**Summary of Findings**

**1. Descriptive Analysis**

* **Most Viewed Content Types**: Certain content types (e.g., "CORE", "SUPPORT") dominate user views.
* **Job Functions**: Users from the top 10 job functions contribute to the majority of KB views.
* **Unique KB Articles**:
  + The distribution of unique KB articles varies significantly across content types.
  + A small subset of content types contributes to most KB article views.

**2. Confusion Matrix Interpretation**

* **True Negatives (2484)**: The model correctly identified non-views.
* **True Positives (0)**: The model failed to identify any KB views.
* **Class Imbalance**:
  + Severe imbalance between classes (0: Not viewed, 1: Viewed) impacted model performance.
  + Class 1 (viewed) constitutes only a small fraction of the data.

A graph with blue squares and numbers

Description automatically generated

**3. Model Performance**

* Models Evaluated:
  + Random Forest, Logistic Regression, Support Vector Machine (SVM).
* **Random Forest**:
  + Best-performing model with mean accuracy of **53%** and low standard deviation (**0.235**).
  + Handles imbalanced data better but still struggles with minority class prediction.
* **Logistic Regression**:
  + Performed poorly with mean accuracy of **32%**, highlighting difficulty with non-linear patterns.
* **Support Vector Machine**:
  + Moderate performance (**39% accuracy**) but with high variability (**0.331 standard deviation**).

A screenshot of a graph

Description automatically generated

**4. Recommendations**

* **Address Class Imbalance**:
  + Apply techniques like SMOTE or class-weight adjustments.
* **Feature Engineering**:
  + Add user engagement metrics and richer features for better model performance.
* **Hyperparameter Tuning**:
  + Improve Random Forest and SVM using GridSearchCV.
* **Alternative Models**:
  + Explore Gradient Boosting models (e.g., XGBoost, LightGBM) for better handling of imbalanced data.

**Next Steps**

1. Implement class balancing techniques to improve minority class prediction.
2. Perform hyperparameter tuning for top-performing models.
3. Develop visual dashboards for easier data exploration.